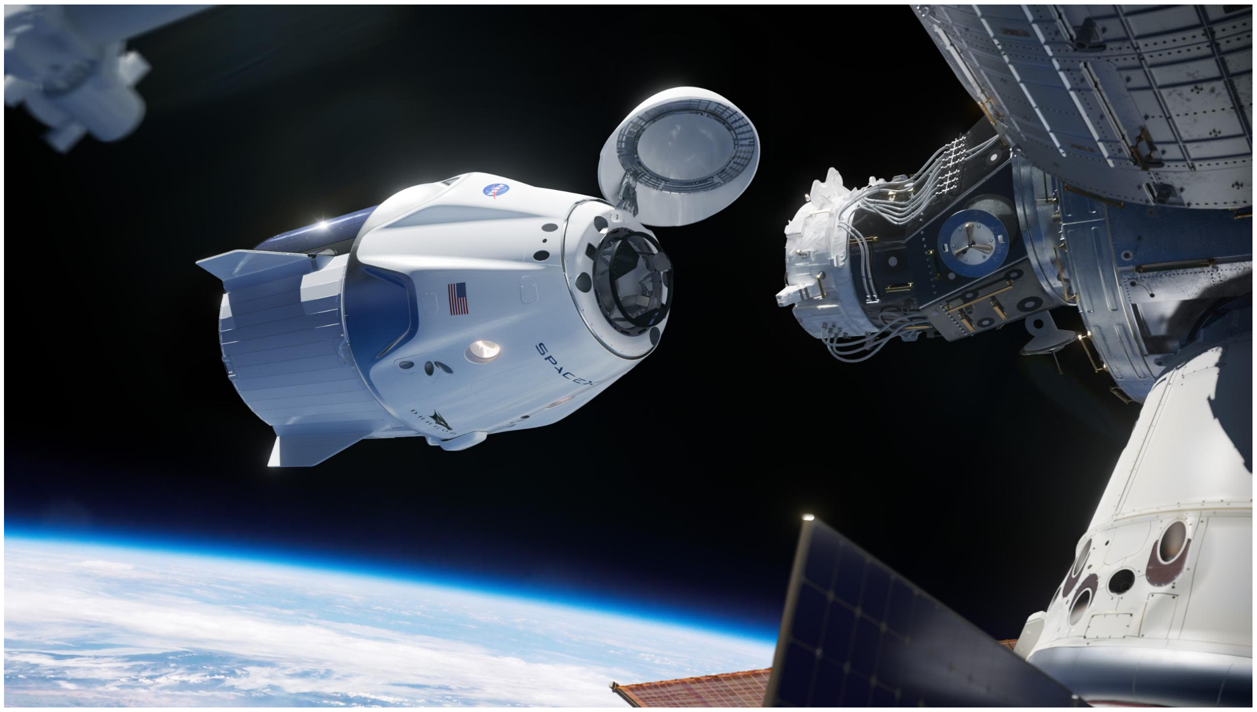
# Project DRAGON-DOCK

Dragon Rendezvous And Gateway for Orbital Navigation and DOCKing

**Alexandre Carlhammar, Brian Check**



AA 279D - Spacecraft Formation-Flying and Rendezvous
Stanford University

## Revision History (Problem Sets 1-8)

| Rev | Changes |
|---|---|
| PS1 | - Created document<br>- Added problem set 1 material |
| PS2 | - Added problem set 2 material<br>- Redid our problem statement and question 1 in PS1<br>- Existing PS1 code already worked for new problem, did not need to redo<br>- Removed question 2 answers from PS1; we were told we didn't need to redo our answers for PS1 |
| PS3 | - Added problem set 3 material<br>- Fixed integrator error from PS2<br>- Added PS3 code to Appendix for easier grading |
| PS4 | - Added problem set 4 material |
| PS5 | - Added problem set 5 material<br>- Added problem set 5 code to Appendix |
| PS6 | - Added problem set 6 material |
| PS7 | - Added problem set 7 material<br>- Re-did impulsive control analysis |
| PS8 | - Added problem set 8 material |

## Revision History 2 (Problem Set 9 - Final Submission)

| Rev | Changes |
|---|---|
| PS1 | - Edited all sections of question 1 to reflect mission changes<br>- Completed question 2 (previously incomplete due to different mission)<br>- Added code to Appendix |
| PS2 | - Redid all questions and figures with corrected nonlinear propagator<br>- Previous results were invalid due to propagator malfunction<br>- Added code to Appendix |
| PS3 | - Updated STM in 2c) for YA matrix to be more accurate<br>- Consolidated code in Appendix |
| PS4 | - Q1: Updated orbital elements to reflect new maneuver<br>- Q3: Redid entirely with corrections<br>- Q4 + Q5: Updated plots with analysis<br>- Q6: Redid entirely<br>- Q7: Updated plots<br>- Q8: Added second plot and re-plotted with corrected STM<br>- Added code to Appendix |
| PS5 | - Q1: Rewrote ”Orbit Modeling Dynamics Approach” section<br>- Q2: Completely redid analysis (previously submitted in PS7)<br>- Q2: Rewrote ”Performance and Results” section |
| PS6 | - Q2: Updated orbital elements values and format (QNSROE) to reflect new maneuver<br>- Q2: Deleted simplified Lyapunov model and replaced with formal, nonlinear definition<br>- Q2: Updated graphs with corrected Lyapunov control law<br>- Q2: Updated results analysis |
| PS7 | - Q2: Updated EKF sensitivity matrix to account for long range QNSROE estimation |
| PS8 | - Q2: Corrected UKF navigation filter formulation<br>- Q2: Corrected variable values with latest<br>- Q2: Updated existing graphs with corrected UKF implementation<br>- Q2: Added error graphs with $\pm 3\Sigma$ bounds<br>- Q2: Added measurement residual graphs<br>- Q2: Added navigation statistic table<br>- Q2: Updated results analysis |
| PS9 | - Added PS9 material and code for all questions |
| Appendix | - Added ’Utils (Conversion Functions)’, ’Propagation Functions’, and ’Control Code Helper Functions’ to Appendix for functions used persistently throughout Problem Sets<br>- Reorganized code by problem set and added all code for each problem set<br>- Created new GitHub link with organized files |
| References | - Added academic references used across PS11 to PS8<br>- Linked to respective PSETS sections where used |

## Contents

## List of Figures

# List of Tables

## 0 Scope

This report covers the requirements for AA279D Spacecraft Formation Flying and Rendezvous project.

## 1 Problem Set 1

The code used for this assignment is available in our GitHub repository [1] and the Appendix. See ”Problem Set 1 Code” for code for each section. Additionally, see ”Propagators” and ”Utils” sections.

### 1.1 Your Mission, Your Challenge

**Mission Name:** *DRAGON-DOCK* (Dragon Rendezvous And Gateway for Orbital Navigation and DOCKing)
**Operators:** NASA (in partnership with SpaceX)
**Nation(s):** United States
**Companies Involved:** SpaceX (Dragon)

#### 1.1.1 Primary Mission Objective:

- Successfully rendezvous and dock the Dragon capsule to the ISS docking port

#### 1.1.2 Secondary Mission Objectives:

- Validate Dragon’s autonomous proximity-operations software in the ISS approach corridor
- Demonstrate fault-tolerant abort and retreat maneuvers during final approach
- Verify spacecraft-to-station data and power interfaces post-docking

#### 1.1.3 Number and Type of Spacecraft

- 1x Dragon Capsule (cargo or crew variant)
- 1x International Space Station

#### 1.1.4 Basic Description of Functioning / Scientific Principle

The Dragon performs a controlled phasing maneuver from its insertion orbit to a co-orbital approach corridor. Using GPS, lidar for inter satellite distance link, and camera, it transitions very close to the ISS before executing the final docking. Upon contact, the ISS’s station arm captures or the docking mechanism latches, establishing a hard-mated interface for cargo transfer and power/data exchange.

### 1.1.5 Key DGN&C Requirements

- Autonomous guidance accuracy: $\pm 1$ m in position, $\pm 0.1°$ in attitude during final approach
- Proximity operations hold at 30 m and 10 m with station-keeping better than 0.5 m
- Relative navigation redundancy: GPS + optical fiducials + rendezvous lidar
- Real-time collision avoidance and abort capability within 5 s
- Fault detection and recovery for GN&C sensors and thruster actuators

### 1.1.6 Classification of DSS

**Type:** Cooperative Rendezvous and Docking
**Separation Class:** Close (0-30 m)
**Required Control Accuracy:**

- Pre-capture hold: $\sim 0.5$ m positional, $\sim 0.5°$ angular
- Final contact: $< 0.1$ m positional, $< 0.1°$ angular

## 1.2 Orbit Simulation, Review of Astrodynamics

### 1.2.1 a) Initial conditions

**Intial Orbital Elements:**

| Spacecraft | $a$ (km) | $e$ | $i$ (deg) | $\Omega$ (deg) | $\omega$ (deg) | $\nu$ (deg) |
|---|---|---|---|---|---|---|
| Chief (ISS) | 6771 | 0.0005 | 51.64 | 257 | 0 | 30 |
| Deputy (Dragon) | 6771 | 0.0015 | 52.14 | 258 | 0.5 | 25 |

Table 1.1: Initial absolute Keplerian orbital elements for chief and deputy

| ROE Component | Initial Value |
|---|---|
| $a_d - a_c$ | 0 |
| $e_d - e_c$ | 0.001 |
| $i_d - i_c$ (deg) | 0.5 |
| $\Omega_d - \Omega_c$ (deg) | 1 |
| $\omega_d - \omega_c$ (deg) | 0.5 |
| $\nu_d - \nu_c$ (deg) | -5 |

Table 1.2: Initial deputy relative Keplerian orbital elements (ROEs)

**Launch Time and Duration:**

- Launch Date: July 2026
- Duration: Phasing and approach (1-2 days), docked operations (2-4 weeks)

### 1.2.2 b) Initial ECI positions and velocities

The initial conditions are treated as osculating Keplerian elements

$$\mathbf{x}_{\mathrm{OE}} = [a,\, e,\, i,\, \Omega,\, \omega,\, \nu],$$

and converted to an inertial Cartesian state vector $\mathbf{x}_{\mathrm{ECI}} = [\mathbf{r};\, \mathbf{v}]$ using the `OE2ECI` function found in the `utils` section (see Appendix).

First, the position and velocity in the perifocal frame are computed via conic equations:

$$\mathbf{r}_p = \frac{p}{1 + e\cos\nu}\begin{bmatrix}\cos\nu \\ \sin\nu \\ 0\end{bmatrix}, \quad \mathbf{v}_p = \sqrt{\frac{\mu}{p}}\begin{bmatrix}-\sin\nu \\ e + \cos\nu \\ 0\end{bmatrix}, \quad p = a(1 - e^2).$$

Next, a rotation from perifocal to ECI is applied using a 3-1-3 Euler sequence:

$$\mathbf{x}_{\mathrm{ECI}} = R_3(\Omega) \cdot R_1(i) \cdot R_3(\omega) \cdot \mathbf{x}_{\mathrm{perifocal}},$$

where each $R_k(\theta)$ is a standard rotation matrix about axis $k \in \{1, 3\}$. The full transformation is implemented in the `OE2ECI` function, which outputs the inertial position and velocity.

$$\mathbf{r}_{chief} = 10^6 \times \begin{bmatrix}0.7278 \\ -6.1835 \\ 2.6535\end{bmatrix} \text{ m}, \qquad \mathbf{v}_{chief} = 10^3 \times \begin{bmatrix}4.8833 \\ 2.8098 \\ 5.2133\end{bmatrix} \text{ m/s},$$

$$\mathbf{r}_{deputy} = 10^6 \times \begin{bmatrix}0.4787 \\ -6.3412 \\ 2.2983\end{bmatrix} \text{ m}, \qquad \mathbf{v}_{deputy} = 10^3 \times \begin{bmatrix}4.8510 \\ 2.3459 \\ 5.4766\end{bmatrix} \text{ m/s}.$$

### 1.2.3 c) Positional propagation

To simulate the motion of the spacecraft in the inertial frame, the osculating initial conditions from part (b) were converted to Cartesian state vectors $[\,\mathbf{r}_0;\, \mathbf{v}_0\,]$, and numerically integrated using `ode113` with the state derivative defined in `getStatedot` (see Appendix, propagators section).

The acceleration model includes both the central two-body term and optional $J_2$ perturbations:

$$\ddot{\mathbf{r}} = -\frac{\mu}{r^3}\mathbf{r} + \mathbf{a}_{J_2},$$

where the $J_2$ acceleration $\mathbf{a}_{J_2}$ is computed via a separate utility function if `perturbated` is true.

Simulations were run both with and without $J_2$, over a time span of 10 orbital periods to capture secular effects such as nodal and apsidal drift. The full trajectory in the inertial frame was then plotted to visualize the impact of perturbations.

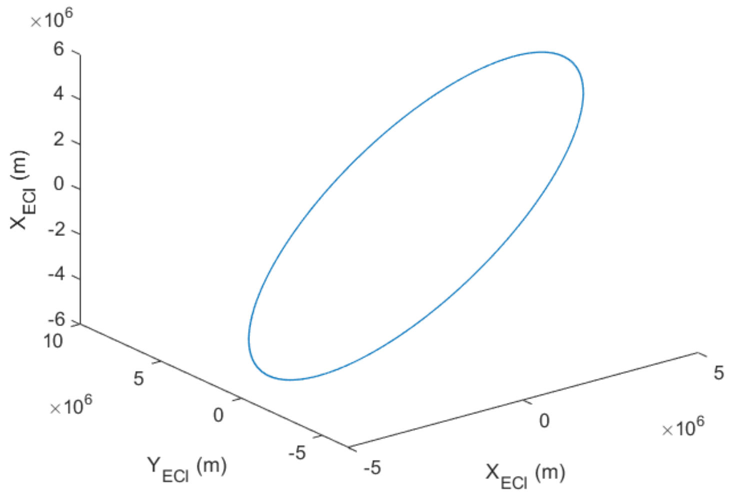


Figure 1.1: Trajectory of the chief spacecraft in ECI frame without J2 effects over 10 orbital periods

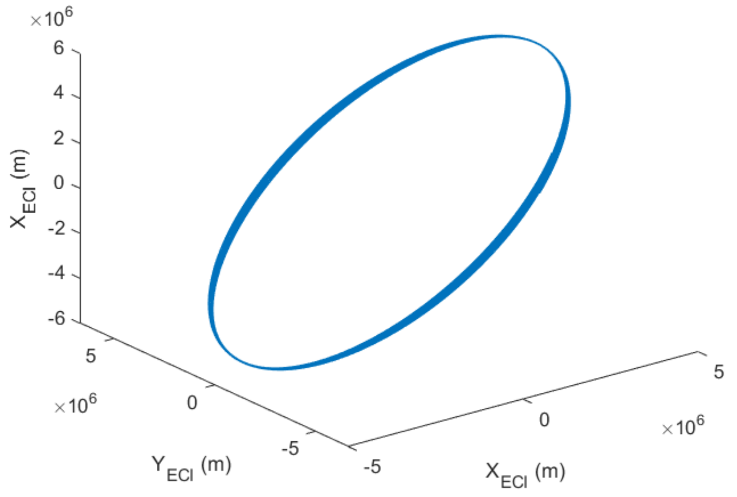


Figure 1.2: Trajectory of the chief spacecraft in ECI frame with J2 effects over 10 orbital periods

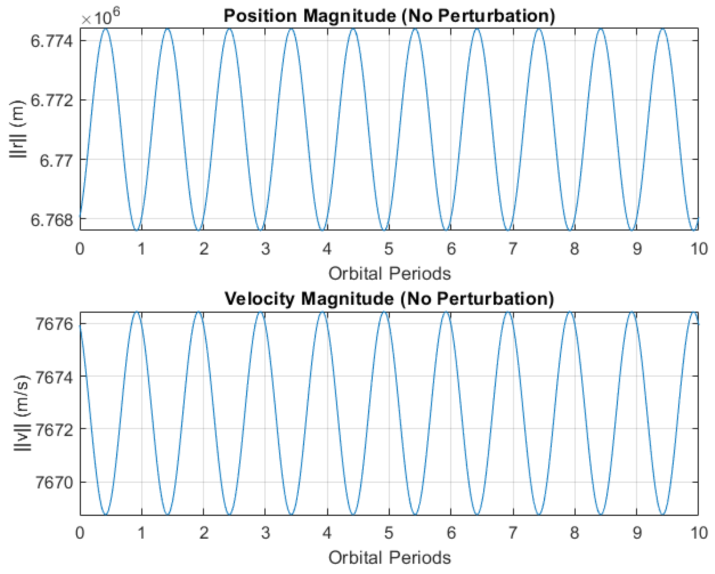


Figure 1.3: Chief radius and velocity magnitudes in ECI without J2 effects over 10 orbital periods

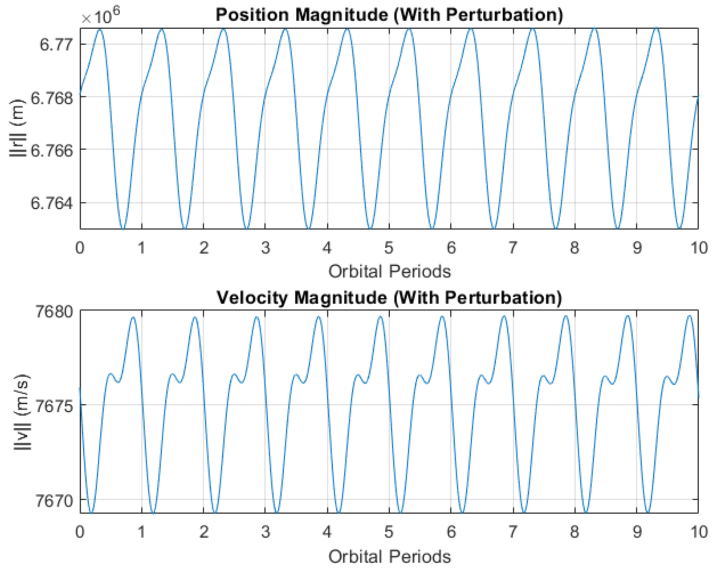


Figure 1.4: Chief radius and velocity magnitudes in ECI with J2 effects over 10 orbital periods

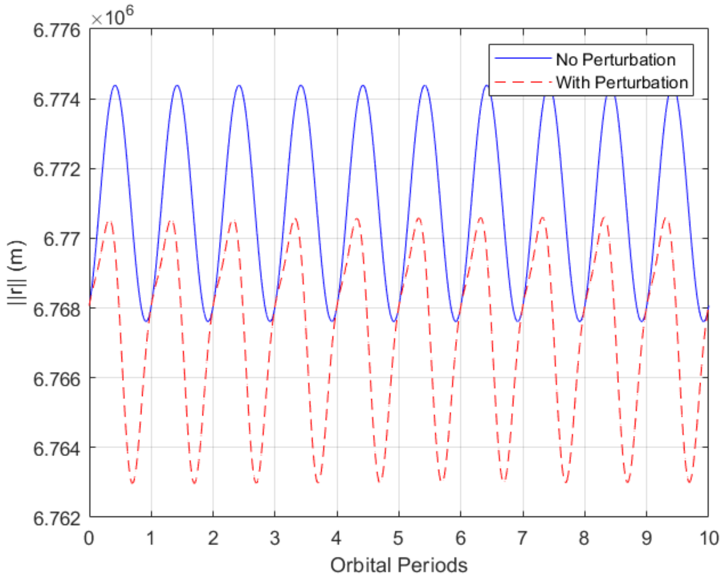


Figure 1.5: Chief radius in ECI with and without J2 effects over 10 orbital periods

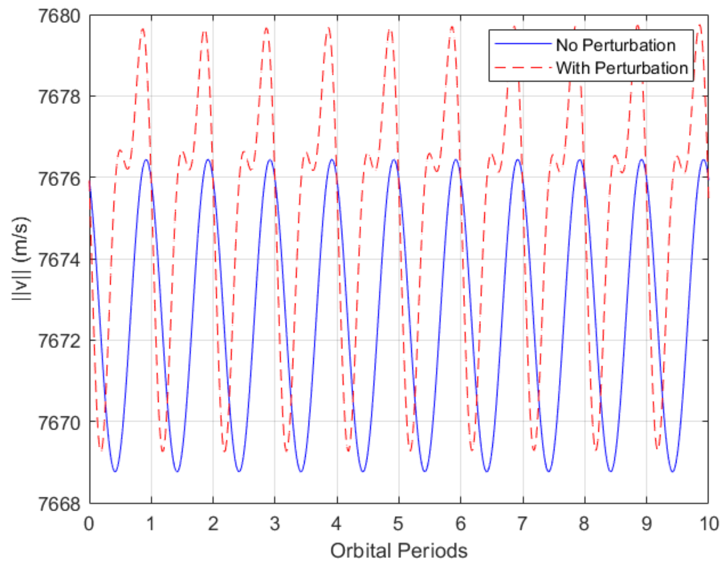


Figure 1.6: Chief velocity in ECI with and without J2 effects over 10 orbital periods

### 1.2.4 d) Comparison of positional and Keplerian propagators

To verify that the Cartesian integrator performs correctly, I compared its output (from part c, without $J_2$) to the analytical Keplerian propagator (part d). Both were initialized with identical osculating elements.

At each time step, the analytical state $\mathbf{x}_{\text{OE}}(t)$ was converted to Cartesian form $\mathbf{x}_{\text{ECI}}^{\text{analytic}}(t)$ using `OE2ECI`, and compared to the numerically integrated ECI state $\mathbf{x}_{\text{ECI}}^{\text{num}}(t)$ by transforming the difference into the RTN frame using `ECI2RTN` (see Appendix, utils section).

This yielded position and velocity errors:

$$\delta \mathbf{x}_{\mathrm{RTN}} = \begin{bmatrix} \mathbf{r}_{\mathrm{analytic}} - \mathbf{r}_{\mathrm{num}} \\ \\ \mathbf{v}_{\mathrm{analytic}} - \mathbf{v}_{\mathrm{num}} \end{bmatrix}_{\mathrm{RTN}}$$

Only small numerical integration errors were observed, as expected. Reducing the step size confirmed convergence and validated the implementation. See the "Problem Set 1" and "Utils" sections in the Appendix for how RTN error calculation was done.

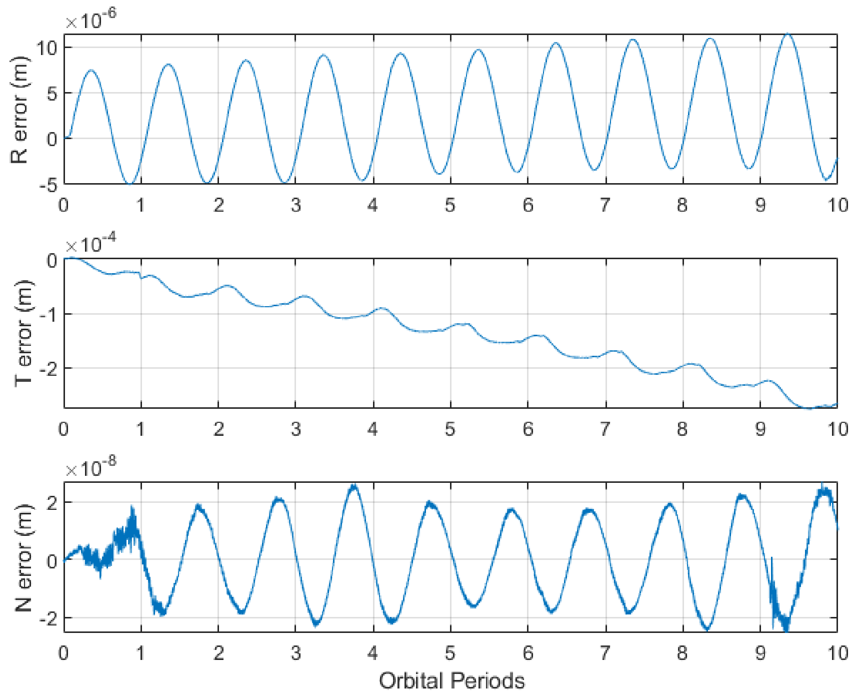


Figure 1.7: RTN positional error over 10 orbits for Keplerian vs. Positional propagators

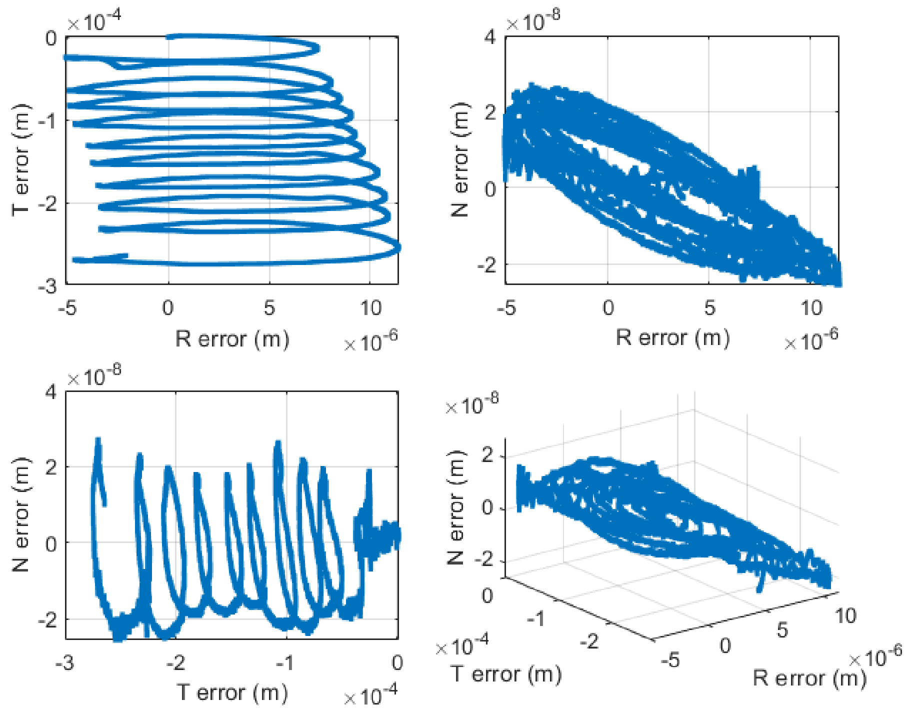


Figure 1.8: RTN positional error over 10 orbits for Keplerian vs. Positional propagators

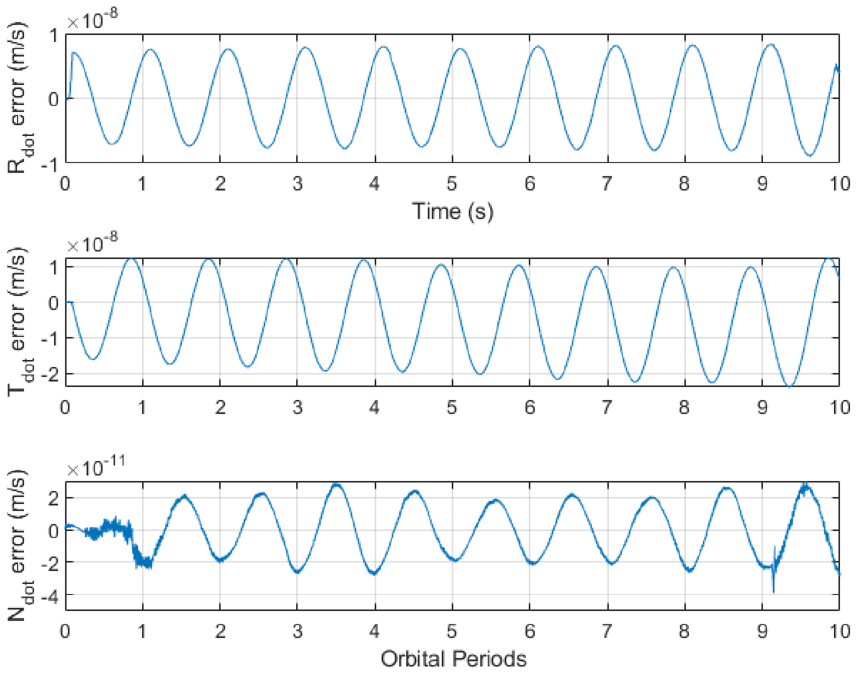


Figure 1.9: RTN velocity error over 10 orbits for Keplerian vs. Positional propagators

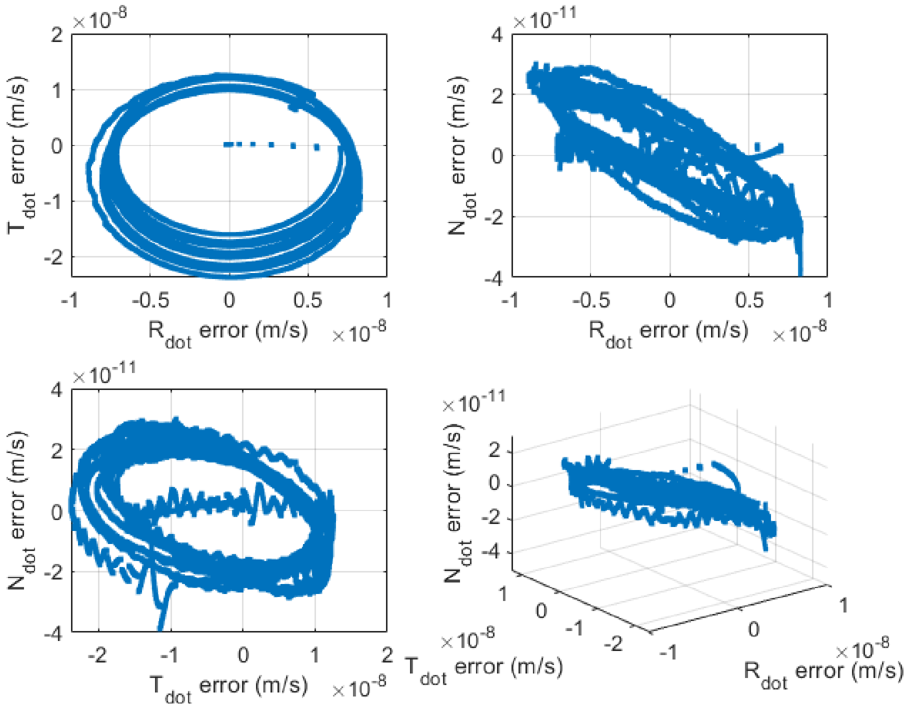


Figure 1.10: RTN velocity error over 10 orbits for Keplerian vs. Positional propagators

### 1.2.5 e) J2 Osculating elements over time

See the "Problem Set 1" and "Propagators" sections in the Appendix for how propagation was done.

To track orbital evolution, the osculating Keplerian elements were propagated using `ode113` with and without $J_2$ perturbations. At each timestep, the propagated elements were converted to inertial position and velocity using `OE2ECI`, and then used to compute:

- The eccentricity vector:
$$\mathbf{e} = \frac{1}{\mu}\left[(\|\mathbf{v}\|^2 - \frac{\mu}{\|\mathbf{r}\|})\mathbf{r} - (\mathbf{r}\cdot\mathbf{v})\mathbf{v}\right]$$
- The angular momentum vector: $\mathbf{h} = \mathbf{r} \times \mathbf{v}$
- The specific mechanical energy: $\epsilon = \frac{1}{2}\|\mathbf{v}\|^2 - \frac{\mu}{\|\mathbf{r}\|}$

Magnitudes of these vectors were plotted to visualize secular trends and conservation behavior under different dynamical models.

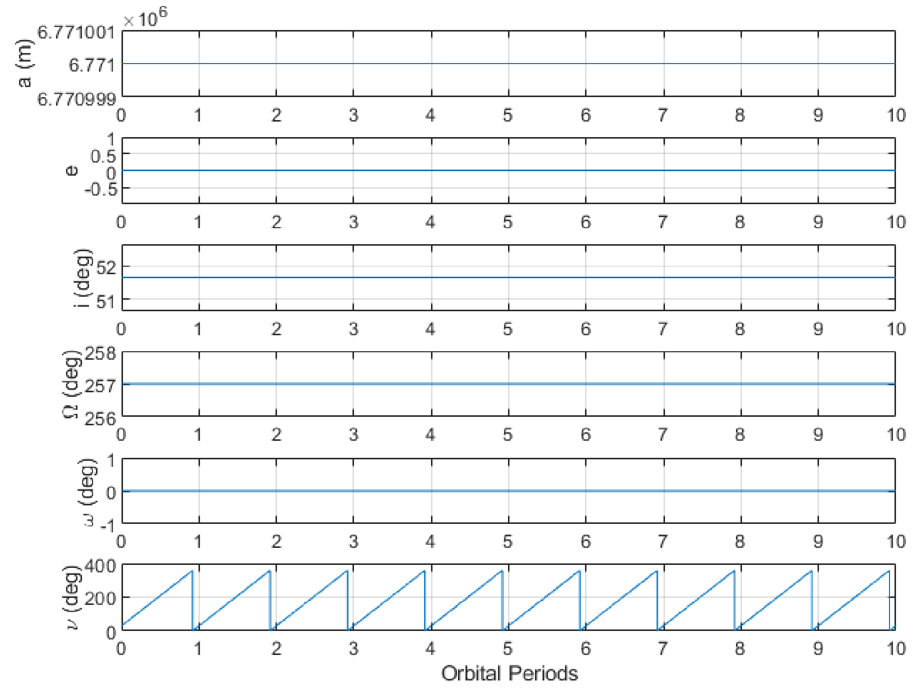


Figure 1.11: Unperturbed Keplerian orbital elements over 10 periods

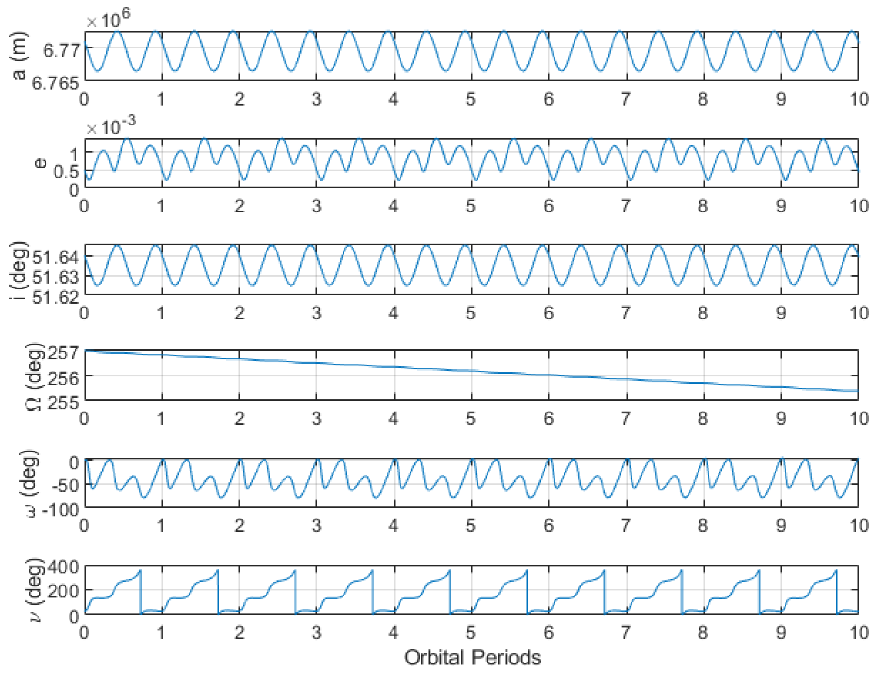


Figure 1.12: Perturbed Keplerian orbital elements over 10 periods

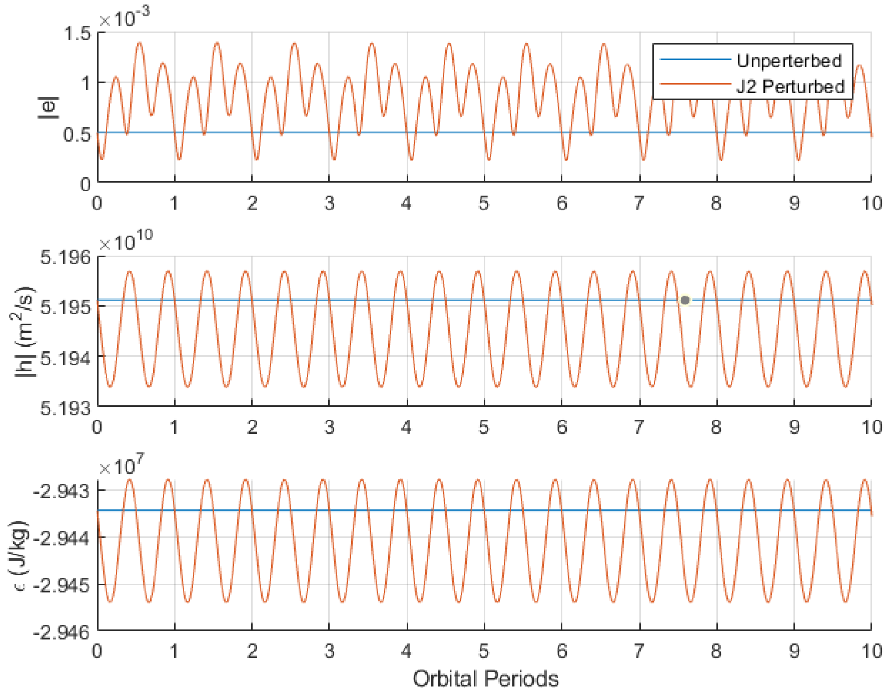


Figure 1.13: Eccentricity (e), Angular Momentum (h), and Specific Mechanical Energy ($\epsilon$) over 10 periods (perturbed and unperturbed)

**Unperturbed ($\mathbf{J_2 = 0}$)**

- $a$, $e$, $i$, $\Omega$, $\omega$ remain exactly constant.
- $\nu$ advances uniformly, wrapping each 360°.
- Conserved quantities $|\mathbf{e}|$, $|\mathbf{h}|$, $\epsilon$ are perfectly flat.

**Perturbed ($\mathbf{J_2 \neq 0}$)**

- $a$ and $\epsilon$ oscillate but exhibit no secular drift (energy-preserving).
- $e$, $i$ undergo small, bounded oscillations. $\omega$ undergoes a larger bounded oscillation.
- $\Omega$ regresses steadily (nodal precession) as predicted by

$$\dot{\Omega} \approx -\tfrac{3}{2} J_2 \left(\tfrac{R}{p}\right)^2 n \cos i \, .$$

- $\nu$ advances nonlinearly, reflecting the rotating $\omega$.
- $|\mathbf{e}|$, $|\mathbf{h}|$, $\epsilon$ show only short-period variations, no drift.

**All results agree with standard $\mathbf{J_2}$ averaging theory.**

### 1.2.6 f) J2 Mean Element Propagation

To model secular evolution, the function `getMeanStatedot` (see Appendix, propagators section) integrates the Lagrange planetary equations averaged over one orbit. It computes the time derivatives of mean elements $[a, e, i, \Omega, \omega, \nu]$ using the closed-form $J_2$-perturbed rates:

$$\frac{d\Omega}{dt} = -\frac{3}{2} n J_2 \left(\frac{R^2}{p^2}\right) \cos i$$
$$\frac{d\omega}{dt} = \frac{3}{4} n J_2 \left(\frac{R^2}{p^2}\right) (5\cos^2 i - 1)$$
$$\frac{dM}{dt} = n + \frac{3}{4} n J_2 \left(\frac{R^2}{p^2}\right) \sqrt{1-e^2}(3\cos^2 i - 1)$$

These were numerically integrated and compared with osculating element propagation from part (e). The results were superimposed, showing good agreement in long-term trends after several orbits, confirming consistency between orbit-averaged and instantaneous dynamics.

However, while the qualitative trends matched (e.g., secular drift in $\omega$ and $\Omega$), noticeable offsets remained due to the difference in initialization; mean element propagation assumes orbit-averaged inputs, while osculating propagation uses instantaneous states.

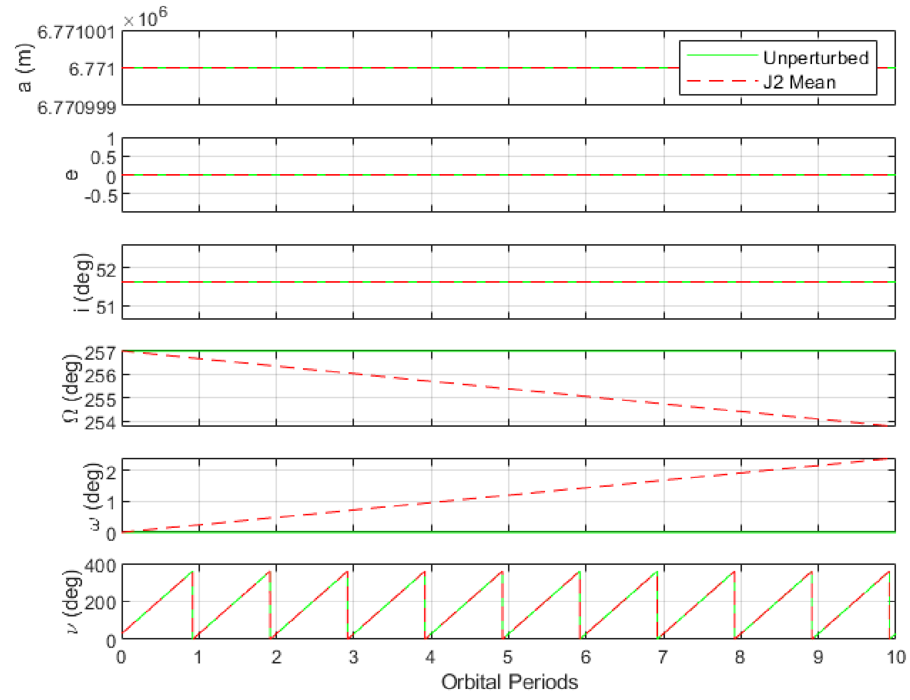


Figure 1.14: Unperturbed vs. J2 mean orbital elements over 10 periods

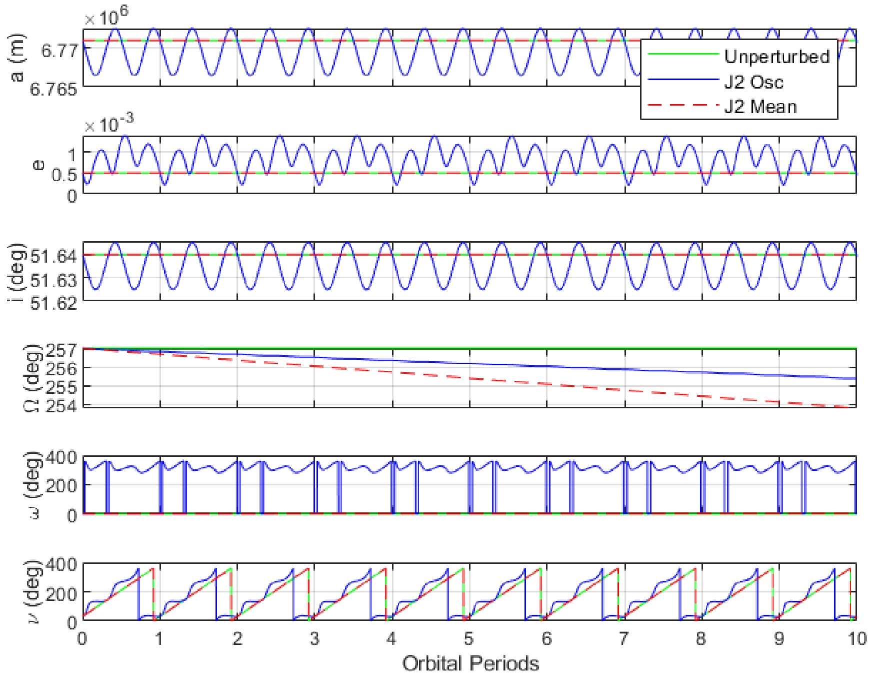


Figure 1.15: Unperturbed vs. J2 mean vs. J2 oscillating orbital elements over 10 periods

**g) Reconciling Osculating vs. Mean Initialization**

To ensure both the osculating-element solver (part c) and the mean-element solver (part f) start from the same physical orbit, one must transform the initial osculating elements into their corresponding mean elements. Two standard methods are:

1. **First-order Brouwer transformation (analytic):**
   Use Brouwer's J$_2$ theory (e.g. Lecture 2 slides 29-30) to compute the short-period corrections $\delta x_{\text{sp}}$ and set
   $$x_{\text{mean}}(0) \;=\; x_{\text{osc}}(0) \;-\; \delta x_{\text{sp}}\,, \quad x \in \{a, e, i, \Omega, \omega\},$$
   then convert $\nu_0 \mapsto M_0$. The inverse mapping (mean $\to$ osculating) is obtained by replacing $J_2 \to -J_2$ in the same formulas.

2. **Numerical orbit-averaging (quadrature):**
   $$x_{\text{mean}}(0) = \frac{1}{T}\int_0^T x_{\text{osc}}(t)\,\mathrm{d}t, \qquad x \in \{a, e, i, \Omega, \omega\},$$
   $$M_0 = \frac{1}{T}\int_0^T M_{\text{osc}}(t)\,\mathrm{d}t\,.$$
   Here $T$ is one orbital period under J$_2$, and $M_{\text{osc}}(t)$ is obtained via $\nu \to E \to M$. This guarantees the mean solver in part f) reproduces exactly the secular trend of the osculating run in part e).

## 2 Problem Set 2

The code used for this assignment is available in our GitHub repository [1] and the Appendix. See "Problem Set 2 Code" for code for each section. Additionally, see "Propagators" and "Utils" sections.

## Everything is Relative

### 2.1 a) Deputy Initial Orbital Elements

| $a$ | $e$ | $i$ | $\Omega$ | $\omega$ | $\nu$ |
|---|---|---|---|---|---|
| 6771 km | 0.0015 | 52.14° | 257.5° | 0.5° | 25° |

Table 2.1: Deputy initial Keplerian orbital elements

Changes:

- e increased by 0.001
- i, $\Omega$, and $\omega$ increased by 0.5°
- $\nu$ decreased by 5°
- **Change from pset 1**: $\Omega$ only increased by 0.5° rather than 1°

### 2.2 b) Deputy Position and Velocity in Chief's Rotating RTN Frame

The nonlinear relative motion in the RTN frame is governed by the following coupled system, implemented in the function `nonlinear_state_dot`. The state consists of the chief's inertial position and velocity, and the deputy's RTN position and velocity, all combined into a 12-dimensional state vector.

**Chief ECI Integration:**

$$\dot{\mathbf{r}}_0 = \mathbf{v}_0$$
$$\dot{\mathbf{v}}_0 = -\frac{\mu}{r_0^3}\mathbf{r}_0$$

**Deputy RTN Integration:**

$$\dot{x}_1 = v_{x_1}, \qquad \dot{y}_1 = v_{y_1}, \qquad \dot{z}_1 = v_{z_1}$$
$$\dot{v}_{x_1} = 2\dot{\theta}_0 v_{y_1} + \ddot{\theta}_0 y_1 + \dot{\theta}_0^2 x_1 - \frac{\mu(r_0 + x_1)}{[(r_0 + x_1)^2 + y_1^2 + z_1^2]^{3/2}} + \frac{\mu}{r_0^2}$$
$$\dot{v}_{y_1} = -2\dot{\theta}_0 v_{x_1} - \ddot{\theta}_0 x_1 + \dot{\theta}_0^2 y_1 - \frac{\mu y_1}{[(r_0 + x_1)^2 + y_1^2 + z_1^2]^{3/2}}$$
$$\dot{v}_{z_1} = -\frac{\mu z_1}{[(r_0 + x_1)^2 + y_1^2 + z_1^2]^{3/2}}$$

The angular rate and its derivative are computed from the chief's motion:

$$\dot{\theta}_0 = \frac{h}{r_0^2}, \qquad \ddot{\theta}_0 = \frac{-2\dot{r}_0\dot{\theta}_0}{r_0}$$

where $h = \|\mathbf{r}_0 \times \mathbf{v}_0\|$, and $\dot{r}_0 = \frac{d}{dt}\|\mathbf{r}_0\|$.

These equations are numerically integrated using MATLAB's `ode113` over ten orbital periods. Figures 1–4 show the deputy's position and velocity in the rotating RTN frame as computed from the nonlinear propagation.

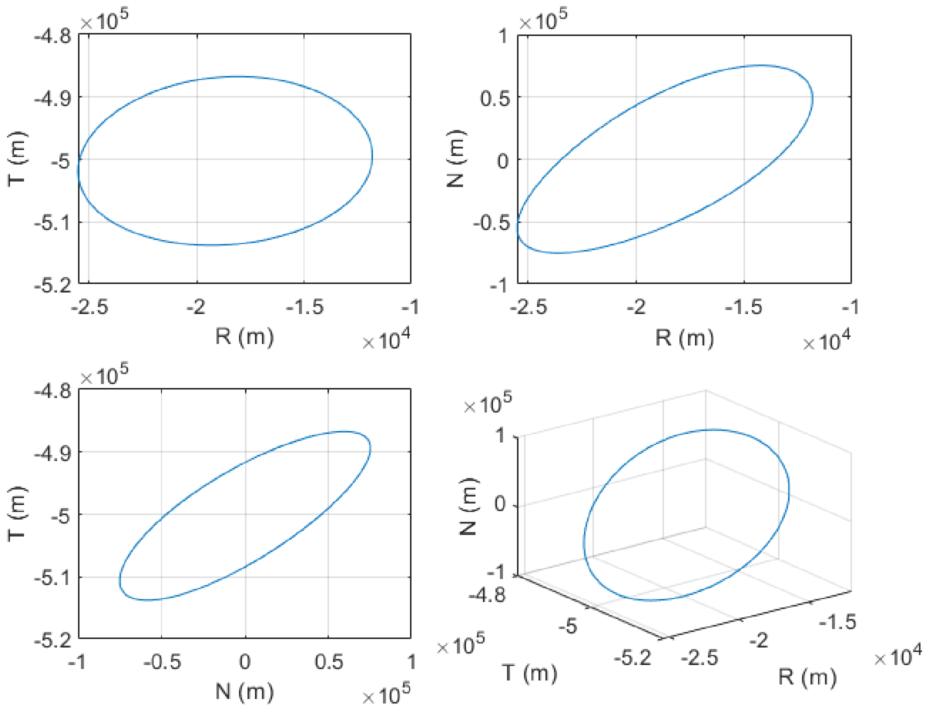


Figure 2.1: Deputy Position in Chief's Rotating RTN Frame over 10 Orbital Periods

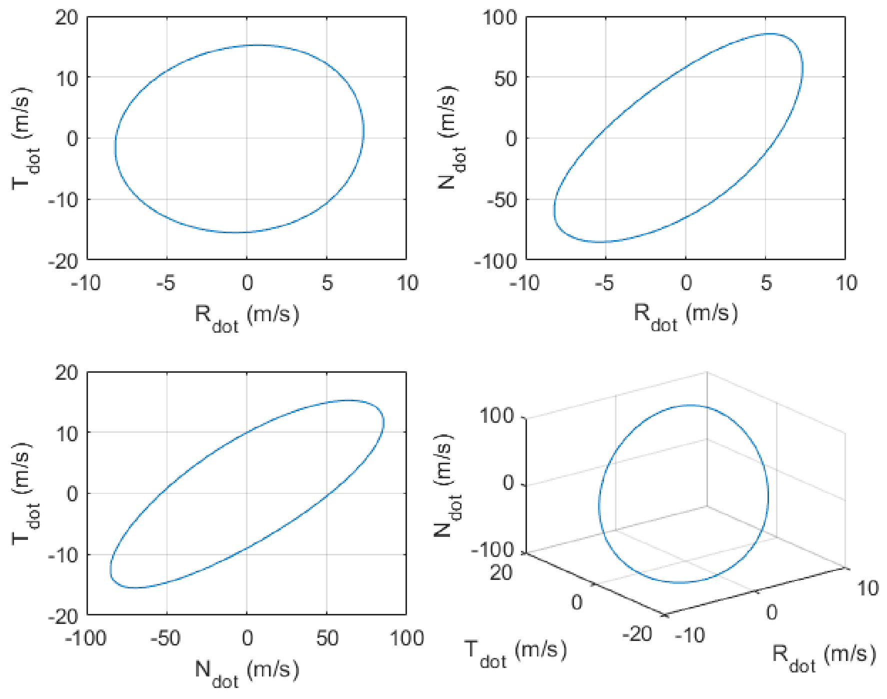


Figure 2.2: Deputy Velocity in Chief's Rotating RTN Frame over 10 Orbital Periods

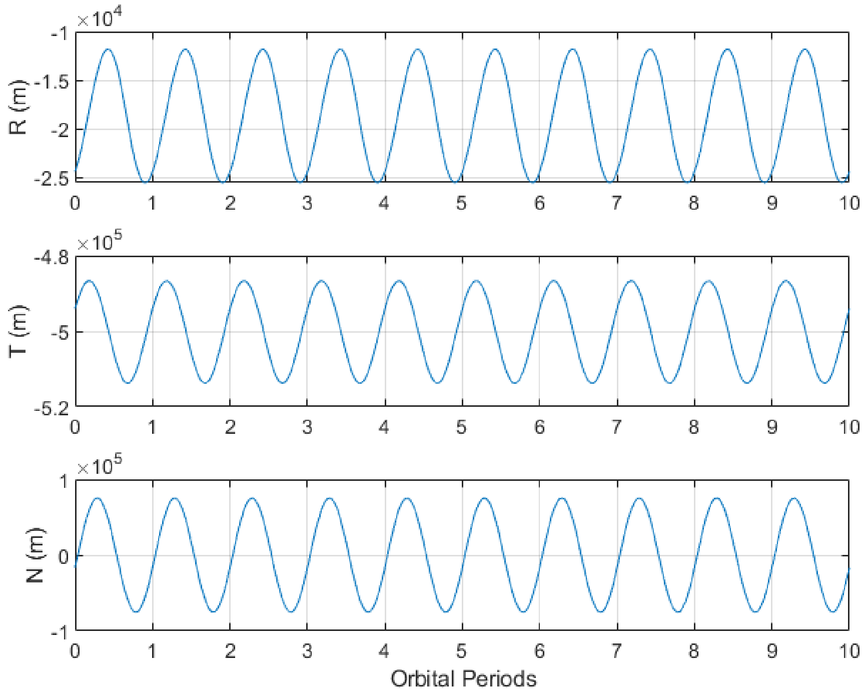


Figure 2.3: Deputy Position in Chief's Rotating RTN Frame over 10 Orbital Periods

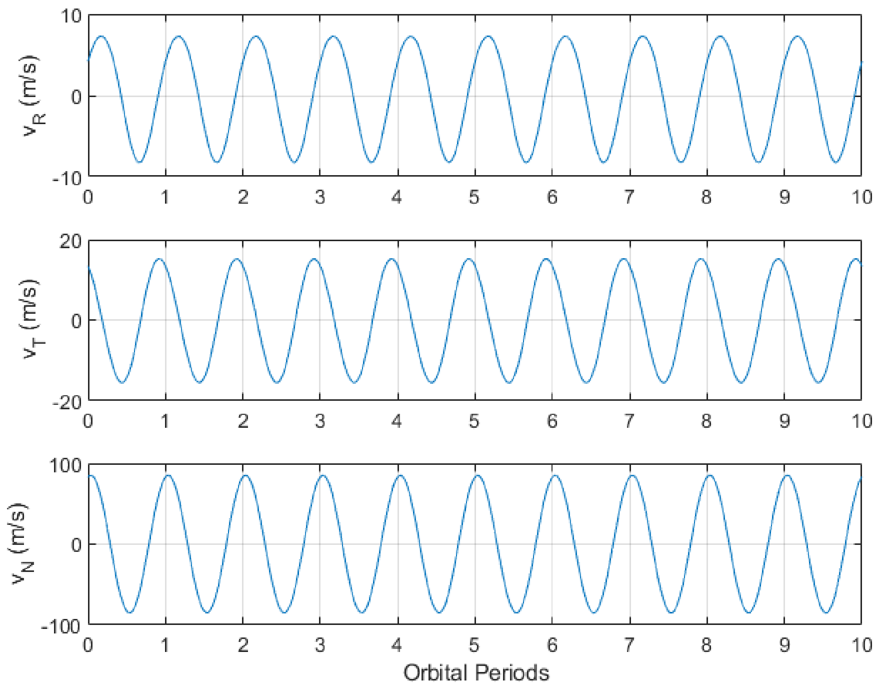


Figure 2.4: Deputy Velocity in Chief's Rotating RTN Frame over 10 Orbital Periods

## 2.3 c) Relative Orbit from Absolute Motion (unperturbed FODE)

The chief and deputy orbits are independently propagated using the fundamental two-body equations of motion:

$$\dot{\mathbf{r}} = \mathbf{v}, \qquad \dot{\mathbf{v}} = -\frac{\mu}{r^3}\mathbf{r}$$

The resulting inertial trajectories are post-processed to compute the relative position and velocity in the RTN frame:

$$\boldsymbol{\rho}_{\text{RTN}} = R_{\text{ECI}\to\text{RTN}}(\mathbf{r}_{\text{chief}}) \cdot (\mathbf{r}_{\text{dep}} - \mathbf{r}_{\text{chief}})$$
$$\dot{\boldsymbol{\rho}}_{\text{RTN}} = R_{\text{ECI}\to\text{RTN}} \cdot (\mathbf{v}_{\text{dep}} - \mathbf{v}_{\text{chief}}) - \boldsymbol{\omega} \times \boldsymbol{\rho}_{\text{RTN}}$$

where $\boldsymbol{\omega} = \frac{h}{r^2}\hat{\mathbf{N}}$ is the RTN frame angular velocity.

Figures below display these RTN-relative trajectories, and show agreement with the nonlinear propagation.

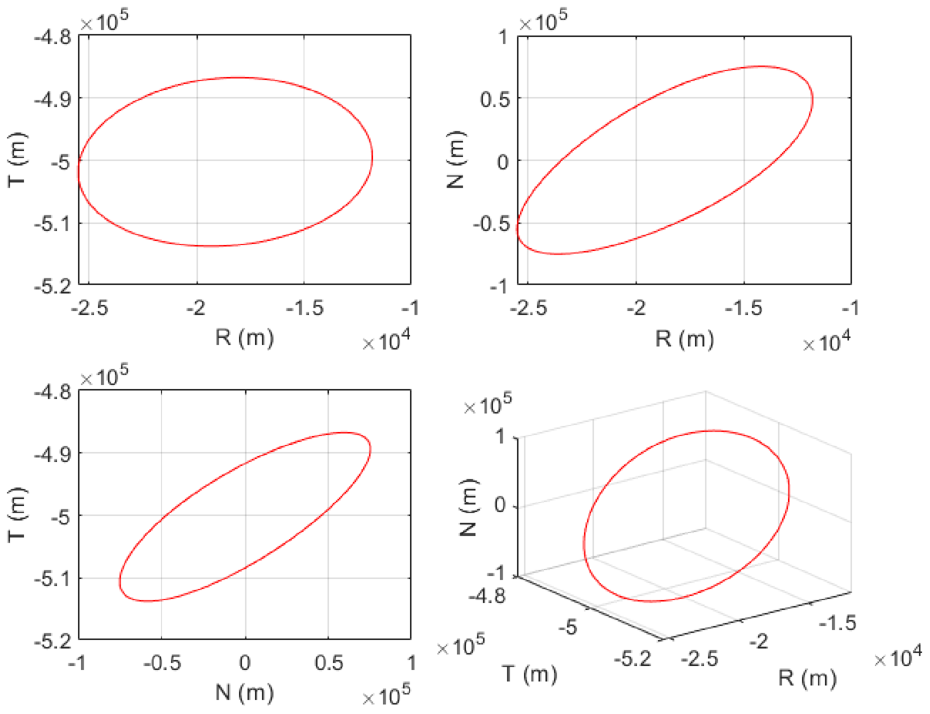


Figure 2.5: Deputy Position in Chief's Rotating RTN Frame over 10 Orbital Periods from FODE

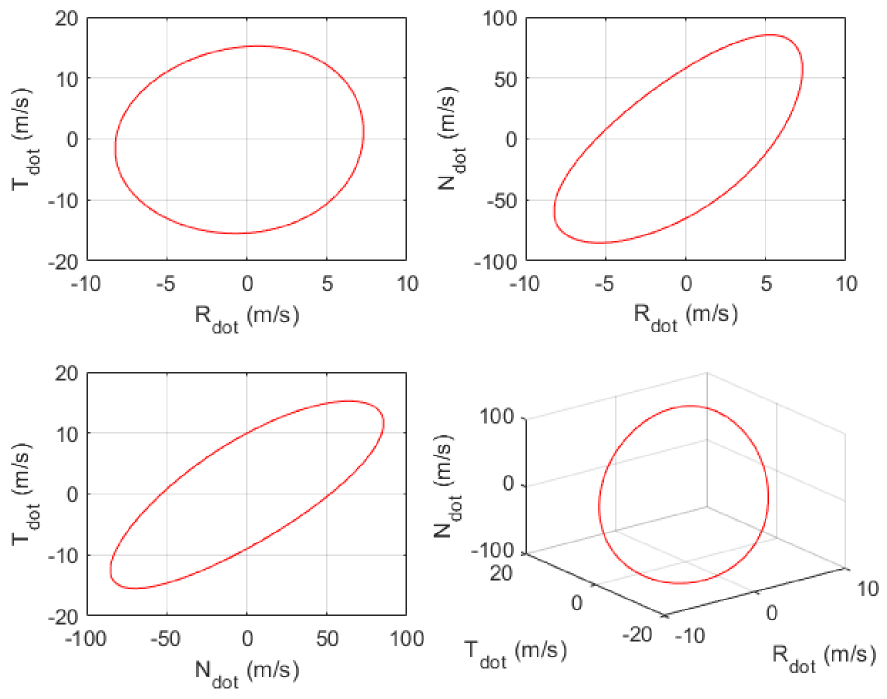


Figure 2.6: Deputy Velocity in Chief's Rotating RTN Frame over 10 Orbital Periods from FODE

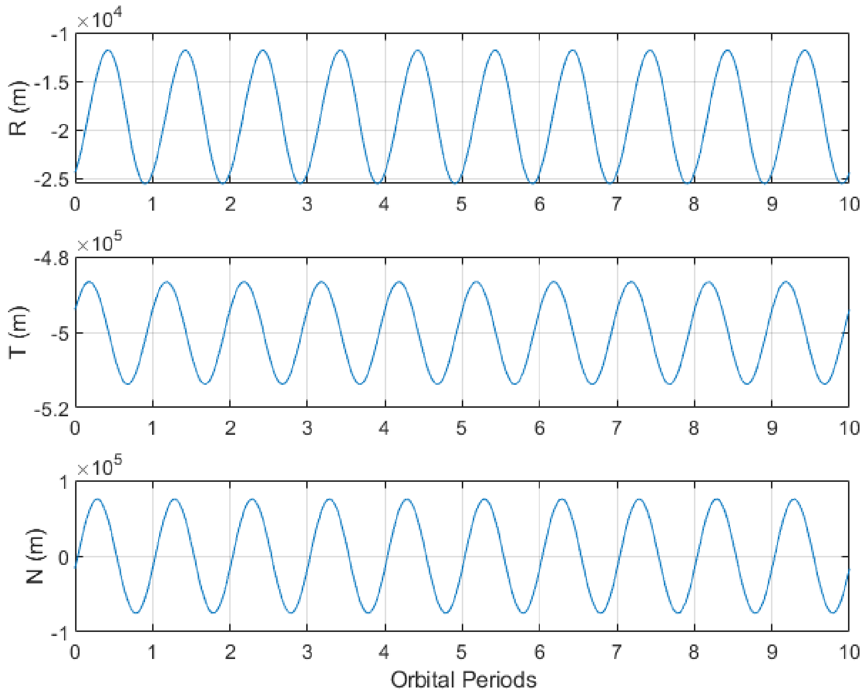


Figure 2.7: Deputy Position in Chief's Rotating RTN Frame over 10 Orbital Periods from FODE

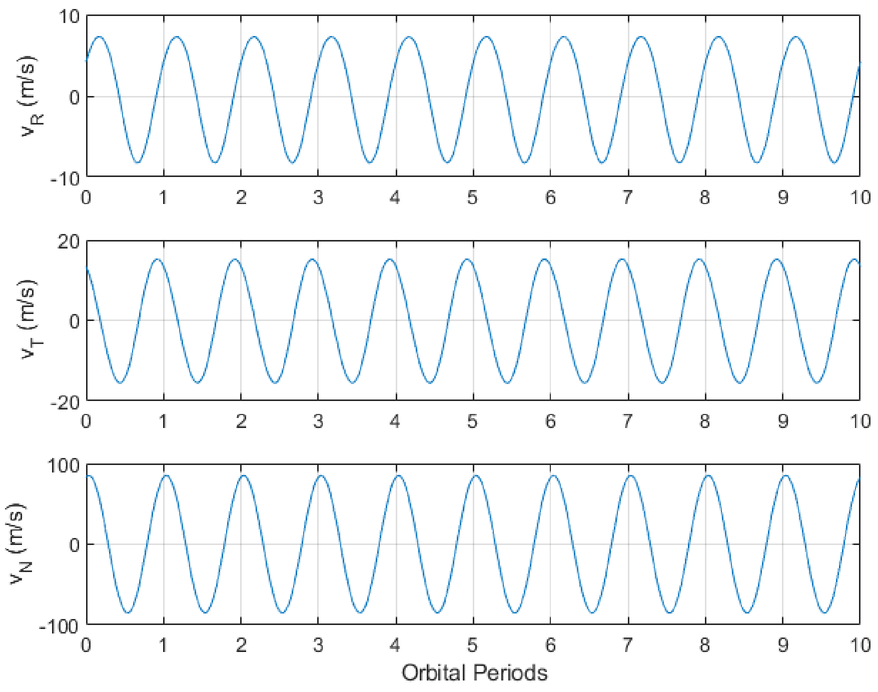


Figure 2.8: Deputy Velocity in Chief's Rotating RTN Frame over 10 Orbital Periods from FODE

## 2.4 d) Comparison Between Nonlinear RTN Integration and FODE-Based Differencing

To validate consistency, we compute the difference between RTN trajectories derived from the nonlinear integration and the FODE-based inertial differencing. The results from (b) and (c) are shown on the plots below and are near identical with only small numerical errors.

With identical initial conditions, both methods show near-identical trajectories (Figures below).

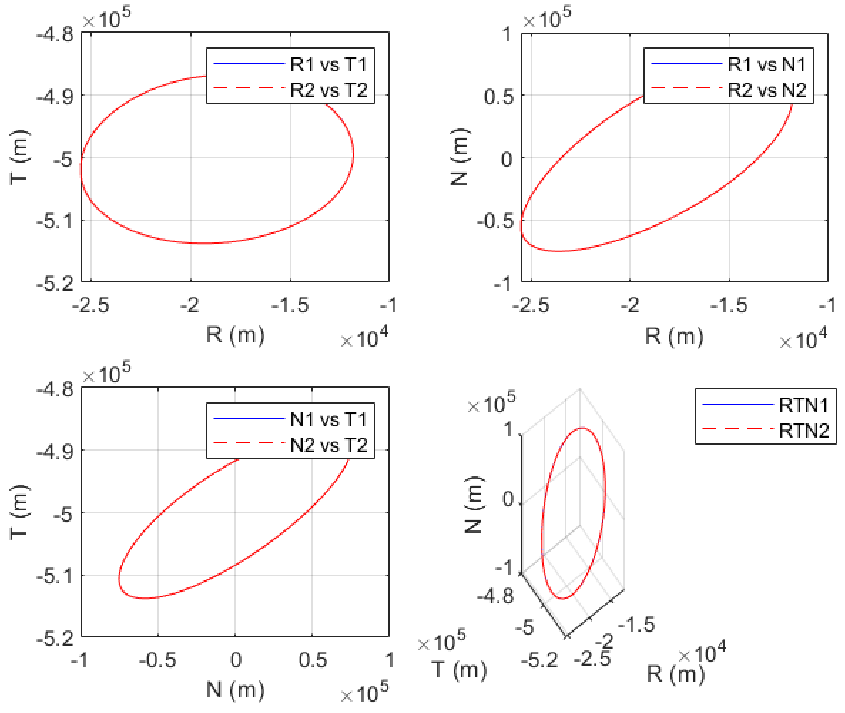


Figure 2.9: Deputy RTN Position over 10 Periods: RTN Nonlinear (1) vs. FODE (2)

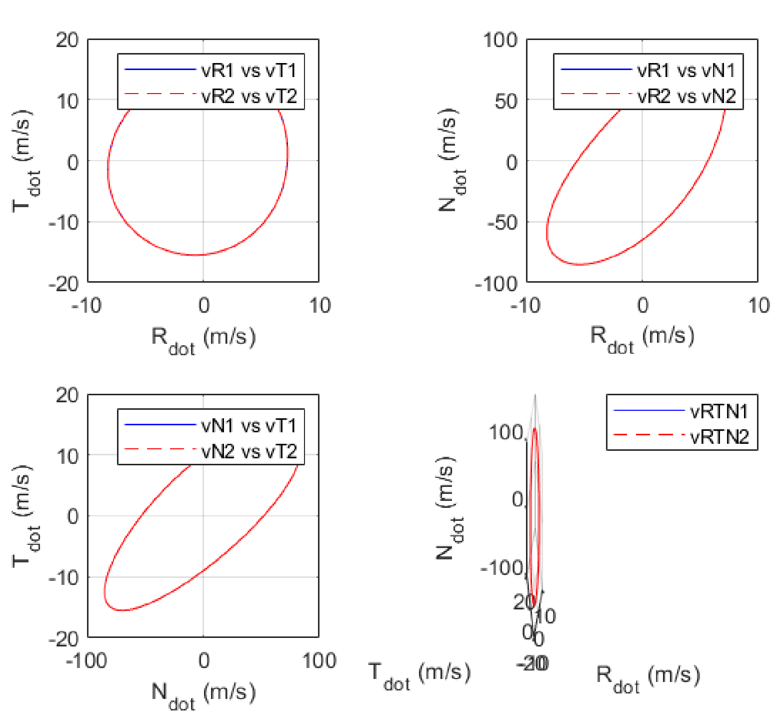


Figure 2.10: Deputy RTN Velocity over 10 Periods: RTN Nonlinear (1) vs. FODE (2)

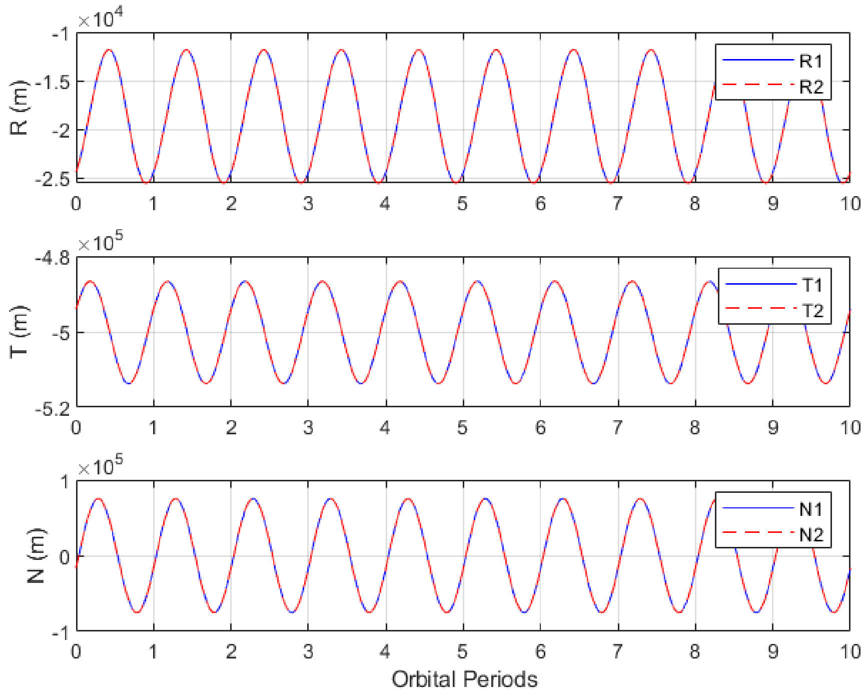


Figure 2.11: Deputy RTN Position over 10 Periods: RTN Nonlinear (1) vs. FODE (2)

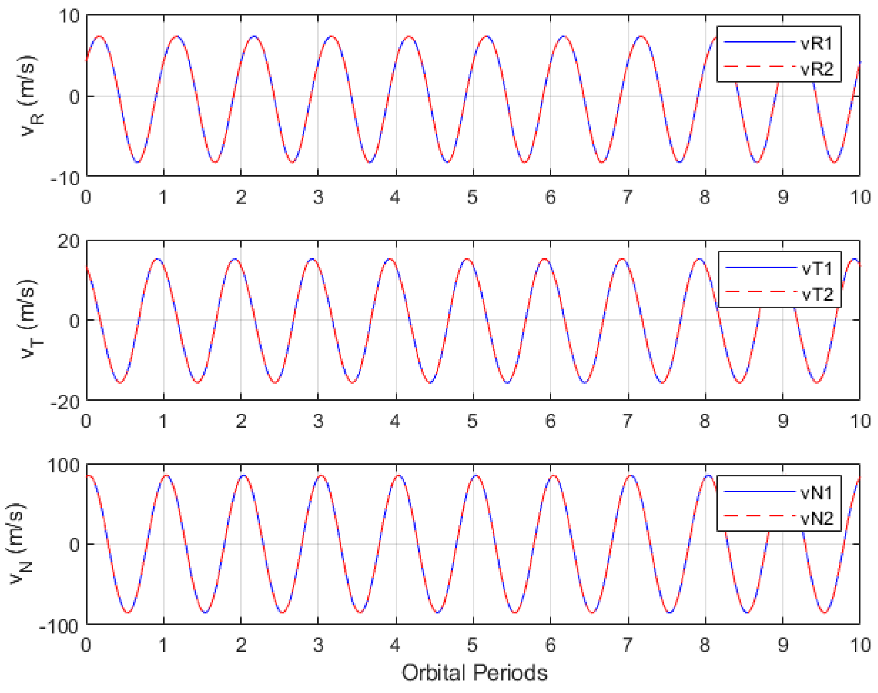


Figure 2.12: Deputy RTN Velocity over 10 Periods: RTN Nonlinear (1) vs. FODE (2)

To assess sensitivity to semi-major axis, we repeat with $\Delta a = 10$ km, inducing secular drift (Figures Below). Despite the drift, both integration methods remain consistent.

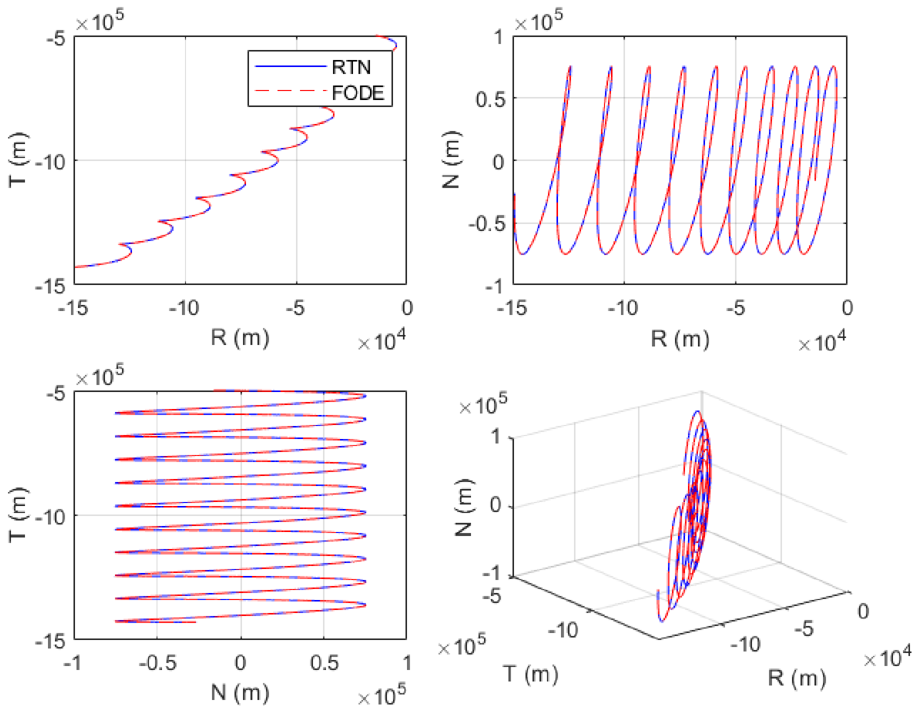


Figure 2.13: Deputy RTN Position over 10 Periods: RTN Nonlinear vs. FODE

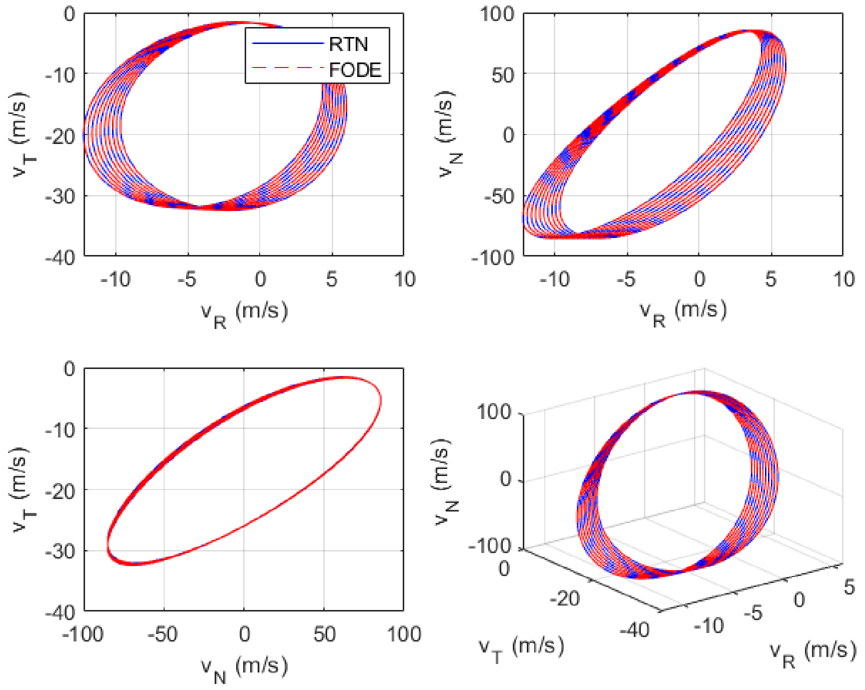


Figure 2.14: Deputy RTN Velocity over 10 Periods: RTN Nonlinear vs. FODE

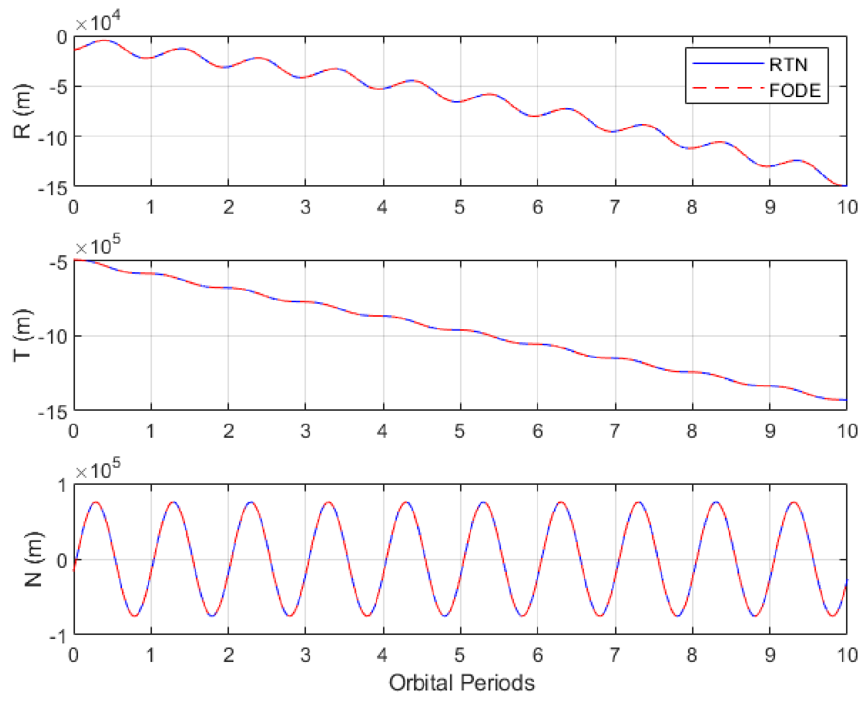


Figure 2.15: Deputy RTN Position over 10 Periods: RTN Nonlinear vs. FODE

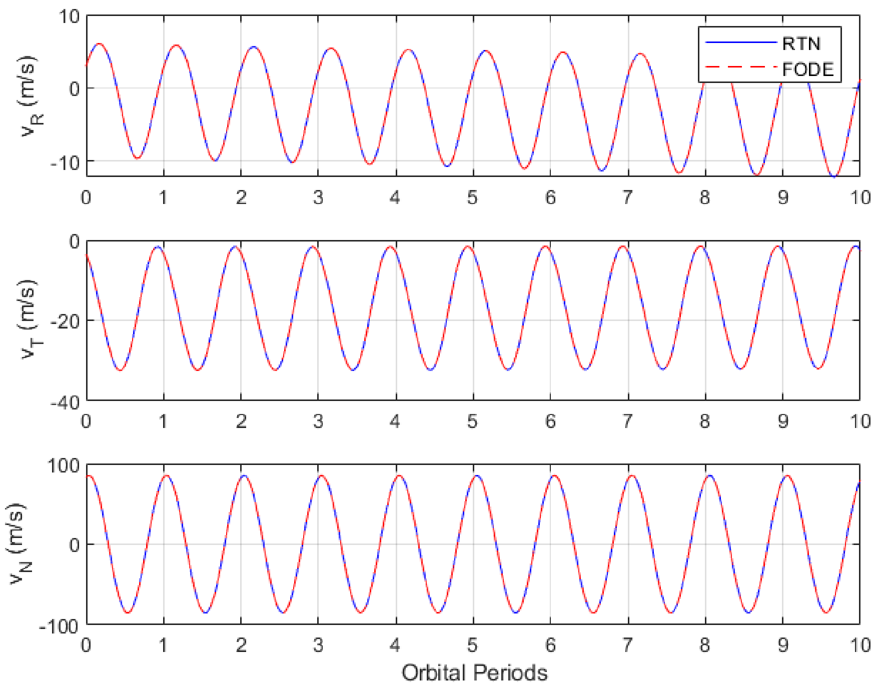


Figure 2.16: Deputy RTN Velocity over 10 Periods: RTN Nonlinear vs. FODE

Again see agreement

## 2.5 e) Optimal Impulsive Maneuver to Re-Establish Bounded Periodic Relative Motion

To remove secular drift and restore bounded motion, we perform a single tangential impulse to adjust the semi-major axis of the deputy. The Gauss variational equation for $a$ under a tangential impulse $\Delta v_T$ is:

$$\Delta a = \frac{2a^2 v}{\mu} \Delta v_T \tag{1}$$

Solving for $\Delta v_T$:

$$\Delta v_T = \frac{\mu}{2a^2 v} \Delta a \tag{2}$$

We compute this at the final time of the drift simulation (after 10 periods - the impulse will be applied at this point in (f)):

$$\begin{aligned} a &= 6.771 \times 10^6 \text{ m}, \\ v &= \|\mathbf{v}_{\text{chief, final}}\| = 7675.9 \text{ m/s}, \\ \Delta a &= -10^4 \text{ m} \end{aligned}$$

$$\Rightarrow \Delta v_T \approx -5.66 \text{ m/s}$$

## 2.6 f) Impulsive maneuver simulation

The computed $\Delta v_T$ is applied at $t = 10T$ by updating the deputy's velocity in the RTN frame. The integrator is then reinitialized with the modified state and propagated forward. The full position and velocity trajectories over 20 orbital periods are shown in the figures below. The deputy's motion becomes periodic and bounded, indicating successful drift correction. A slight residual drift remains, reflecting that the applied tangential velocity was not perfectly accurate. This is expected due to the GVE equation being an approximation, but overall the maneuver effectively reestablishes periodic motion and reduces long-term drift.

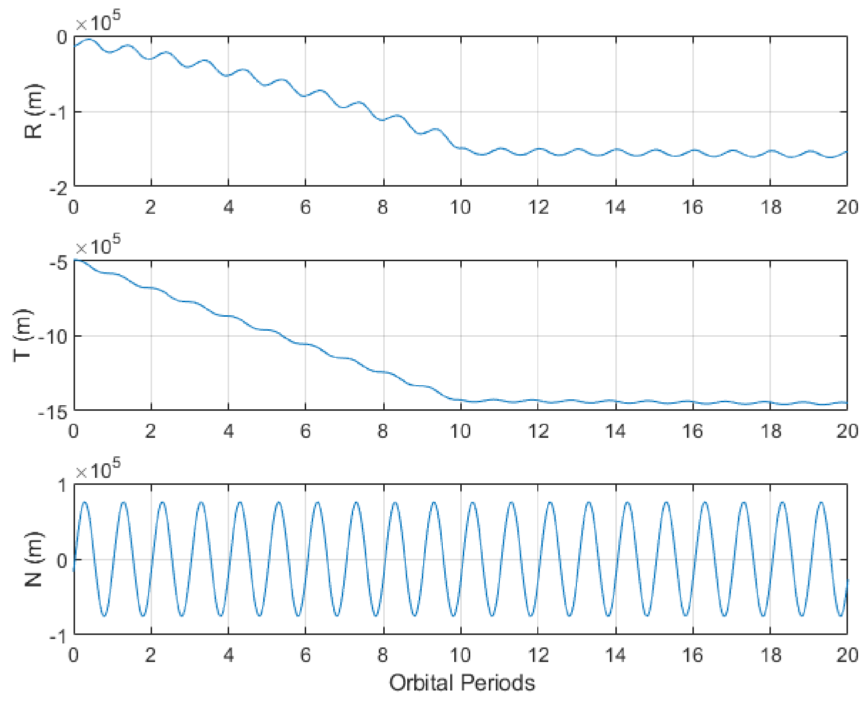


Figure 2.17: Deputy RTN position over 20 orbital periods with tangential impulse at T = 10

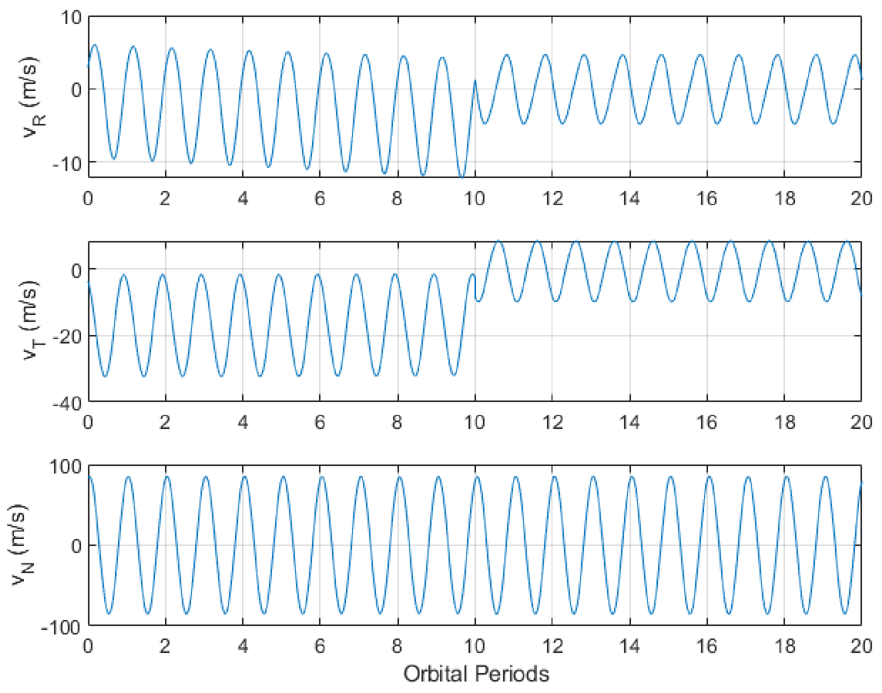


Figure 2.18: Deputy RTN velocity over 20 orbital periods with tangential impulse at T = 10

# 3 Problem Set 3

The code used for this assignment is available in our GitHub repository [1] and the Appendix. See "Problem Set 3 Code" for code for each section. Additionally, see "Propagators" and "Utils" sections.

Our work was inspired by [2] in addition to other cited sources.

## 3.1 We are Close in Near-Circular Orbits

### 3.1.1 a) Orbital Elements for HCW Validity

For the assumptions of the HCW equations to hold, the following conditions must be met:

- Small relative separation: $\rho \ll r_0$ (e.g., $\rho/r_0 \approx 0.001$)
- Near-circular orbits: $e \ll 1$ (e.g., $e \approx 0.001$)

The initial conditions chosen for the chief and deputy spacecraft are summarized in Table 3.11. The separations are small and eccentricities near zero, satisfying the assumptions of the HCW linearization.

| Spacecraft | $a$ [km] | $e$ | $i$ [°] | $\Omega$ [°] | $\omega$ [°] | $f$ [°] |
|---|---|---|---|---|---|---|
| Chief | 6771 | 0.0005 | 51.64 | 257.00 | 0.00 | 30.00 |
| Deputy | 6771 | 0.0006 | 51.69 | 257.05 | 0.05 | 29.95 |

Table 3.1: Initial absolute Keplerian orbital elements for the chief and deputy spacecraft.

The resulting relative separation magnitude is:

$$\rho = 3.8584 \text{ km} \approx 0.0006 \times r_0$$

which validates the use of HCW equations.

### 3.1.2 b) Initial Conditions and Relative Motion Analysis

#### i) Inertial States and Orbital Elements at $t_0 = 0$

The absolute initial conditions (position and velocity vectors) in the Earth-Centered Inertial (ECI) frame, as well as the corresponding Keplerian orbital elements, are given below for both the chief and the deputy.

| Spacecraft | $x$ [km] | $y$ [km] | $z$ [km] |
|---|---|---|---|
| Chief Position $\mathbf{r}_0$ | 727.797 | −6183.520 | 2653.512 |
| Deputy Position $\mathbf{r}_1$ | 730.872 | −6181.826 | 2655.113 |

Table 3.2: Chief and deputy inertial positions at $t_0 = 0$ (ECI frame).

| Spacecraft | $v_x$ [km/s] | $v_y$ [km/s] | $v_z$ [km/s] |
|---|---|---|---|
| Chief Velocity $\mathbf{v}_0$ | 4.883 | 2.810 | 5.213 |
| Deputy Velocity $\mathbf{v}_1$ | 4.877 | 2.815 | 5.217 |

Table 3.3: Chief and deputy inertial velocities at $t_0 = 0$ (ECI frame).

**ii) Relative Position, Velocity in chief's RTN Frame, and QNS-ROE**

The initial relative state of the deputy with respect to the chief, expressed in the chief's Radial-Tangential-Normal (RTN) frame, is:

$$\Delta\mathbf{r}_{\text{RTN}} = \begin{bmatrix} -0.5894 \\ 3.6637 \\ -1.0569 \end{bmatrix} \text{ km}, \quad \Delta\mathbf{v}_{\text{RTN}} = \begin{bmatrix} 0.00038 \\ 0.00133 \\ 0.00843 \end{bmatrix} \text{ km/s}$$

This result accounts for the correction due to the angular velocity of the chief's RTN frame:

$$\boldsymbol{\omega}_{\text{RTN}} = \frac{h}{r_0^2}\hat{\mathbf{N}}, \quad \text{where } h = \|\mathbf{r}_0 \times \mathbf{v}_0\|$$

and includes the cross-product term $\boldsymbol{\omega}_{\text{RTN}} \times \Delta\mathbf{r}_{\text{RTN}}$ in the relative velocity calculation.

The Quasi-Non-Singular Relative Orbital Elements (QNS-ROE) provide a well-conditioned set of relative orbital parameters even in near-circular orbital regimes. The QNS-ROE parameters between the deputy and the chief are defined as follows:

$$\delta a = \frac{a_1 - a_0}{a_0}$$

$$\delta\lambda = (M_1 + \omega_1) - (M_0 + \omega_0) + (\Omega_1 - \Omega_0)\cos i_0$$

$$\delta e_x = e_1 \cos\omega_1 - e_0 \cos\omega_0, \quad \delta e_y = e_1 \sin\omega_1 - e_0 \sin\omega_0$$

$$\delta i_x = i_1 - i_0, \quad \delta i_y = (\Omega_1 - \Omega_0)\sin i_0$$

The resulting QNS-ROE values at $t_0 = 0$ for the selected initial conditions are summarized in Table 3.4.

| Parameter | Value |
|---|---|
| $\delta a$ | $+0.000 \times 10^{0}$ |
| $\delta\lambda$ | $+5.416 \times 10^{-4}$ rad |
| $\delta e_x$ | $+1.000 \times 10^{-4}$ |
| $\delta e_y$ | $+5.236 \times 10^{-7}$ |
| $\delta i_x$ | $+8.727 \times 10^{-4}$ rad |
| $\delta i_y$ | $+6.843 \times 10^{-4}$ rad |

Table 3.4: Initial quasi-non-singular relative orbital elements (QNS-ROE) at $t_0 = 0$.

### 3.1.3 c) Computation of HCW Integration Constants

The Hill - Clohessy - Wiltshire (HCW) equations describe the linearized relative motion of a deputy spacecraft with respect to a chief on a circular orbit. The equations for deputy relative position and velocity in the chief's RTN frame are given by:

$$x(t) = \left(4x_0 + \frac{2\dot{y}_0}{n}\right) + \frac{\dot{x}_0}{n}\sin(nt) - \left(3x_0 + \frac{2\dot{y}_0}{n}\right)\cos(nt), \tag{1}$$

$$y(t) = -\left(6nx_0 + 3\dot{y}_0\right)t + \left(y_0 - \frac{2\dot{x}_0}{n}\right) + \left(6x_0 + \frac{4\dot{y}_0}{n}\right)\sin(nt) + \frac{2\dot{x}_0}{n}\cos(nt), \tag{2}$$

$$z(t) = \frac{\dot{z}_0}{n}\sin(nt) + z_0\cos(nt), \tag{3}$$

$$\dot{x}(t) = \dot{x}_0\cos(nt) + (3nx_0 + 2\dot{y}_0)\sin(nt), \tag{4}$$

$$\dot{y}(t) = -\left(6nx_0 + 3\dot{y}_0\right) + (6nx_0 + 4\dot{y}_0)\cos(nt) - 2\dot{x}_0\sin(nt), \tag{5}$$

$$\dot{z}(t) = \dot{z}_0\cos(nt) - nz_0\sin(nt). \tag{6}$$

Here, $x_0$, $y_0$, $z_0$, $\dot{x}_0$, $\dot{y}_0$, and $\dot{z}_0$ denote the initial relative position and velocity components of the deputy with respect to the chief, expressed in the chief-centered RTN frame at time $t_0 = 0$.

The six integration constants $c_1$ to $c_6$ are computed directly from the initial conditions $(x_0, y_0, z_0, \dot{x}_0, \dot{y}_0, \dot{z}_0)$ and the chief's mean motion $n$:

$$c_1 = -\left(3x_0 + \frac{2\dot{y}_0}{n}\right), \tag{3}$$

$$c_2 = \frac{\dot{x}_0}{n}, \tag{4}$$

$$c_3 = 4x_0 + \frac{2\dot{y}_0}{n}, \tag{5}$$

$$c_4 = y_0 - \frac{2\dot{x}_0}{n}, \tag{6}$$

$$c_5 = z_0, \tag{7}$$

$$c_6 = \frac{\dot{z}_0}{n}. \tag{8}$$

These constants fully parameterize the solution of the HCW equations for the given initial state.

Rewriting the equations with the constants:

$$x(t) = c_3 + c_1\cos(nt) + c_2\sin(nt), \tag{7}$$

$$y(t) = c_4 - 2c_1\sin(nt) + 2c_2\cos(nt) - \frac{3}{2}nc_3t, \tag{8}$$

$$z(t) = c_5\cos(nt) + c_6\sin(nt), \tag{9}$$

$$\dot{x}(t) = -c_1n\sin(nt) + c_2n\cos(nt), \tag{10}$$

$$\dot{y}(t) = -2c_1n\cos(nt) - 2c_2n\sin(nt) - \frac{3}{2}nc_3, \tag{11}$$

$$\dot{z}(t) = -c_5n\sin(nt) + c_6n\cos(nt). \tag{12}$$

The computed values of the integration constants for the current problem setup are presented in Table 3.5.

| Constant | Value |
|---|---|
| $c_1$ | $-5.792 \times 10^2$ |
| $c_2$ | $+3.358 \times 10^2$ |
| $c_3$ | $-1.023 \times 10^1$ |
| $c_4$ | $+2.992 \times 10^3$ |
| $c_5$ | $-1.057 \times 10^3$ |
| $c_6$ | $+7.434 \times 10^3$ |

Table 3.5: Integration constants $c_1$ to $c_6$ for the HCW equations, computed at $t_0 = 0$.

### 3.1.4 d) Propagation with HCW Equations

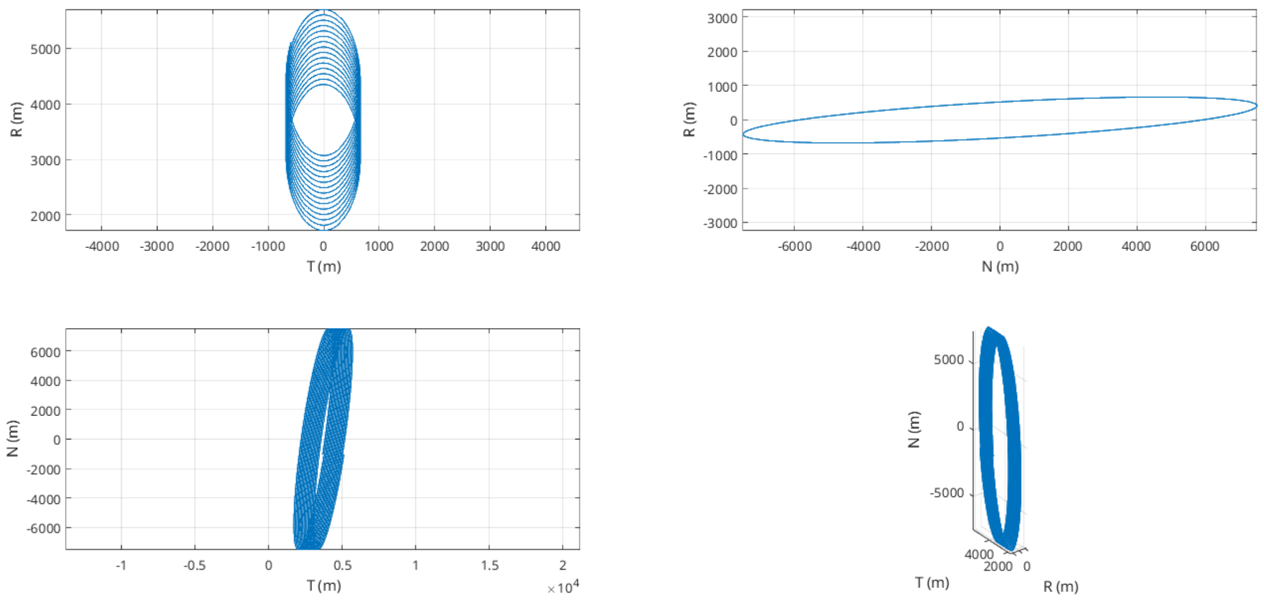


Figure 3.1: Deputy position in chief's rotating RTN frame propagated over 15 orbits using HCW solutions

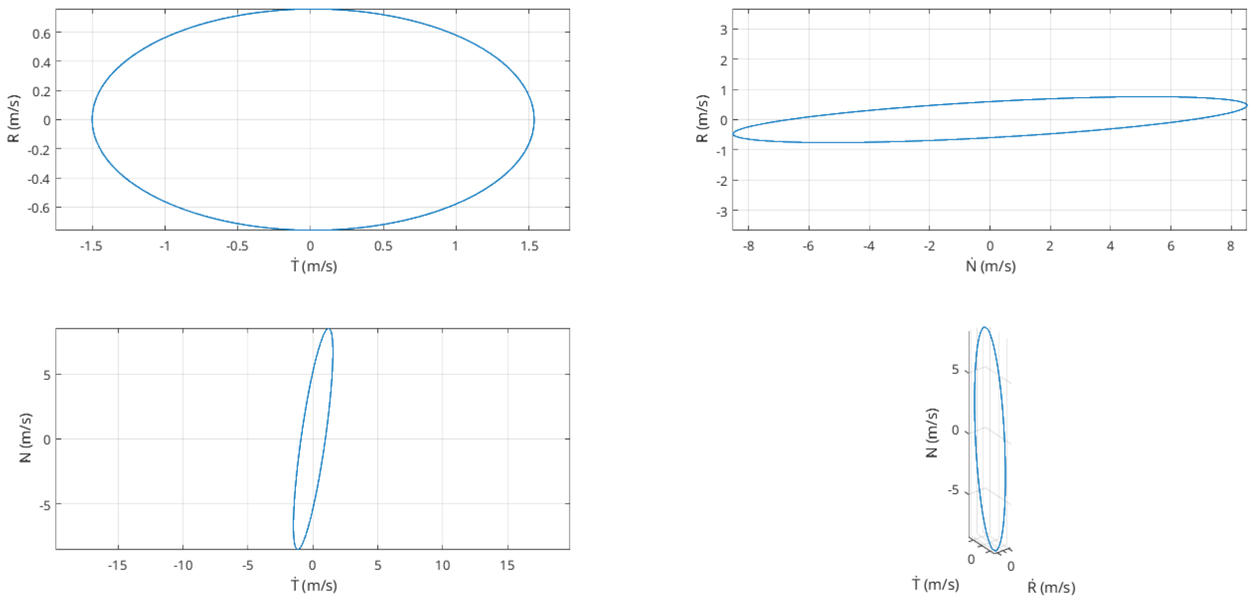


Figure 3.2: Deputy velocity in chief's rotating RTN frame propagated over 15 orbits using HCW solutions

### 3.1.5 e) Discussions on Observed Behavior and Expectations

We observe that the motion of the deputy spacecraft in the chief' s RTN frame is bounded in the radial (R) and cross-track (N) components. The motion remains periodic in these directions, showing no secular drift.

However, the along-track (T) component exhibits a clear linear drift over time, with an augmentation of the along-track distance. Although locally sinusoidal, the T-component shows a global drift, leading to an unbounded increase in the along-track separation.

According to our initial orbital element configuration, we explicitly set $\delta a = 0$, i.e the deputy's semi-major axis matches the chief's (energy matching condition). In a fully nonlinear treatment of the relative motion, this energy matching is expected to lead to bounded relative motion.

The HCW solution for the along-track motion contains a drift term:

$$y_{drift}(t) = -(6nx_0 + 3\dot{y}_0)t = -\frac{3}{2}nc_3t \tag{9}$$

In our case, the calculated value of $c_3 = 4x_0 + \frac{2\dot{y}_0}{n} \neq 0$, the along track drift should be expected.

The HCW model is derived from a linearization of the full nonlinear equations of relative motion about a circular chief orbit, keeping only first-order terms. This linearization truncates the nonlinear coupling between the semi-major axis difference.

While the energy matching condition guarantees boundedness in the fully nonlinear system, this is a global constraint that does not translate into the linearized HCW framework. In the HCW model, boundedness is only a local property, valid within the regime of the linear approximation.

The HCW model requires a separate condition to eliminate this drift based on initial condition. Specifically, we want:

$$\dot{y}_0 = -2nx_0. \tag{10}$$

This relationship directly cancels the secular drift term by ensuring that $c_3 = 0$.

## 3.2 We are Close in Eccentric Orbits

### 3.2.1 a) Initial Conditions

| | $a$ | $e$ | $i$ | $\Omega$ | $\omega$ | $\nu$ |
|---|---|---|---|---|---|---|
| Chief | 6771km | 0.1005 | 51.64° | 257° | 0° | 30° |
| Deputy | 6771km | 0.1006 | 51.69° | 257.05° | 0.05° | 29.95° |

Table 3.6: Initial absolute keplerian orbit elements of chief and deputy at $t_0 = 0$

| Spacecraft | $x$ [km] | $y$ [km] | $z$ [km] |
|---|---|---|---|
| Chief Position $\mathbf{r}_0$ | 663.0 | $-5633.4$ | 2417.4 |
| Deputy Position $\mathbf{r}_1$ | 665.8 | $-5631.6$ | 2418.8 |

Table 3.7: Chief and deputy inertial positions at $t_0 = 0$ (ECI frame).

| Spacecraft | $v_x$ [km/s] | $v_y$ [km/s] | $v_z$ [km/s] |
|---|---|---|---|
| Chief Velocity $\mathbf{v}_0$ | 5.3745 | 2.7165 | 5.8445 |
| Deputy Velocity $\mathbf{v}_1$ | 5.3678 | 2.7229 | 5.8492 |

Table 3.8: Chief and deputy inertial velocities at $t_0 = 0$ (ECI frame).

| | R | T | N |
|---|---|---|---|
| Relative Position (km) | -0.8662 | 3.3376 | -0.9629 |
| Relative Velocity (km/s) | -0.00019 | 0.00247 | 0.00914 |

Table 3.9: Deputy relative position and velocity in chief's RTN Frame at t $t_0 = 0$

Justification: We just increased the eccentricity by 0.1, keeping all relative differences the same. Maybe if there is a space station in an eccentric orbit some time in the future this could be helpful.

The resulting relative separation magnitude is:

$$\rho = 3.58003 \text{ km} \approx 0.0006 \times r_0$$

which validates the use of Tschauner-Hempel equations.

Separation = Magnitude of Relative Position = 3.58 km $<$ 6.771 km $= 0.001 r_0$

### 3.2.2 b) Integration Constants of the YA Solution

We normalize the deputy's state in the chief's RTN frame. The relative position components are normalized by the chief's semi-major axis $a_0$, and the relative velocity components are normalized by the product of $a_0$ and the chief's angular rate at the true anomaly $f_0$, denoted as $\dot{f}_0$:

$$\bar{x} = \frac{x}{a_0}, \quad \bar{y} = \frac{y}{a_0}, \quad \bar{z} = \frac{z}{a_0}, \quad \dot{\bar{x}} = \frac{\dot{x}}{a_0 \dot{f}_0}, \quad \dot{\bar{y}} = \frac{\dot{y}}{a_0 \dot{f}_0}, \quad \dot{\bar{z}} = \frac{\dot{z}}{a_0 \dot{f}_0}. \tag{11}$$

In these expressions:

- $x, y, z$ are the relative position components between the deputy and the chief, expressed in the chief's RTN frame.
- $\dot{x}, \dot{y}, \dot{z}$ are the corresponding relative velocity components in the RTN frame, accounting for the rotation of the frame.
- $a_0$ is the chief's semi-major axis.
- $\dot{f}_0$ is the rate of change of true anomaly, given by:

$$\dot{f}_0 = \frac{h}{r_0^2}, \tag{12}$$

  where $h = \|\mathbf{r}_0 \times \mathbf{v}_0\|$ is the specific angular momentum of the chief, and $r_0 = \|\mathbf{r}_0\|$ is the chief's radial distance at true anomaly $f_0$.

The Yamanaka Ankersen (YA) solution provides an analytical closed-form expression for the linearized relative motion of a deputy spacecraft with respect to a chief in eccentric orbits. The YA integration constants are expressed:

$$\begin{bmatrix} K_1 \\ K_2 \\ K_3 \\ K_4 \\ K_5 \\ K_6 \end{bmatrix} = \phi^{-1}\big(f(0)\big) \begin{bmatrix} \bar{x}\big(f(0)\big) \\ \bar{y}\big(f(0)\big) \\ \bar{z}\big(f(0)\big) \\ \dot{\bar{x}}\big(f(0)\big) \\ \dot{\bar{x}}\big(f(0)\big) \\ \dot{\bar{x}}\big(f(0)\big) \end{bmatrix}$$

The inverse matrix $\phi^{-1}(f(0))$ relates the normalized initial relative state vector to the integration constants $K_1, \ldots, K_6$. It is expressed

$$\phi^{-1}\big(f(0)\big) = \frac{1}{\eta^2} \begin{bmatrix} -3s\dfrac{k+e^2}{k^2} & c-2e & 0 & -s\dfrac{k+1}{k} & 0 & 0 \\ -3\Big(e+\frac{c}{k}\Big) & -s & 0 & -\Big(c\frac{k+1}{k}+e\Big) & 0 & 0 \\ 3k-\eta^2 & e\,s & 0 & k^2 & 0 & 0 \\ -3e\,s\dfrac{k+1}{k^2} & -2+e\,c & \eta^2 & -e\,s\dfrac{k+1}{k} & 0 & 0 \\ 0 & 0 & 0 & 0 & \eta^2\cos f & -\eta^2\sin f \\ 0 & 0 & 0 & 0 & \eta^2\sin f & \eta^2\cos f \end{bmatrix} \tag{5.127 [3]}$$

The quantities appearing in the matrix $\phi^{-1}$ are defined as:

- $e$: eccentricity of the chief's orbit.
- $f$: true anomaly at the initial time $t = 0$.
- $\eta = \sqrt{1-e^2}$: eccentricity function.
- $k = 1 + e\cos f$.
- $s = k\sin f$.
- $c = k\cos f$.

These variables arise from the Yamanaka Ankersen (YA) formulation for eccentric relative motion, which provides the analytic solution coefficients for the linearized relative dynamics in eccentric orbits.

And we obtain (see code):

$$\begin{bmatrix} K_1 \\ K_2 \\ K_3 \\ K_4 \\ K_5 \\ K_6 \end{bmatrix} = \begin{bmatrix} 0.0269 \\ -0.3172 \\ 0.1420 \\ -0.3799 \\ -0.1019 \\ 0.1768 \end{bmatrix}$$

### 3.2.3 c) Propagation of the Normalized Deputy State with the YA Equations

Starting from the initial constants computed in the previous section, we propagate the Yamanaka-Ankersen state transition matrix (STM) to obtain the normalized RTN state as a function of true anomaly. Specifically,

$$f_i = f_0 + i\,\Delta f, \quad \bar{\boldsymbol{x}}(f_i) = \Phi(f_i, f_0)\,\mathbf{C}, \quad i = 0, 1, \ldots, N,$$

where

$$\Delta f = \text{step\_size}, \quad f_0 = \alpha_0(6),$$

Define

$$\rho(f) \;=\; 1 + e\cos f, \qquad \rho_0 = \rho(f_0), \qquad k_2 \;=\; \frac{\sqrt{\mu}}{[\,a(1-e^2)\,]^{3/2}}, \qquad J = k_2\,(t - t_0).$$

Then the $6 \times 6$ STM is written in block form as

$$\Phi(f, f_0) = \underbrace{\begin{pmatrix} \frac{1}{\rho(f)}\,I_3 & 0_{3\times3} \\ k_2\,e\sin f\,I_3 & k_2\,\rho(f)\,I_3 \end{pmatrix}}_{N(f)} \underbrace{\begin{pmatrix} \Phi_{xy}(f) & 0_{4\times2} \\ 0_{2\times4} & \Phi_z(f) \end{pmatrix}}_{\Phi_{\text{core}}(f)} \underbrace{\begin{pmatrix} \rho_0\,I_3 & 0_{3\times3} \\ -\,e\sin f_0\,I_3 & \frac{1}{k_2\,\rho_0}\,I_3 \end{pmatrix}}_{N_0},$$

where

$$\Phi_{xy}(f) = \begin{pmatrix} s_f & 0 & 2 - 3\,e\,s_f\,J & -c_f \\ c_f\left(1 + \frac{1}{\rho(f)}\right) & 1 & -3\,\rho(f)^2 J & s_f\left(1 + \frac{1}{\rho(f)}\right) \\ s'_f & 0 & -3\,e\left(s'_f\,J + s_f/\rho(f)^2\right) & -c'_f \\ -2\,s_f & 0 & -3\left(1 - 2\,e\,s_f\,J\right) & 2\,c_f - e \end{pmatrix}, \quad \Phi_z(f) = \frac{1}{\rho(f - f_0)} \begin{pmatrix} c_{f-f_0} & s_{f-f_0} \\ -\,s_{f-f_0} & c_{f-f_0} \end{pmatrix}.$$

Here

$$s_f = \rho(f)\,\sin f, \quad c_f = \rho(f)\,\cos f, \quad s'_f = \frac{d}{df}\,(\rho(f)\sin f), \quad c'_f = \frac{d}{df}\,(\rho(f)\cos f),$$

and $I_3$ is the $3 \times 3$ identity.

The returned normalized state

$$\bar{\boldsymbol{x}} = [\,\bar{x},\ \dot{\bar{x}},\ \bar{y},\ \dot{\bar{y}},\ \bar{z},\ \dot{\bar{z}}\,]^T$$

is converted to physical RTN coordinates via

$$x = a\,\bar{x}, \quad v_R = a\,\dot{f}_0\,\dot{\bar{x}},$$

with

$$\dot{f}_0 = \frac{h}{r_0^2}, \quad h = \|\mathbf{r}_0 \times \mathbf{v}_0\|, \quad r_0 = \|\mathbf{r}_0\|.$$

The same scaling is applied to $y, \dot{y}, z, \dot{z}$.

Propagating the deputy's relative state over 15 orbital periods, we obtain the following graphs:

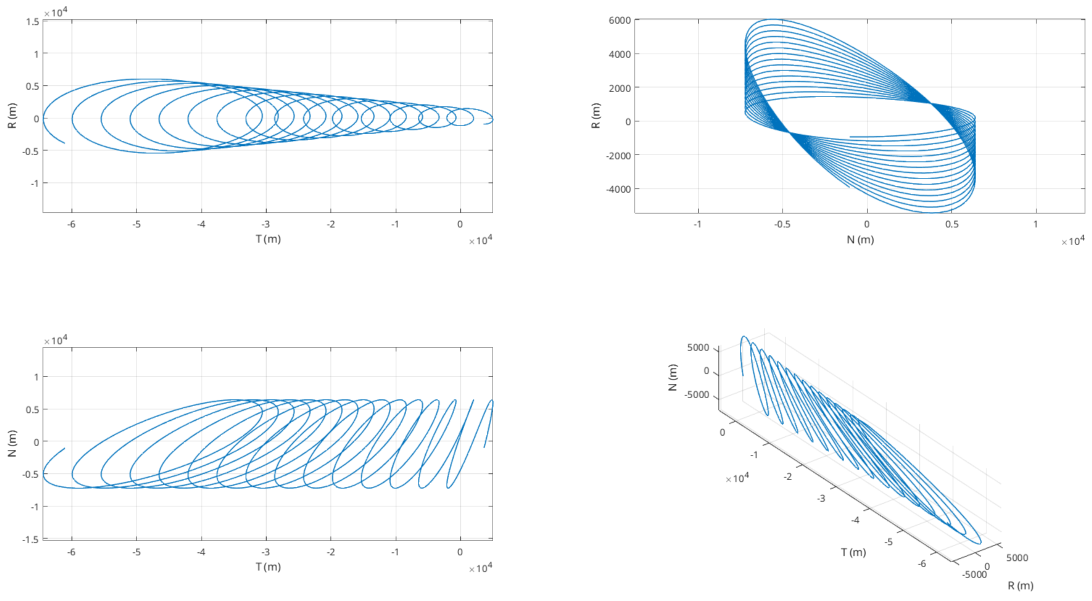


Figure 3.3: Deputy position in chief's rotating RTN frame propagated over 15 orbits using YA solutions

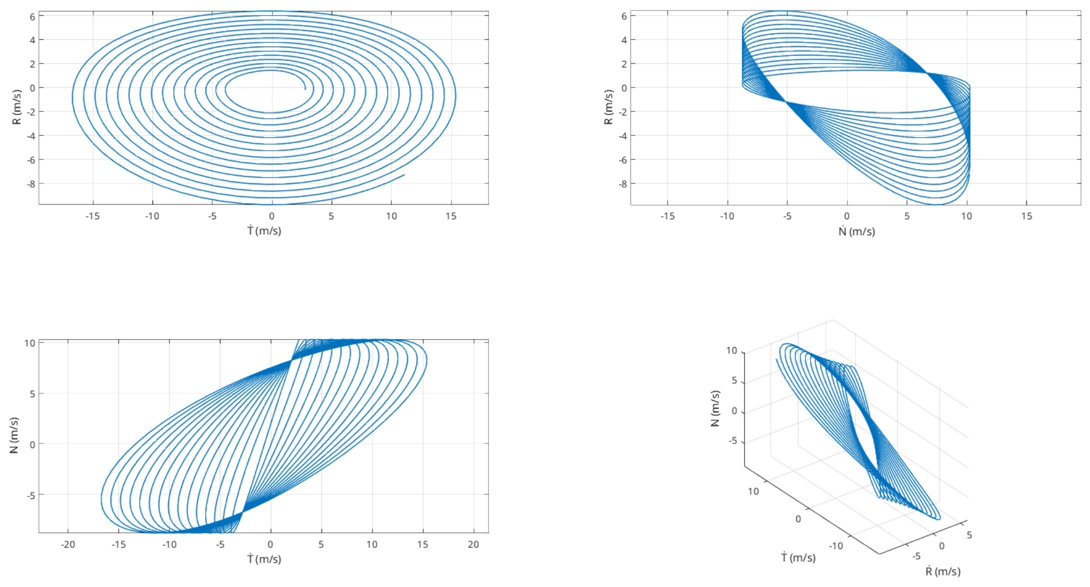


Figure 3.4: Deputy velocity in chief's rotating RTN frame propagated over 15 orbits using YA solutions

### 3.2.4 d) Discussions on Observed Behavior and Expectations

In the propagation of the deputy's relative motion using the YA equations, we observe a very clear drift behavior in the radial (R) and along-track (T) components. While the local evolution remains sinusoidal as expected from the harmonic structure of the YA solution, the amplitude of the radial component increases over time, and the along-track separation exhibits a linear growth. In contrast, the cross-track (N) component remains bounded and purely oscillatory.

This global growth in the in-plane motion leads to an overall unbounded relative trajectory, which contradicts the bounded behavior we would expect from the energy matching condition $\delta a = 0$. Energy matching implies that the semi-major axes of the chief and deputy are identical, which is a global condition ensuring boundedness in the fully nonlinear relative motion equations.

However, the YA solution is derived from a first-order linearization of the nonlinear dynamics about an eccentric reference orbit. This linearization inherently truncates the coupling between semi-major axis differences and phase evolution. As a result, even when $\delta a = 0$, the linearized YA model can exhibit secular drift, especially in the radial and tangential directions, because it does not fully capture the nonlinear coupling terms responsible for energy commensurability and phase locking.

To mitigate this unbounded behavior, one could either directly impose specific initial conditions on the normalized state or integration constants $K_1, \ldots, K_6$ that suppress the drift terms within the YA linear model, though this typically requires fine-tuning and may not fully generalize.

### 3.2.5 e) Computation of Quasi-Non-Singular Relative Orbital Elements

The The quasi-non-singular relative orbital elements (QNS-ROE) parameters between the deputy and the chief spacecraft are computed using the following expressions:

$$\delta a = \frac{a_1 - a_0}{a_0}, \tag{13}$$

$$\delta\lambda = (M_1 + \omega_1) - (M_0 + \omega_0) + (\Omega_1 - \Omega_0)\cos i_0, \tag{14}$$

$$\delta e_x = e_1 \cos\omega_1 - e_0 \cos\omega_0, \quad \delta e_y = e_1 \sin\omega_1 - e_0 \sin\omega_0, \tag{15}$$

$$\delta i_x = i_1 - i_0, \quad \delta i_y = (\Omega_1 - \Omega_0)\sin i_0. \tag{16}$$

Here:

- $a$ is the semi-major axis,
- $e$ is the eccentricity,
- $i$ is the inclination,
- $\Omega$ is the right ascension of the ascending node (RAAN),
- $\omega$ is the argument of perigee,
- $M$ is the mean anomaly.

The mean anomaly $M$ is calculated from the true anomaly $f$ via the eccentric anomaly $E$ using the following relations:

$$E = 2\arctan\left(\sqrt{\frac{1-e}{1+e}}\tan\frac{f}{2}\right), \tag{17}$$

$$M = E - e\sin E. \tag{18}$$

The resulting QNS-ROE values at $t_0 = 0$ for the selected initial conditions are:

$$\delta\alpha = 1.0e{-}03 \times \begin{bmatrix} 0.0000 \\ 0.5992 \\ 0.1000 \\ 0.0878 \\ 0.8727 \\ 0.6843 \end{bmatrix}.$$

### 3.2.6 f) Propagation of Relative Motion Using Geometric Linear Mapping

We now propagate the relative position and velocity of the deputy spacecraft using the geometric linear mapping based on the QNS-ROE. This method applies to arbitrary eccentricity and provides an analytical linearized solution by directly mapping the relative orbital elements into the RTN position and velocity states through a time-varying transformation matrix.

The geometric linear mapping solution relates the deputy's relative state $\mathbf{x}(t)$ to the QNS-ROE vector $\delta\alpha$ via:

$$\mathbf{x}(t) = S\,B(t)\,\delta\alpha, \tag{19}$$

where:

- $\mathbf{x}(t) = \begin{bmatrix} x & y & z & \dot{x} & \dot{y} & \dot{z} \end{bmatrix}^{\mathrm{T}}$ is the relative state vector in the chief's RTN frame,
- $\delta\alpha = \begin{bmatrix} \delta a & \delta\lambda & \delta e_x & \delta e_y & \delta i_x & \delta i_y \end{bmatrix}^{\mathrm{T}}$ is the quasi-non-singular relative orbital elements vector,
- $S$ is a scaling matrix defined as:

$$S = \begin{bmatrix} a\eta^2\,\mathbf{I}_{3\times 3} & \mathbf{0}_{3\times 3} \\ \mathbf{0}_{3\times 3} & \dfrac{an}{\eta}\,\mathbf{I}_{3\times 3} \end{bmatrix}, \tag{20}$$

  with $a$ the chief's semi-major axis, $e$ the chief's eccentricity, $\eta = \sqrt{1-e^2}$, and $n = \sqrt{\mu/a^3}$ the mean motion.

The matrix $B(t)$, which maps the QNS-ROE elements to the relative state vector in the RTN frame, contains time-varying coefficients that depend on the chief's orbital geometry, eccentricity, and inclination. The general form of $B(t)$ is:

$$B(t) = \begin{bmatrix} b_{x1} & b_{x2} & b_{x3} & b_{x4} & 0 & b_{x6} \\ b_{y1} & b_{y2} & b_{y3} & b_{y4} & 0 & b_{y6} \\ 0 & 0 & 0 & 0 & b_{z5} & b_{z6} \\ b_{xd1} & b_{xd2} & b_{xd3} & b_{xd4} & 0 & b_{xd6} \\ b_{yd1} & b_{yd2} & b_{yd3} & b_{yd4} & 0 & b_{yd6} \\ 0 & 0 & 0 & 0 & b_{zd5} & b_{zd6} \end{bmatrix},$$

where each entry $b_{ij}$ is a function of the true anomaly $f$, the argument of latitude $u = \omega + f$, and the scaling factor $k(f) = 1 + e\cos f$. The derivatives of $k(f)$ with respect to $f$ is $k'(f) = -e\sin f$.

The full expressions for each coefficient in $B(t)$ are implemented directly in the provided code and can be referenced for numerical evaluation at any true anomaly $f$. This mapping enables the propagation of relative motion linearly while respecting the elliptical nature of the chiefâs orbit.

We propagate the deputy's relative motion over 15 orbits, using the initial QNS-ROE derived previously. The resulting position and velocity trajectories are plotted alongside the YA propagation results for direct comparison.

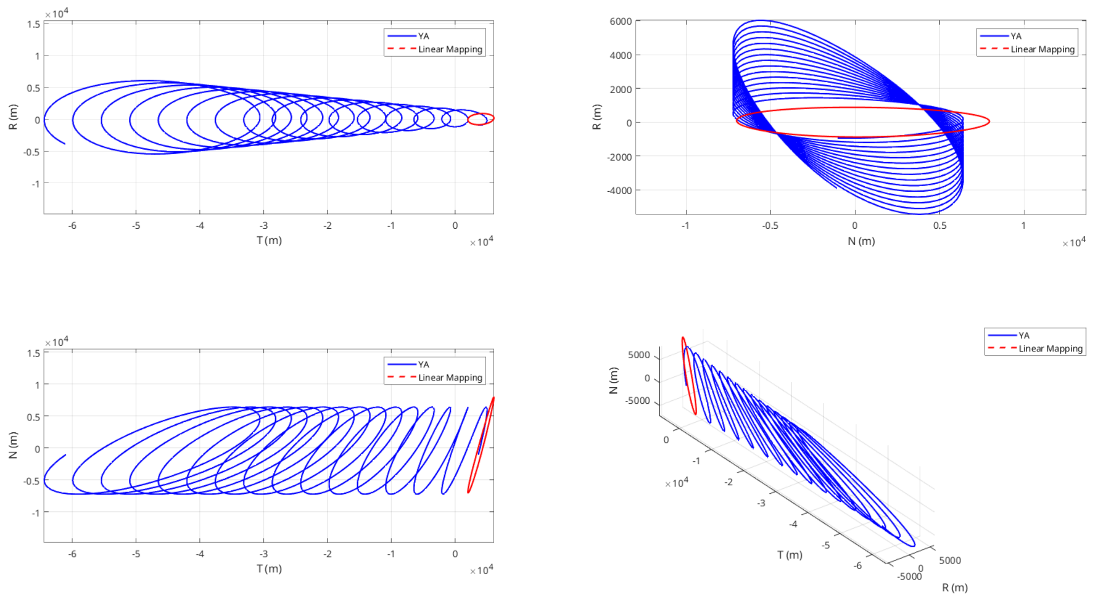


Figure 3.5: Deputy Position in Chief's Rotating RTN Frame over 15 Orbital Periods: Y.A. Solution (Blue) vs. Geometric Linear Mapping Solution (Red)

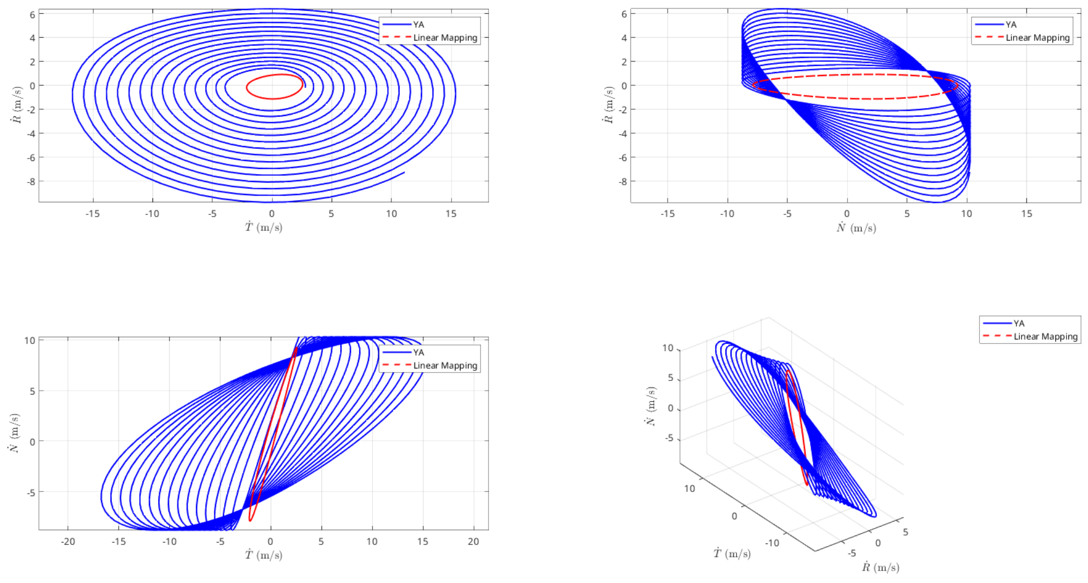


Figure 3.6: Deputy Velocity in Chief's Rotating RTN Frame over 15 Orbital Periods: Y.A. Solution (Blue) vs. Geometric Linear Mapping Solution (Red)

### 3.2.7 g) Discussion on Observed Behavior and Comparison to Previous Methods

The propagation of the relative motion using the geometric linear mapping approach based QNS-ROE yields a fully bounded solution. Specifically, we observe that the relative position and velocity components in the radial (R), along-track (T), and cross-track (N) directions remain sinusoidal and bounded over the entire simulation window.

This behavior directly contrasts with the results obtained using the YA and HCW solutions in the previous parts of the assignment, where we observed secular drift in the along-track and radial components despite enforcing the energy matching condition $\delta a = 0$.

Both the HCW and YA formulations are based on first-order linearizations of the nonlinear relative motion equations. In these linearizations, the coupling between the semi-major axis difference and the relative phase evolution is truncated, which prevents the full enforcement of the energy matching condition at the dynamics level. This leads to the presence of drift modes.

In contrast, the geometric linear mapping approach using QNS-ROE directly incorporates the energy matching condition within its formulation. The quasi-non-singular elements include the eccentricity vector and mean argument of latitude differences, which fully capture the energy and phase relationships between the chief and deputy orbits.

### 3.2.8 h) Comparison of Numerical Nonlinear, Y.A., and Geometric Linear Mapping

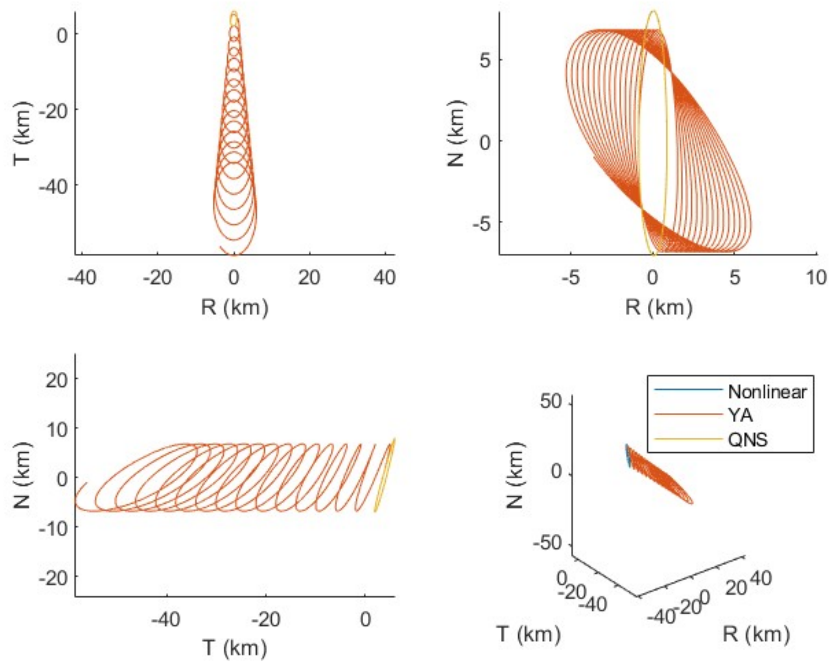


Figure 3.7: Deputy Position in Chief's Rotating RTN Frame over 15 Orbital Periods: Numerical Nonlinear vs Y.A. vs Geometric Linear Mapping

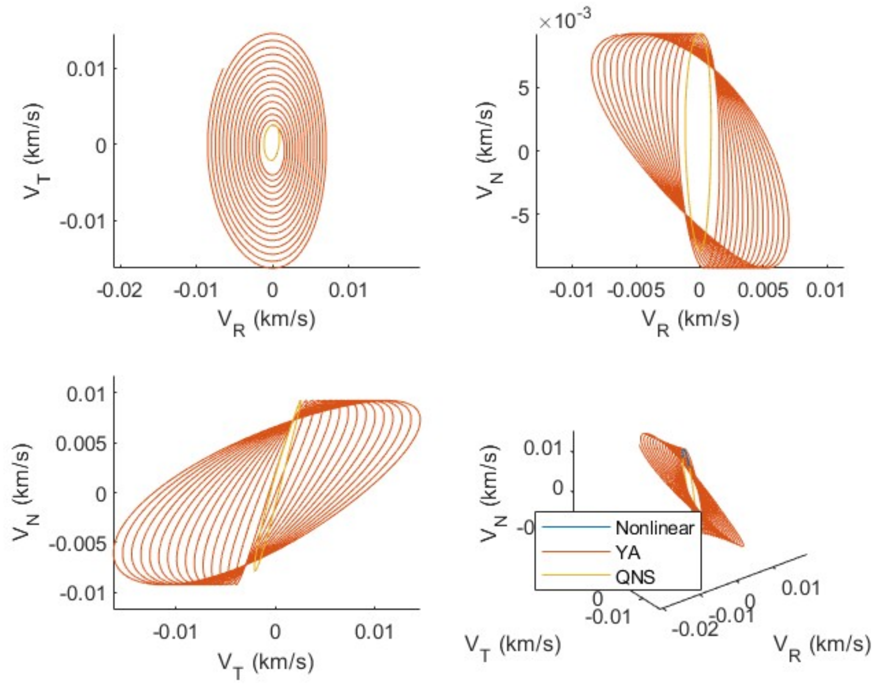


Figure 3.8: Deputy Velocity in Chief's Rotating RTN Frame over 15 Orbital Periods: Numerical Nonlinear vs Y.A. vs Geometric Linear Mapping

## ERRORS

Errors were calculated by simple subtraction of the calculated RTN positions(numerical - YA/QNS)

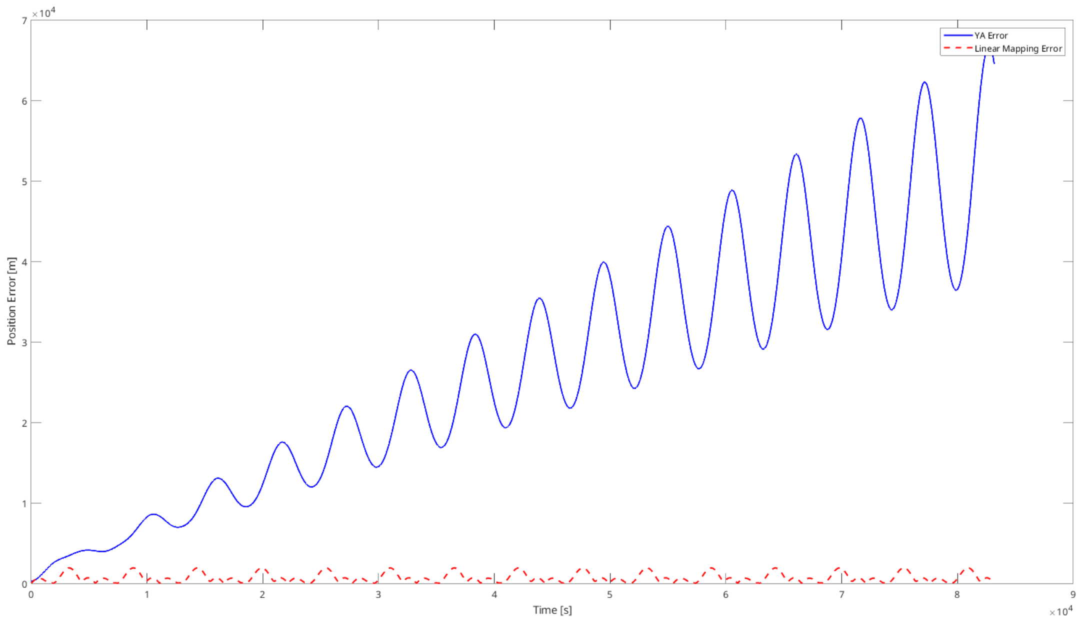


Figure 3.9: Error of Deputy Position in Chief's Rotating RTN Frame over 15 Orbits for Y.A. and Geometric Linear Mapping

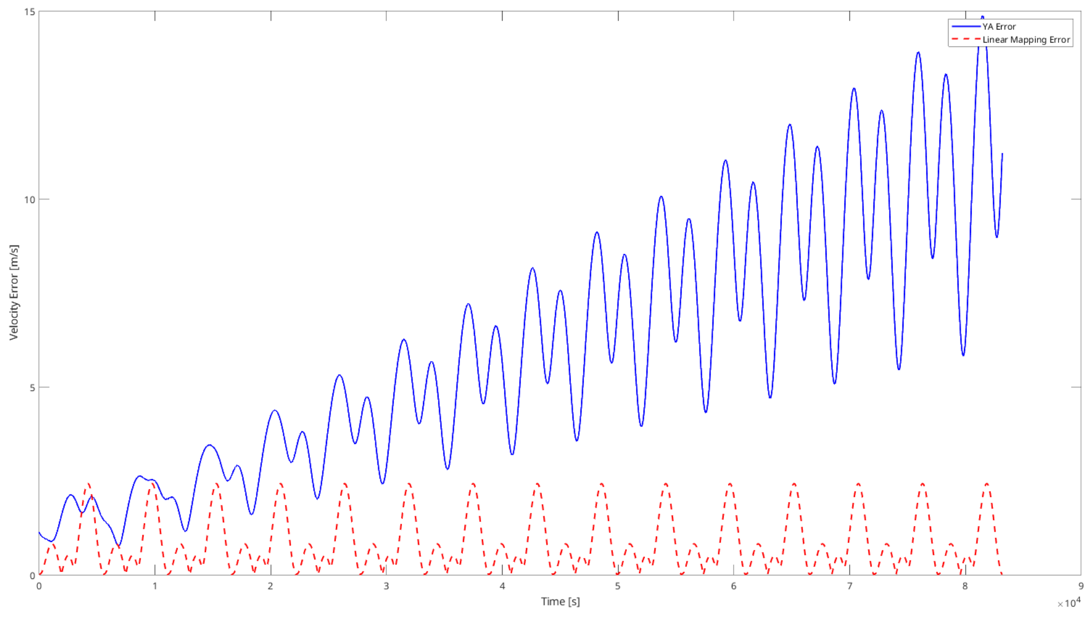


Figure 3.10: EError of Deputy Velocity in Chief's Rotating RTN Frame over 15 Orbits for Y.A. and Geometric Linear Mapping

The error in the Y.A. solution is much larger than the error from linear geometric mapping. This is because YA is only a first-order linearization that neglects accumulating nonlinear eccentricity effects, it drifts over time, whereas the geometric mapping's secondâorder expansion captures those terms and stays more accurate.

### 3.2.9 i) Repeat Exercise for Varying Initial Orbital Elements

#### i. Introducing Delta a

| Spacecraft | $a$ [km] | $e$ | $i$ [°] | $\Omega$ [°] | $\omega$ [°] | $f$ [°] |
|---|---|---|---|---|---|---|
| Chief | 6771 | 0.1005 | 51.64 | 257.00 | 0.00 | 30.00 |
| Deputy | 6770.5 | 0.1006 | 51.69 | 257.05 | 0.05 | 29.95 |

Table 3.10: Initial absolute Keplerian orbital elements for the chief and deputy spacecraft with small $\delta$a introduced

Justification: The relative distance at the start is still valid for small separation assumptions (separation = 3.7163 km), but a slight difference in a should be sufficient to show along track drift over 15 orbital periods.

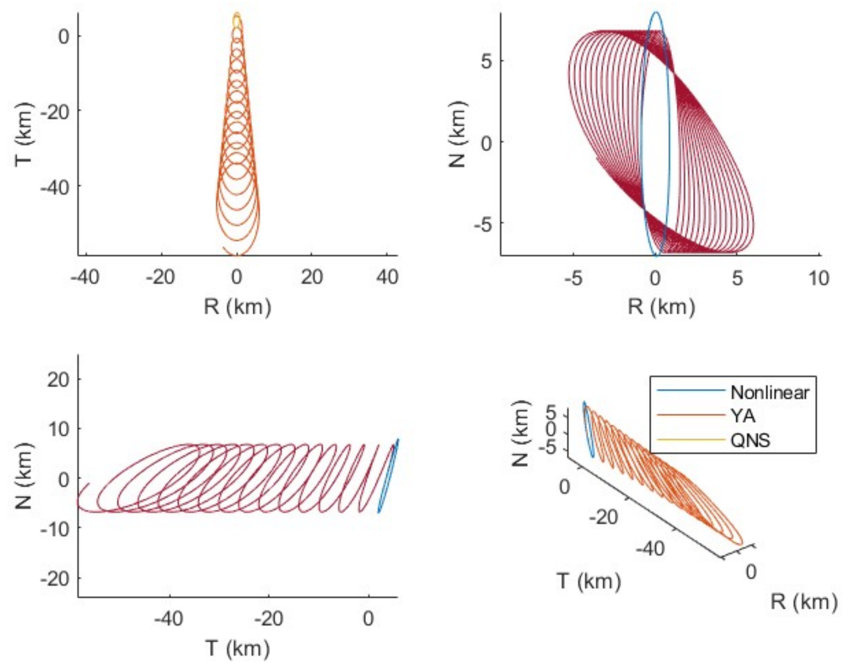


Figure 3.11: Deputy Position in Chief's Rotating RTN Frame over 15 Orbital Periods: Numerical Nonlinear vs Y.A. vs Geometric Linear Mapping

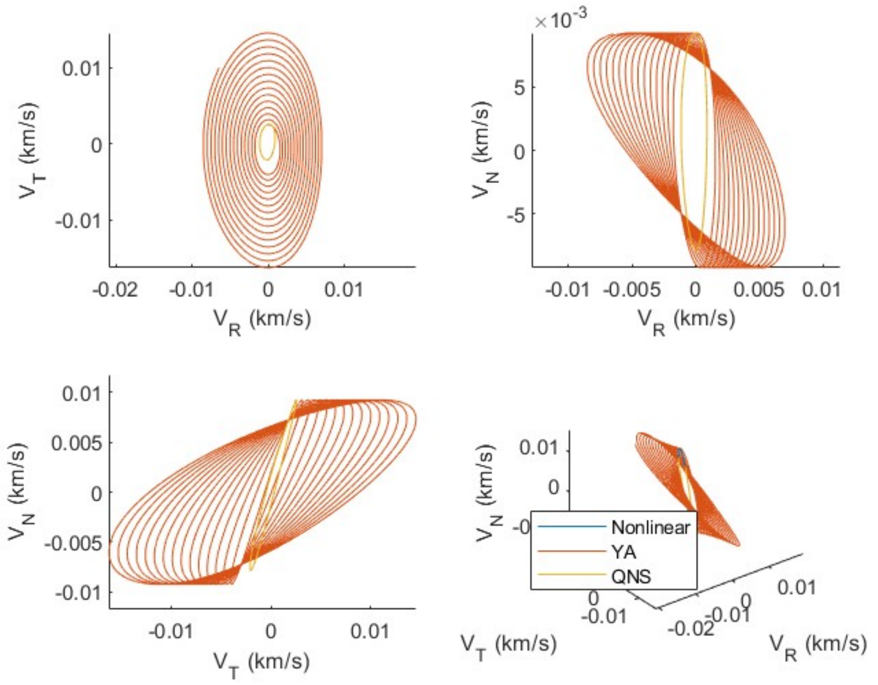


Figure 3.12: Deputy Velocity in Chief's Rotating RTN Frame over 15 Orbital Periods: Numerical Nonlinear vs Y.A. vs Geometric Linear Mapping

**ERRORS**

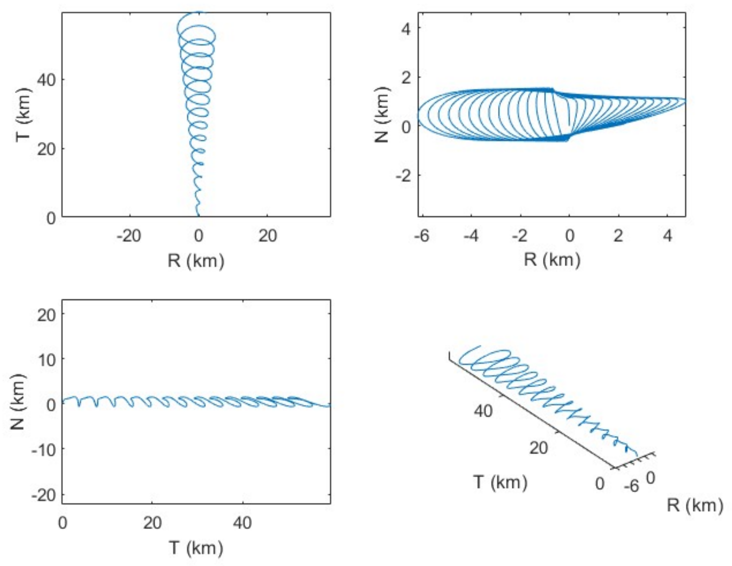


Figure 3.13: Error of Deputy Position in Chief's Rotating RTN Frame over 15 Orbits: Y.A.

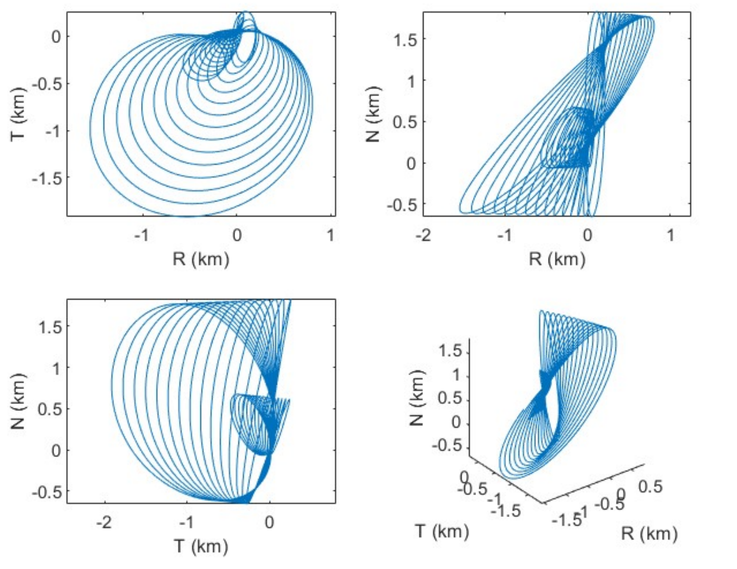


Figure 3.14: Error of Deputy Position in Chief's Rotating RTN Frame over 15 Orbits: Geometric Linear Mapping

We can see that there is now significant along track drift in both Nonlinear and YA, with QNS drift remaining about the same (small). QNS appears to not capture the along track drift well.

**ii. Large Eccentricity**

| Spacecraft | $a$ [km] | $e$ | $i$ [°] | $\Omega$ [°] | $\omega$ [°] | $f$ [°] |
|---|---|---|---|---|---|---|
| Chief | 6771 | 0.5005 | 51.64 | 257.00 | 0.00 | 30.00 |
| Deputy | 6771 | 0.5006 | 51.69 | 257.05 | 0.05 | 29.95 |

Table 3.11: Initial absolute Keplerian orbital elements for the chief and deputy spacecraft with small $\delta$a introduced

Justification: The relative distance at the start is still valid for small separation assumptions (separation = 2.3409 km), but e was increased to 0.5.

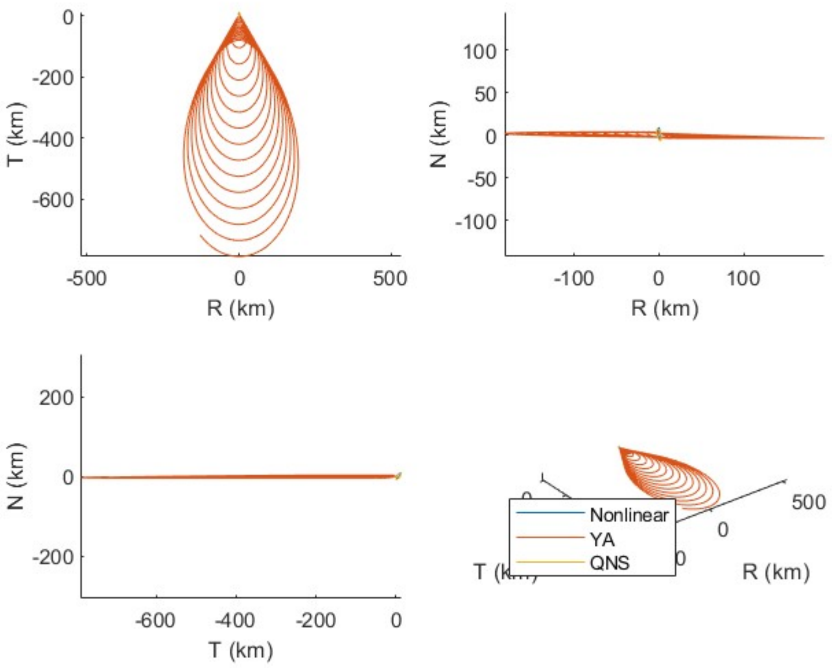


Figure 3.15: Deputy Position in Chief's Rotating RTN Frame over 15 Orbital Periods: Numerical Nonlinear vs Y.A. vs Geometric Linear Mapping

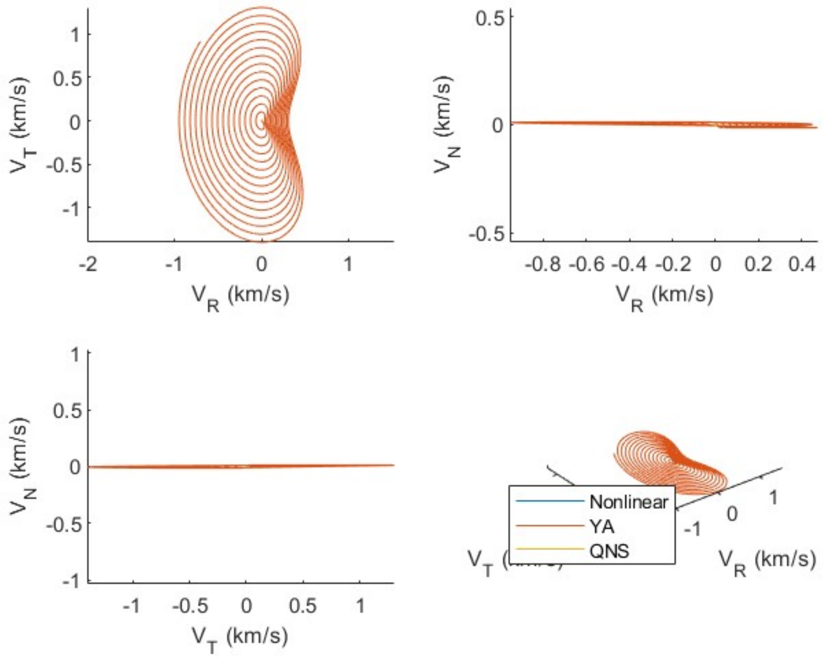


Figure 3.16: Deputy Velocity in Chief's Rotating RTN Frame over 15 Orbital Periods: Numerical Nonlinear vs Y.A. vs Geometric Linear Mapping

The YA results are vastly different from the QNS or Nonlinear results. To show the comparison between QNS and Nonlinear, here is a graph omitting YA:

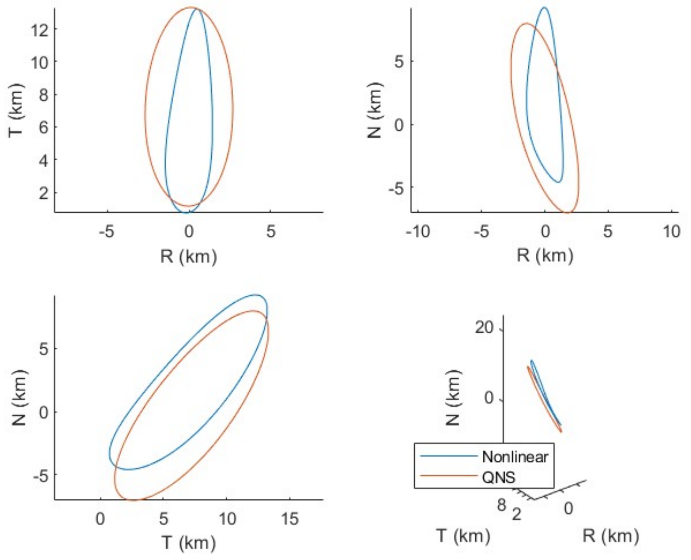


Figure 3.17: Deputy Position in Chief's Rotating RTN Frame over 15 Orbital Periods: Numerical Nonlinear vs Geometric Linear Mapping

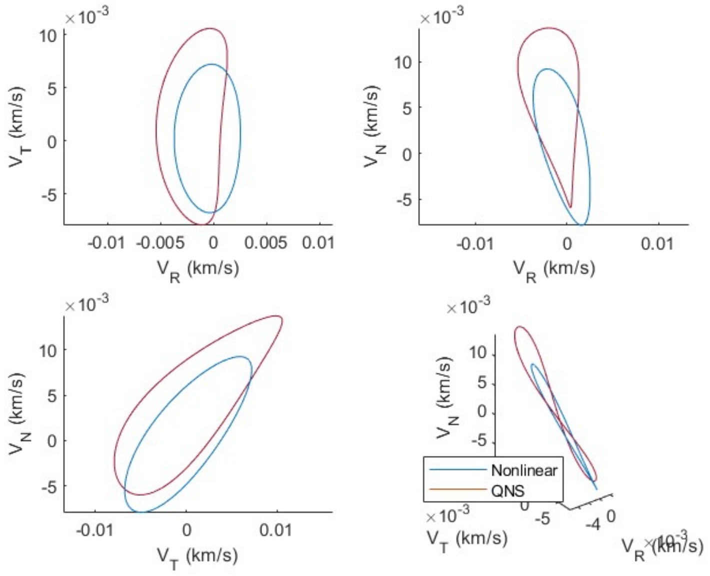


Figure 3.18: Deputy Velocity in Chief's Rotating RTN Frame over 15 Orbital Periods: Numerical Nonlinear vs Geometric Linear Mapping

**ERRORS**

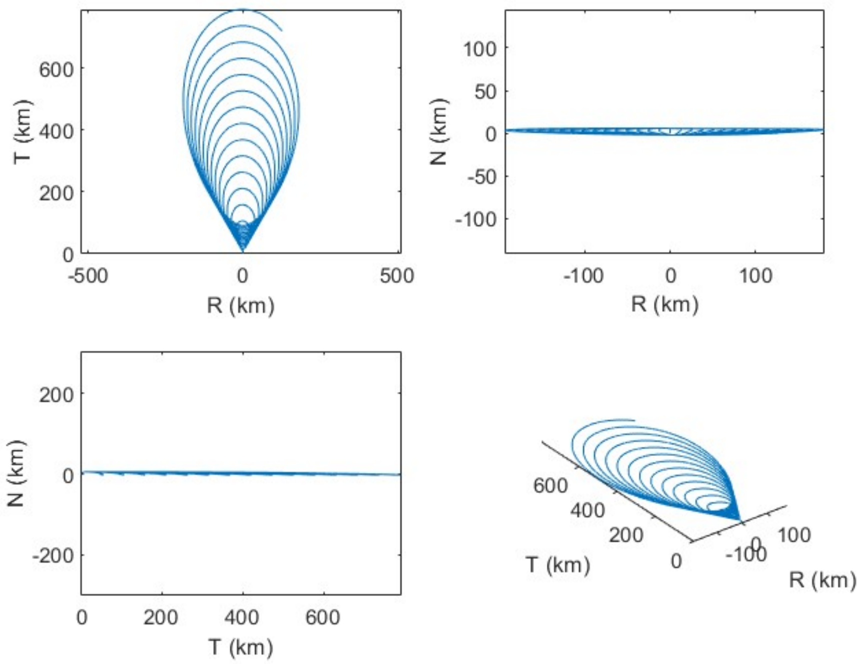


Figure 3.19: Error of Deputy Position in Chief's Rotating RTN Frame over 15 Orbits: Y.A.

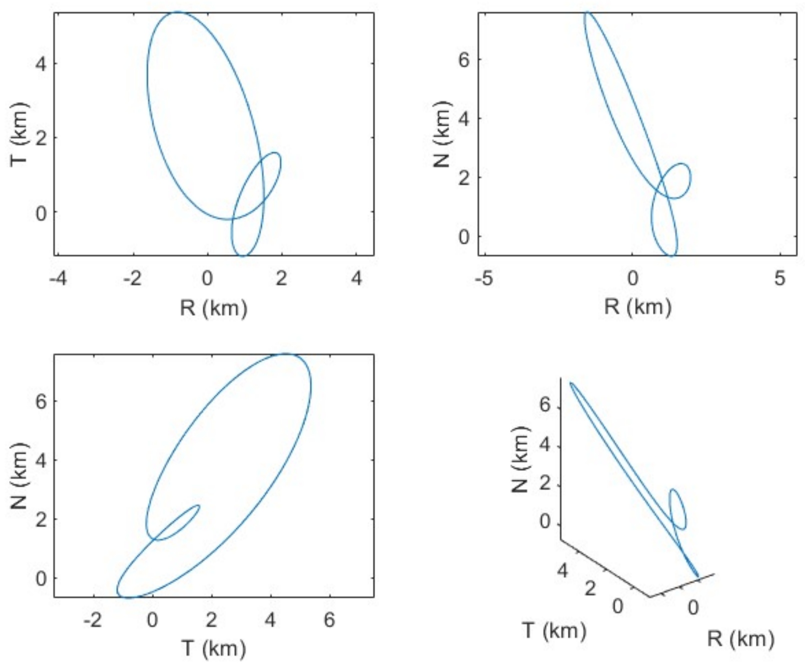


Figure 3.20: Error of Deputy Position in Chief's Rotating RTN Frame over 15 Orbits: Geometric Linear Mapping

Again, Y.A. is much less accurate than Geometric Mapping. This is for the same reasons as stated earlier. There is more tangential drift from all solutions when a delta a is introduced.

## 4 Problem Set 4

The code used for this assignment is available in our GitHub repository [1] and the Appendix. See ”Problem Set 4 Code” for code for each section. Additionally, see ”Propagators” and ”Utils” sections.

### 4.1 a) Osculating Initial Conditions for Chief and Deputy (Part 1)

We use the same osculating initial conditions for the chief spacecraft as in the previous assignment. These parameters lie within the validity range of the Hill–Clohessy–Wiltshire (HCW) linearized equations, i.e., they assume a small eccentricity ($e \ll 1$), which we justify below.

| Spacecraft | $a$ (km) | $e$ | $i$ (deg) | $\Omega$ (deg) | $\omega$ (deg) | $\nu$ (deg) |
|---|---|---|---|---|---|---|
| Chief (ISS) | 6771 | 0.0005 | 51.64 | 257 | 0 | 30 |
| Deputy (Dragon) | 6771 | 0.0015 | 52.14 | 257.5 | 0.5 | 25 |

Table 4.1: Initial absolute Keplerian orbital elements for chief and deputy

These elements represent a near-circular, low-Earth orbit consistent with the assumptions of linearized relative motion models.

Converting to QNS relative orbital elements given this chief and deputy, we have:

| | $\delta a$ [m] | $\delta\lambda$ [m] | $\delta e_x$ [m] | $\delta e_y$ [m] | $\delta i_x$ [m] | $\delta i_y$ [m] |
|---|---|---|---|---|---|---|
| Deputy (rel.) | 0 | -500310 | 6770 | 90 | 59090 | 46330 |

Table 4.2: Initial relative quasi-nonsingular orbital elements (rQNS) of the deputy

### 4.2 b) Osculating Initial Conditions for Deputy (Part 2)

We now consider a second set of initial conditions, this time expressed in the relative quasi-nonsingular (rQNS) orbital element form.

| | $\delta a$ [m] | $\delta\lambda$ [m] | $\delta e_x$ [m] | $\delta e_y$ [m] | $\delta i_x$ [m] | $\delta i_y$ [m] |
|---|---|---|---|---|---|---|
| Deputy (rel.) | 0 | 100 | 50 | 100 | 30 | 200 |

Table 4.3: Initial relative quasi-nonsingular orbital elements (rQNS) of the deputy

These elements are scaled by the chief's semi-major axis $a_c = 6771$ km, and thus expressed in meters.

To simulate full nonlinear dynamics, we must convert the deputy's initial conditions from the relative rQNS orbital element vector $\delta\boldsymbol{\alpha_{QNS}}$ into an absolute osculating Keplerian state vector $\boldsymbol{\alpha}_d = [a_d, e_d, i_d, \Omega_d, \omega_d, f_d]$.

The chief's orbital elements $\boldsymbol{\alpha}_c = [a_c, e_c, i_c, \Omega_c, \omega_c, f_c]$ serve as the reference frame. The following equations describe the complete analytical mapping:

$$a_d = a_c(1 + \delta a),$$

$$\begin{aligned}
i_d &= i_c + \delta i_x, \\
\Omega_d &= \Omega_c + \delta i_y \sin(i_c), \\
\alpha &= \delta e_y + e_c \sin \omega_c, \\
\beta &= \delta e_x + e_c \cos \omega_c, \\
e_d &= \sqrt{\alpha^2 + \beta^2}, \\
\omega_d &= \tan^{-1}\left(\frac{\alpha}{\beta}\right).
\end{aligned}$$

To compute the mean anomaly of the chief $M_c$, we first compute the eccentric anomaly $E_c$ using:

$$\begin{aligned}
E_c &= 2\tan^{-1}\left(\sqrt{\frac{1-e_c}{1+e_c}}\tan\left(\frac{f_c}{2}\right)\right), \\
M_c &= E_c - e_c \sin E_c.
\end{aligned}$$

The deputy's mean anomaly is then given by:

$$M_d = \delta\lambda - (\Omega_d - \Omega_c)\cos(i_c) + (M_c + \omega_c) - \omega_d.$$

The eccentric anomaly $E_d$ of the deputy is recovered from $M_d$ and $e_d$ by solving Kepler's equation using the Newton-Raphson method, implemented in the function `newton_raphson`:

$$E_d = \texttt{newton\_raphson}(M_d, e_d)$$

where Kepler's equation is:

$$M_d = E_d - e_d \sin E_d$$

Finally, the true anomaly of the deputy is computed as:

$$f_d = 2\tan^{-1}\left(\sqrt{\frac{1+e_d}{1-e_d}}\tan\left(\frac{E_d}{2}\right)\right)$$

This completes the transformation from rQNS elements to full osculating orbital elements of the deputy at time $t_0$.

This mapping is implemented in the helper function `rQNSOE2OE` in our codebase. It takes the chief's osculating orbital elements as input and applies the above transformation to compute the deputy's absolute orbital state at time $t_0$.

| | $a$ [km] | $e$ | $i$ [°] | $\Omega$ [°] | $\omega$ [°] | $\nu$ [°] |
|---|---|---|---|---|---|---|
| Deputy | 6771.00 | 5.1569e-04 | 51.64 | 257.00 | 45.58 | 29.42 |

Table 4.4: Recovered osculating Keplerian orbital elements of deputy at $t_0 = 0$

We confirm that both chief and deputy orbits lie within the validity region for HCW linear dynamics:

- Relative separation: $\rho = 181.54$ m, satisfying $\rho/r_0 \ll 1$
- Eccentricities: $e_c = 0.0005$, $e_d = 0.0005$, satisfying $e_c, e_d \ll 1$

### 4.3 c) Numerical Propagation and Element Extraction

We begin by converting the osculating Keplerian elements of both chief and deputy into Cartesian state vectors (position $\mathbf{r}$ and velocity $\dot{\mathbf{r}}$) in the Earth-centered inertial frame. These state vectors are then numerically integrated for 15 orbital periods with MATLAB's `ode113`, using the equations of motion

$$\ddot{\mathbf{r}} = -\frac{\mu}{r^3}\,\mathbf{r} + \mathbf{a}_{J_2},$$

where $\mathbf{a}_{J_2}$ is included only in the perturbed cases. Integration tolerances are set to `RelTol`=1e-12, `AbsTol`=1e-14.

At each integration step:

1. Convert the updated $(\mathbf{r}, \dot{\mathbf{r}})$ to osculating orbital elements via ECI-to-OE routines.
2. Transform to quasi-nonsingular form (QNS) for numerical stability when $e \ll 1$.
3. Compute relative QNS (rQNS) offsets between deputy and chief.
4. Use mapping to obtain mean (secular) slopes for both absolute QNS elements and the relative QNS elements, plotted in red dashed lines.

We consider four cases:

1. Case 1: original ICs from (a), no $J_2$
2. Case 2: original ICs from (a), with $J_2$
3. Case 3: modified deputy offsets from (b), no $J_2$
4. Case 4: modified deputy offsets from (b), with $J_2$

Results are shown below.

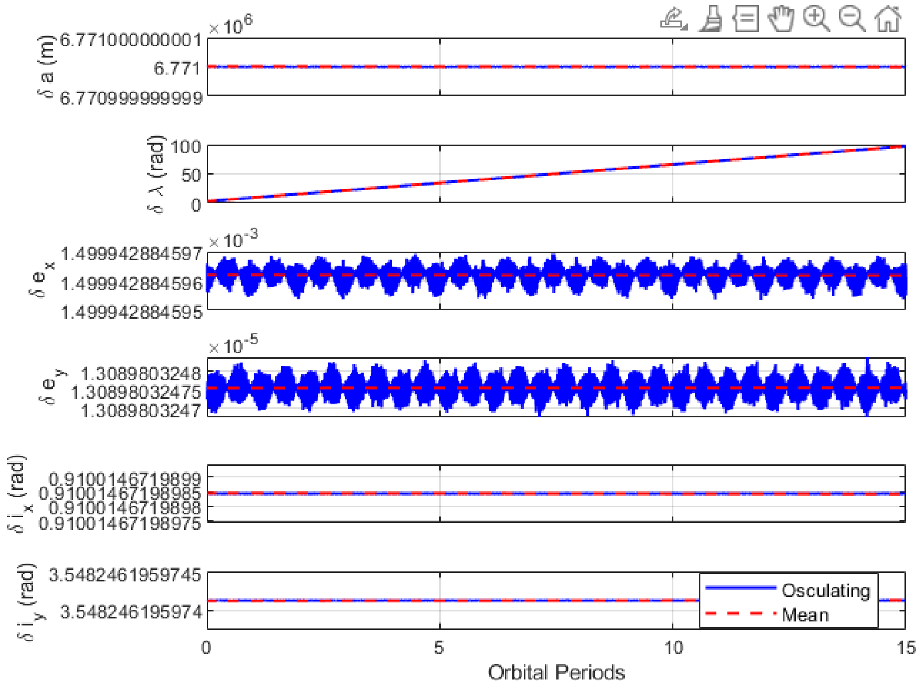


Figure 4.1: Absolute QNS elements for Case 1 (no $J_2$).

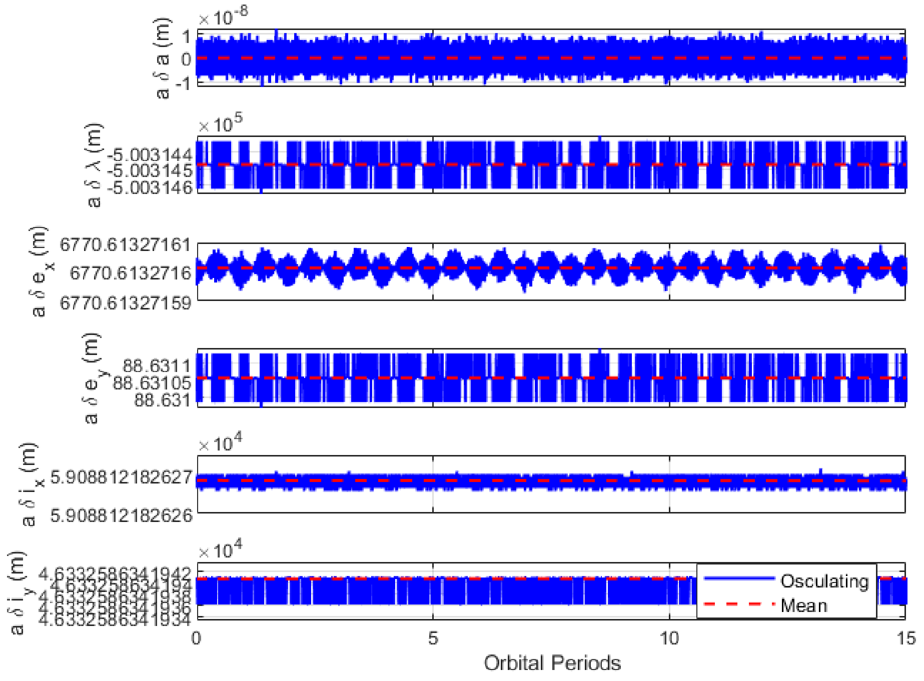


Figure 4.2: Relative QNS elements for Case 1 (no $J_2$).

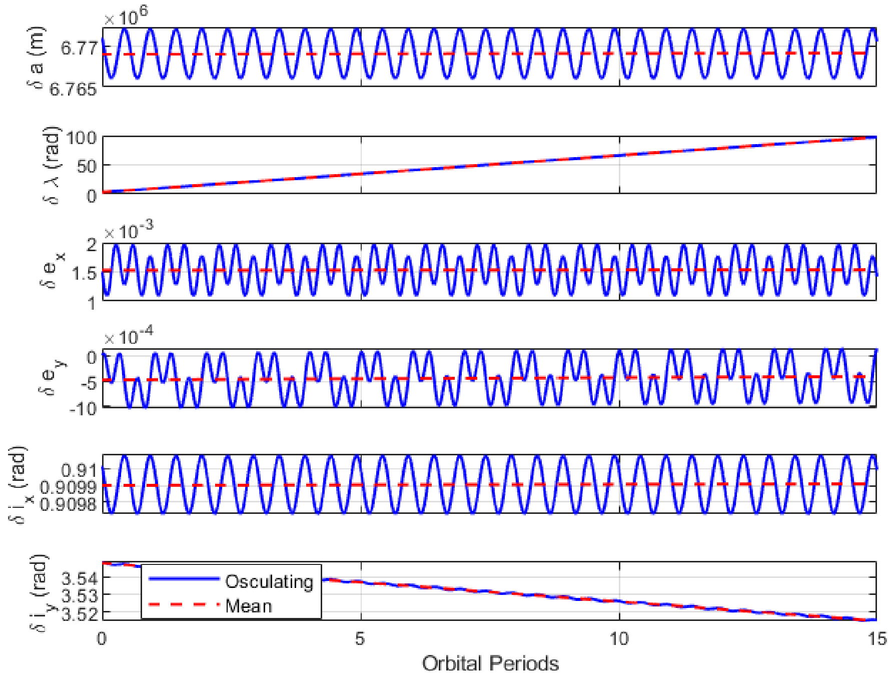


Figure 4.3: Absolute QNS elements for Case 2 (with $J_2$).

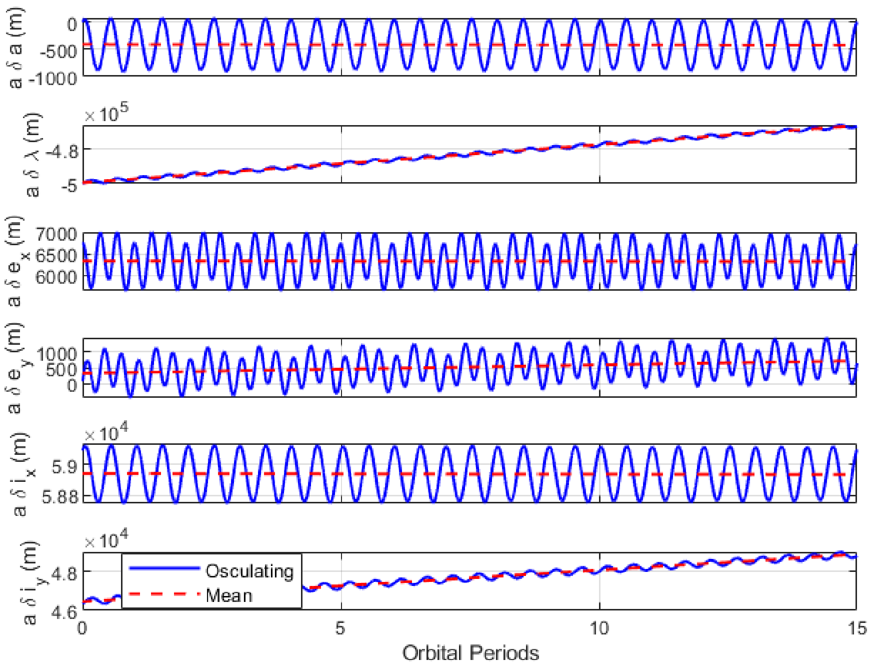


Figure 4.4: Relative QNS elements for Case 2 (with $J_2$).

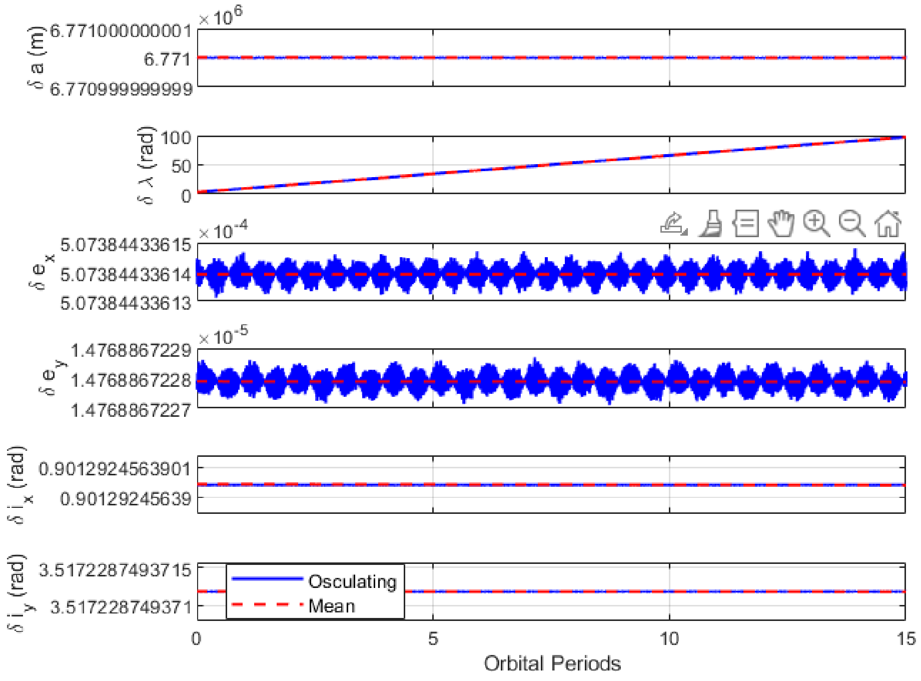


Figure 4.5: Absolute QNS elements for Case 3 (no $J_2$, modified deputy).

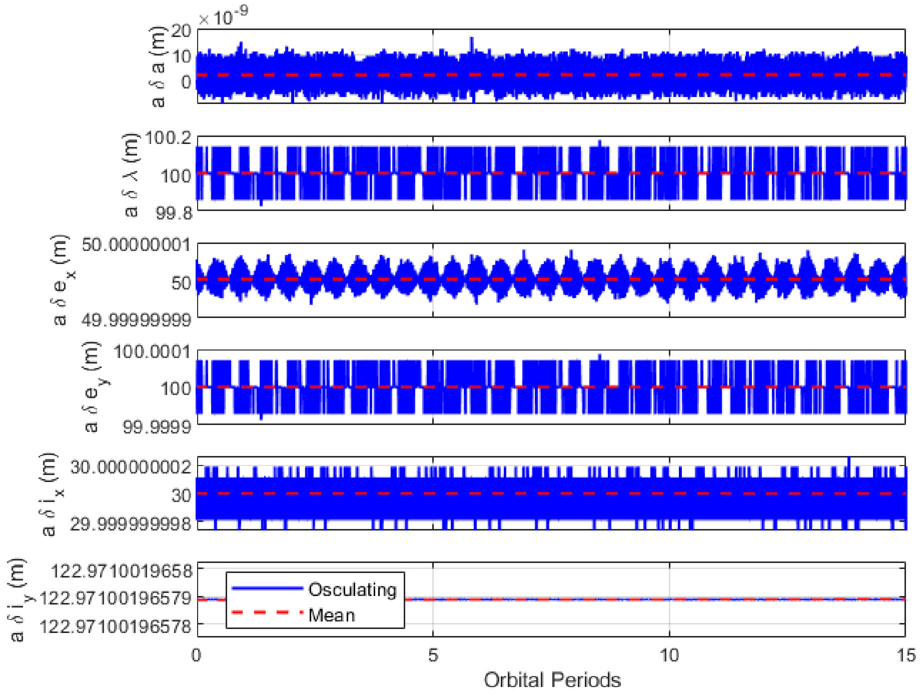


Figure 4.6: Relative QNS elements for Case 3 (no $J_2$, modified deputy).

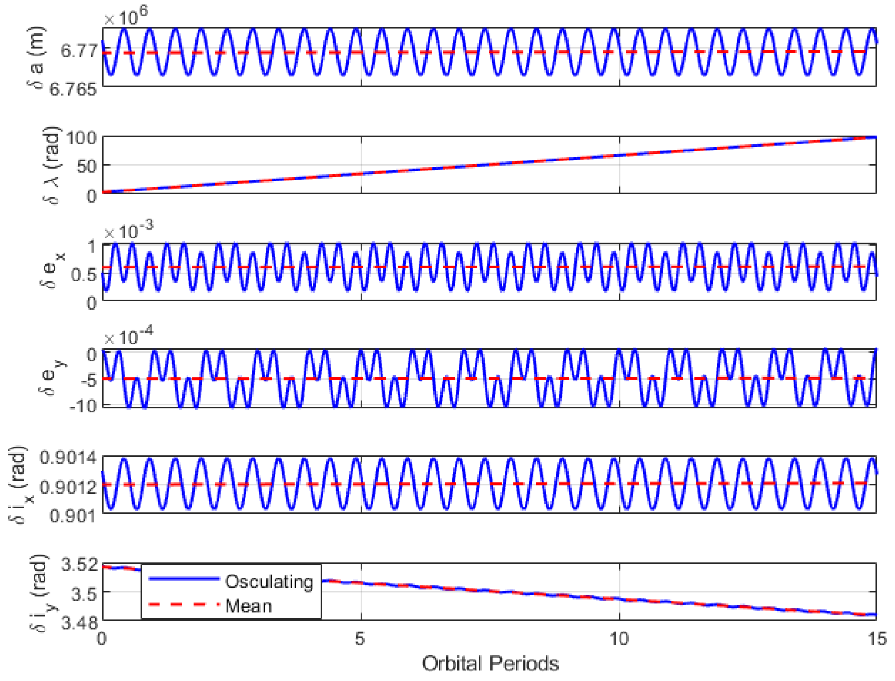


Figure 4.7: Absolute QNS elements for Case 4 (with $J_2$, modified deputy).

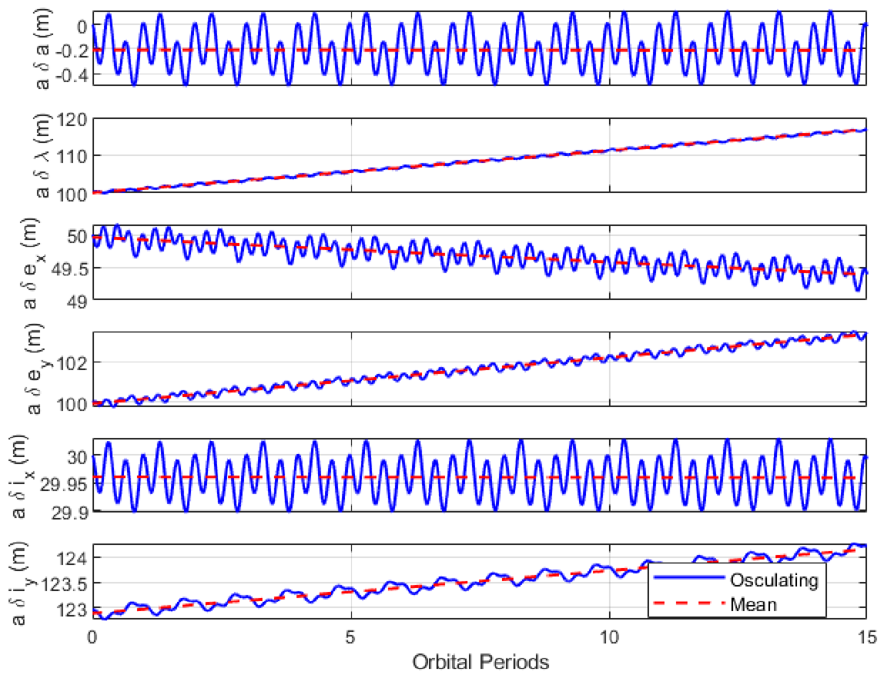


Figure 4.8: Relative QNS elements for Case 4 (with $J_2$, modified deputy).

**Secular Trends and Takeaways**

- **Unperturbed (Cases 1 and 3):** Osculating and mean QNS elements coincide and remain constant, confirming no secular drift under two-body dynamics.
- **Perturbed (Cases 2 and 4):** The $J_2$ perturbation produces steady trends in key elements:
  - Mean longitude $\lambda$ drifts steadily (along-track separation).
  - Nodal angle (reflected in $\delta i_y$) shifts over time (out-of-plane drift).
  - Eccentricity vector $(e_x, e_y)$ rotates slowly (perigee precession).
- **Implication:** These secular rates must be counteracted (for example, via the normal burn in Part 6) to maintain formation stability.

## 4.4 d) RTN Trajectories

Using the position and velocity histories from Case 2, we compute the deputy's relative coordinates in the local RTN frame at each time step. Figure below shows the resulting trajectories for both the unperturbed (red dashed) and $J_2$-perturbed (blue solid) cases, projected in 3D as well as in the TR, NR, and TN planes.

In the absence of $J_2$, the deputy traces a fixed, closed elliptical path consistent with linearized HCW theory. The shape remains centered and repeats each orbit. When $J_2$ is included, the ellipse thickens and shifts progressively in the along-track (T) direction. This is especially visible in the TR and TN projections, where the blue trajectory shows a clear secular drift in T, while remaining bounded in the radial (R) and cross-track (N) directions. This confirms that the dominant effect of $J_2$ is along-track drift due to differential nodal and perigee precession, with negligible impact on radial or out-of-plane relative position.

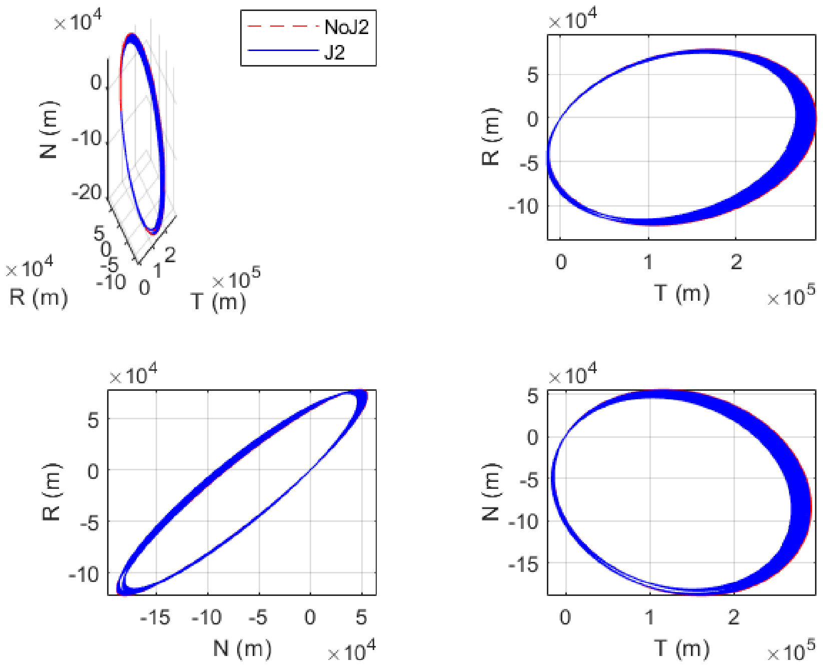


Figure 4.9: Relative motion in the RTN frame for Case 2: unperturbed (red dashed) vs. $J_2$-perturbed (blue solid).

## 4.5 e) State-Space Plots

Figure 4.10 shows the unperturbed relative QNSROE state space (Case 1). Each subplot collapses to a fixed point or horizontal line: the $a\,\delta e_x$ vs. $a\,\delta e_y$ and $a\,\delta i_x$ vs. $a\,\delta i_y$ points are stationary. These results confirm that without $J_2$ there is no secular drift beyond numerical noise.

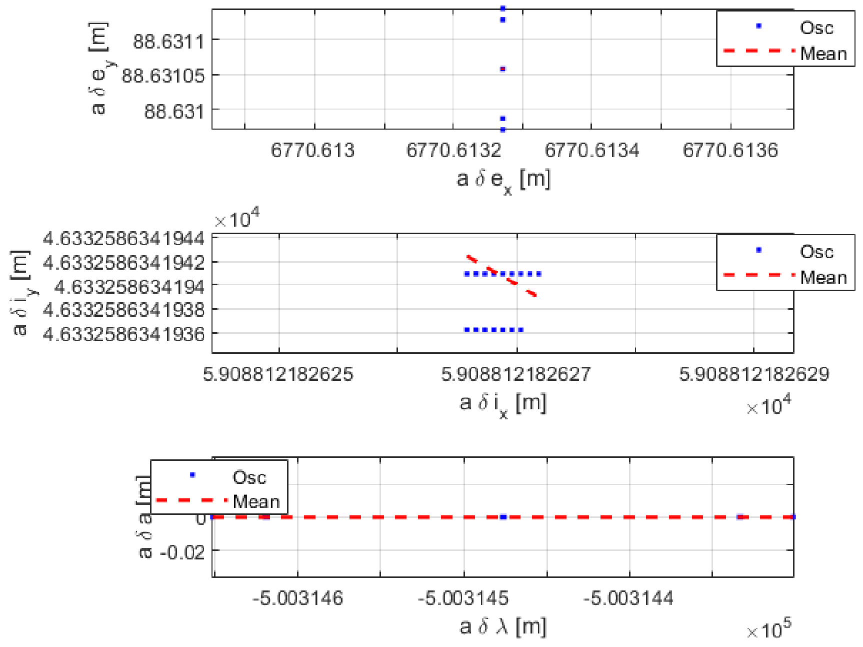


Figure 4.10: Unperturbed relative state space (Case 1): osculating in blue, mean in red dashed.

Figure 4.11 shows the $J_2$-perturbed state space (Case 2). The eccentricity loop's center shifts over time along both the $a\,\delta e_x$ and $a\,\delta e_y$ axes, indicating secular drift in those components. The inclination oscillation moves upward, reflecting drift in $a\,\delta i_y$. The $a\,\delta\lambda$ vs. $a\,\delta a$ plot exhibits a sloped mean line, indicating a steady change in $a\,\delta\lambda$, while $a\,\delta a$ remains zero. These secular trends are exactly those expected from $J_2$-induced perigee and nodal precession. There is oscillation on all relative orbital elements, as expected.

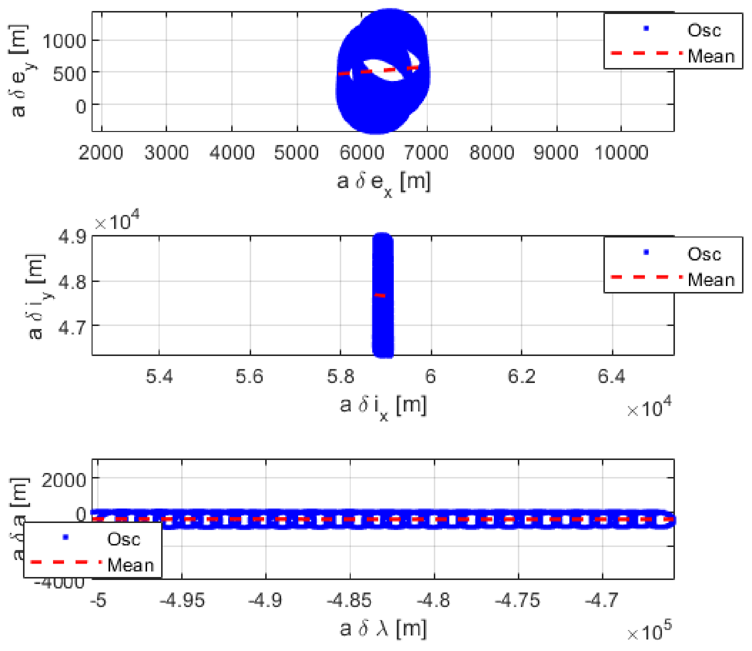


Figure 4.11: $J_2$-perturbed relative state space (Case 2): osculating in blue, mean in red dashed.

## 4.6 f) Removal of $J_2$ secular drift

$$\left.\frac{d\phi}{du}\right|_{J2} = \frac{3}{2}\,\gamma\,(5\cos^2(i) - 1) \tag{21}$$

$$\gamma = \frac{J2}{2}\left(\frac{R_E}{a}\right)^2\frac{1}{\eta^4} \tag{22}$$

$$\left.\frac{d(\delta i)}{du}\right|_{J2} = 3\,\gamma\,\sin^2(i)\,\delta i_x \tag{23}$$

$$\left.\frac{d(\delta\lambda)}{du}\right|_{J2} = -\frac{21}{2}\left(\gamma\,\sin(2\,i)\,\delta i_x + \frac{1}{7}\,\delta a\right) \tag{24}$$

To remove all J2 secular drift we require both

$$\left.\frac{d(\delta i)}{du}\right|_{J2} = 0 \quad\Longrightarrow\quad 3\,\gamma\,\sin^2(i)\,\delta i_x = 0$$

and

$$\left.\frac{d(\delta\lambda)}{du}\right|_{J2} = 0 \quad\Longrightarrow\quad -\tfrac{21}{2}\left(\gamma\,\sin(2\,i)\,\delta i_x + \tfrac{1}{7}\,\delta a\right) = 0.$$

Since $J2 \neq 0$, $R_E \neq 0$, $a \neq 0$, $\eta \neq 0$, and for an inclined chief orbit $0 < i < \pi$ we have $\sin^2(i) > 0$. Hence

$$\delta i_x = 0 \quad\text{and}\quad \delta a = 0.$$

Thus the *minimum* change to the initial ROEs is

$$\boxed{\Delta\,\delta i_x = -\,\delta i_x^{(0)}.}$$

Given

| | $\delta a$ [m] | $\delta\lambda$ [m] | $\delta e_x$ [m] | $\delta e_y$ [m] | $\delta i_x$ [m] | $\delta i_y$ [m] |
|---|---|---|---|---|---|---|
| Deputy (rel.) | 0 | -500310 | 6770 | 90 | 59090 | 46330 |

Table 4.5: Initial relative quasi-nonsingular orbital elements (rQNS) of the deputy

In QNSROE form, $\delta i_x = a\,\Delta i$, so

$$\Delta i = \frac{\Delta\,\delta i_x}{a} = \frac{-59090}{6.771\times 10^6} \approx -8.7266\times 10^{-3}\text{ rad.}$$

For $a = 6.771\times 10^6$ m and $\mu = 3.986\times 10^{14}\ m^3/s^2$,

$$n = \sqrt{\frac{\mu}{a^3}} \approx 0.0011335\text{ rad/s.}$$

Applying Gauss's out-of-plane GVE at $u_M = 0°$ (ascending node),

$$\Delta v_N = n\,a\,\Delta i = 0.0011335\times 6.771\times 10^6\times(-8.7266\times 10^{-3}) \approx -67.0\text{ m/s.}$$

Thus the required impulsive maneuver is

$$\boxed{\Delta v = 67.0\text{ m/s}\quad\text{normal to the orbital plane, at } u_M = 0°.}$$

## 4.7 g) Simulation Re-run

The simulation was re-run with the same initial conditions, except that $\delta i_x$ was set to zero. As expected, the RTN relative position and velocity time series in Figure 4.12 remain unchanged, but the secular drift in the inclination and mean-argument components is now eliminated. The eccentricity components still show the small secular drift variations inherent to the HCW motion.

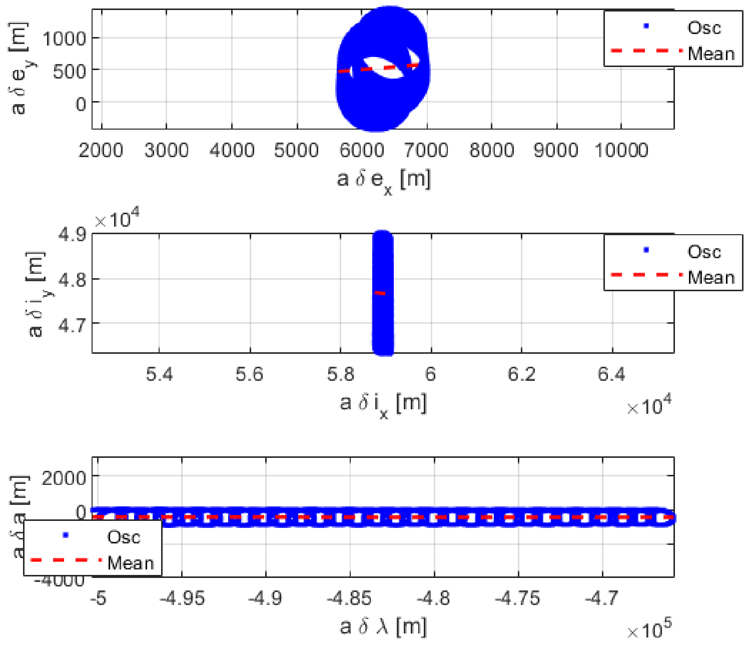


Figure 4.12: $J_2$-perturbed relative state space (Case 2) with $\delta i_x = 0$.

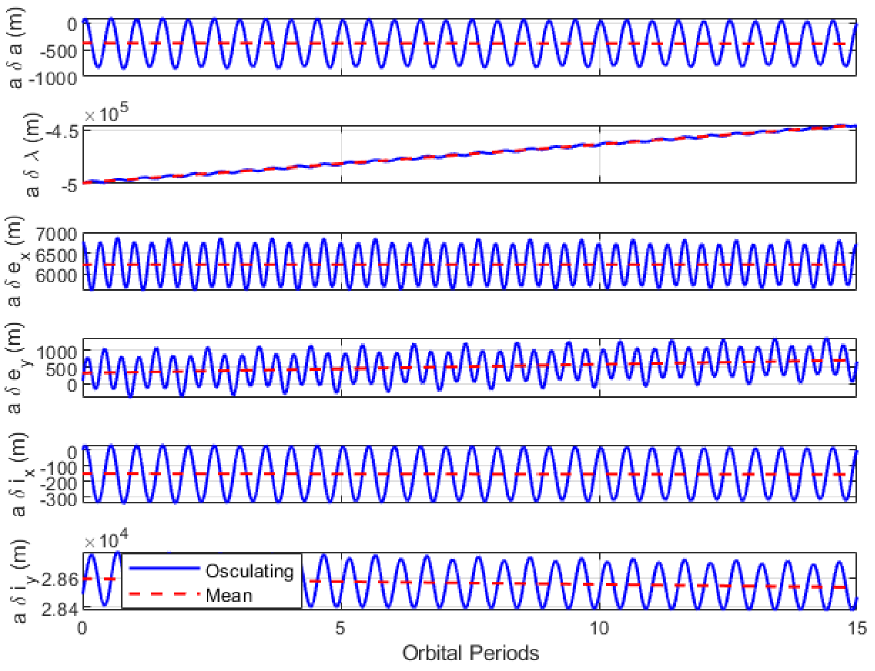


Figure 4.13: Relative orbital element histories with $\delta i_x = 0$.

### 4.8 h) Analytical $J_2$ STM and comparison with the numerical results

See code in "Problem Set 4" section in Appendix or Github.

The secular evolution of the relative quasi-nonsingular elements is governed by the State-Transition Matrix (STM). For a chief with osculating elements $\boldsymbol{\alpha}_c = (a, e, i, \Omega, \omega, M)$ and a time interval $\tau = t - t_0$, the closed-form STM reads

$$\delta\boldsymbol{\alpha}(t) = \boldsymbol{\Phi}_{\text{qns}}^{J_2}(\tau), \delta\boldsymbol{\alpha}(t_0), \qquad \delta\boldsymbol{\alpha} = \begin{bmatrix}\delta a & \delta\lambda & \delta e_x & \delta e_y & \delta i_x & \delta i_y\end{bmatrix}^{!\top} \tag{25}$$

where (chief-dependent) scalars that appear in the matrix are

$$n = \sqrt{\tfrac{\mu}{a^3}}, \qquad \eta = \sqrt{1-e^2}, \qquad p = a(1-e^2), \qquad \kappa = \tfrac{3}{2}\,J_2\,n\Big(\tfrac{R_e}{p}\Big)^2,$$

$$P = \tfrac{1}{2}(1-5\cos^2 i), \quad Q = \tfrac{1}{4}(5\cos^2 i - 1), \quad R = \tfrac{1}{2}\cos i, \quad S = \tfrac{1}{4}\sin i \cos i, \quad T = Q, \quad F = \frac{1}{\eta}, \quad G = \frac{1}{\eta^3},$$

$$\dot\omega = \kappa Q, \qquad \dot\Omega = -2\kappa R,$$

and the initial and final chief eccentricity-vector components are

$$e_x^{(i)} = e\cos\omega, \quad e_y^{(i)} = e\sin\omega, \qquad e_x^{(f)} = e\cos(\omega+\dot\omega\tau),\ e_y^{(f)} = e\sin(\omega+\dot\omega\tau).$$

Only the non-zero entries of $\boldsymbol{\Phi}_{\text{qns}}^{J_2}$ differ from the identity; they are reproduced here for clarity:

$$\boldsymbol{\Phi}_{\text{qns}}^{J_2} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ -\big(\tfrac{3}{2}n + \tfrac{7}{2}\kappa\eta P\big)\tau & 1 & \kappa e_x^{(i)} FGP\tau & \kappa e_y^{(i)} FGP\tau & -\kappa FS\tau & 0 \\ \tfrac{7}{2}\kappa e_y^{(f)} Q\tau & 0 & \cos\dot\omega\tau - 4\kappa e_x^{(i)} e_y^{(f)} GQ\tau & -\sin\dot\omega\tau - 4\kappa e_y^{(i)} e_y^{(f)} GQ\tau & 5\kappa e_y^{(f)} S\tau & 0 \\ -\tfrac{7}{2}\kappa e_x^{(f)} Q\tau & 0 & \sin\dot\omega\tau + 4\kappa e_x^{(i)} e_x^{(f)} GQ\tau & \cos\dot\omega\tau + 4\kappa e_y^{(i)} e_x^{(f)} GQ\tau & -5\kappa e_x^{(f)} S\tau & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ \tfrac{7}{2}\kappa S\tau & 0 & -4\kappa e_x^{(i)} GS\tau & -4\kappa e_y^{(i)} GS\tau & 2\kappa T\tau & 1 \end{bmatrix}.$$

Two different initial relative states are propagated using the analytical $J_2$ state-transition matrix:

(1) $\delta\boldsymbol{\alpha}_0^{(1)} = (0,\ -500310,\ 6770,\ 90,\ 59090,\ 46330)^\top$

(2) $\delta\boldsymbol{\alpha}_0^{(2)} = (0,\ -500310,\ 6770,\ 90,\ 0,\ 46330)^\top$ (with $\delta i_x = 0$ to cancel inclination drift)

These vectors are given in meters and correspond to the full relative quasi-nonsingular orbital element state: $a\delta a$, $a\delta\lambda$, $a\delta e_x$, $a\delta e_y$, $a\delta i_x$, $a\delta i_y$.

For the first set of initial relative QNS orbital elements, we analyze the orbit geometry using the same plots as in the previous section. The results are shown below:

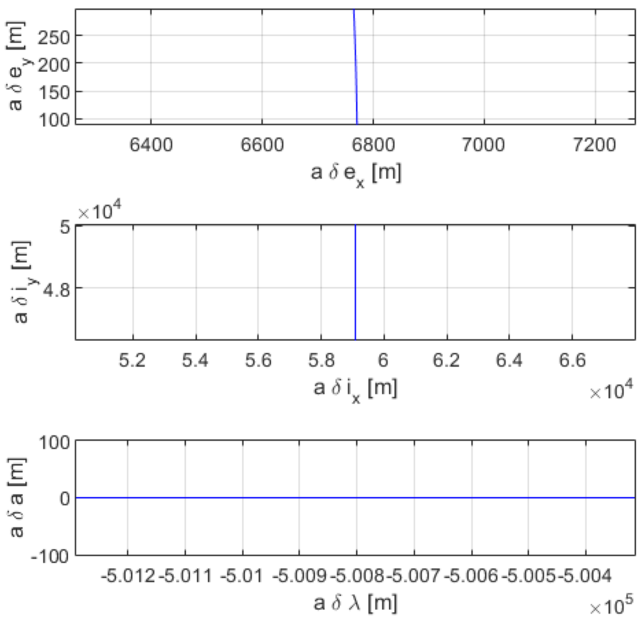


Figure 4.14: Evolution of Elements with STM - 1st set of rQNSOE

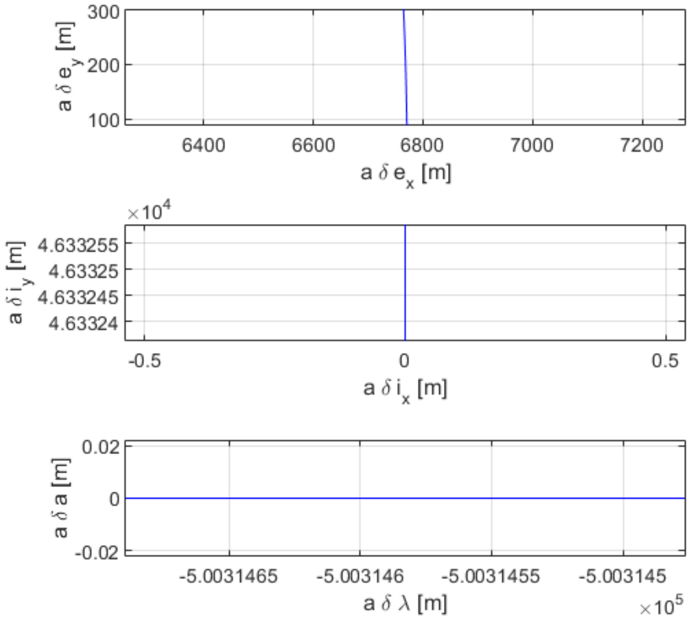


Figure 4.15: Evolution of Elements with STM - 2nd set of rQNSOE

We observe that the results are consistent across both cases, as demonstrated in earlier sections. Unlike the osculating propagation, the STM-based analytical propagation does not exhibit short-period oscillations. Instead, it captures only the secular evolution of the relative orbital elements by propagating their mean values over time.

## 5 Problem Set 5

The code used for this assignment is available in our GitHub repository [1] and the Appendix. See "Problem Set 5 Code" for code for each section. Additionally, see "Propagators", "Control Helpers" and "Utils" sections.

### 5.1 What are the control objectives?

#### 5.1.1 Operational Modes

The Dragon spacecraft operates in the following key modes during approach and docking:

- **Rendezvous Mode:** Guides the capsule to the vicinity of the target using coarser navigation and phasing maneuvers.
- **Station-Keeping Mode:** Holds the capsule's relative position near the target while preparing for final docking, enabling transition to high-precision control.
- **Docking Mode:** Executes the final approach and docking sequence using fine attitude and translational control for secure connection to the target.

#### 5.1.2 Mode Definitions

The rendezvous begins with the following initial and final Quasi Non-Singular relative orbital elements (QNSROEs):

$$\delta\alpha_{\text{initial}} = \begin{bmatrix} \delta a \\ \delta\lambda \\ \delta e_x \\ \delta e_y \\ \delta i_x \\ \delta i_y \end{bmatrix} = \begin{bmatrix} 0 \\ -3.3773^\circ \\ 0.0007 \\ 0.0007 \\ 0.4989^\circ \\ 0.7850^\circ \end{bmatrix}, \quad \delta\alpha_{\text{final}} = \begin{bmatrix} 0 \\ -0.04997^\circ \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}$$

**Station-Keeping Mode:**

This mode maintains the QNSROEs achieved at the end of rendezvous:

$$\delta\alpha_{\text{initial}} = \delta\alpha_{\text{final}} = \begin{bmatrix} 0 \\ -0.04997^\circ \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}$$

**Docking Mode:**

This final phase eliminates the residual mean longitude error:

$$\delta\alpha_{\text{initial}} = \begin{bmatrix} 0 \\ -0.04997^\circ \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}, \quad \delta\alpha_{\text{final}} = \begin{bmatrix} 0 \\ 0^\circ \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}$$

### 5.1.3 Control Requirements by Mode

**Rendezvous Mode:** Coarse control is sufficient during this phase, with allowable separations on the order of kilometers to hundreds of meters. Navigation errors within tens of meters are tolerable. Actuation can involve larger burns but must respect fuel efficiency and safety corridors. Collision avoidance constraints apply beyond 250 m from the target.

**Station-Keeping Mode:** Requires tighter position and velocity control, typically maintaining separation within a few meters. Control precision must be sufficient to counteract environmental perturbations (such as J2 and drag). Actuation is limited to small corrections with minimal fuel expenditure. Knowledge accuracy should be sub-meter, especially in relative navigation.

**Docking Mode:** Demands the highest control fidelity. Separation tolerance tightens to centimeters or less. Relative position knowledge must be accurate to within a few millimeters, and actuation must be extremely precise and smooth to prevent impact forces or misalignment. Constraints on burns are strict and typically use low-thrust, high-precision systems like RCS jets or magnetic docking alignment.

### 5.1.4 Reconfiguration Control Requirements

| **Transition** | **Time Scale** | **Control Constraints** | **Safety Concerns** |
|---|---|---|---|
| Rendezvous $\rightarrow$ Station-Keeping | Orbit-scale (hours) | Efficient burns, minimal $\Delta V$, alignment of orbital elements | Avoiding keep-out zones, predictable relative motion |
| Station-Keeping $\rightarrow$ Docking | Minutes to hours | High-precision sensing and control activation | Abort capability, contact safety margins |
| Docking $\rightarrow$ Docked/Idle | Immediate | Final approach alignment | Secure latching, eliminate relative motion |

Table 5.1: Control requirements during mode transitions

### 5.1.5 Actuation Considerations

We plan to use low-thrust, high-specific-impulse actuators for the rendezvous and station-keeping phases. These are likely electric or cold-gas micro-thrusters that offer fine, continuous control with minimal propellant mass. High-thrust, low-Isp chemical engines (such as Draco) are available for major maneuvers or aborts, but all nominal burns use the low-thrust, high-efficiency system.

### 5.1.6 Orbit Dynamics Modeling Approach

We model relative motion using a linear time-varying STM in ROE space that explicitly captures the dominant J2 effects, rather than using a pure two-body propagator. In particular, we use the state transition matrix derived by Chernick and D'Amico (based on Koenig et al.'s approach), which propagates mean relative orbital elements under two-body plus J2 perturbations [4]. This provides a compact, analytically tractable model that retains secular and long-period drift from Earth's oblateness without requiring full nonlinear integration.

- **Linear ROE Dynamics with J2:** The STM, $\Phi(t \mid t_0)$, advances the mean ROE vector $\delta\alpha$ linearly:

  $$\delta\alpha(t) = \Phi(t \mid t_0)\, \delta\alpha_0$$

  and includes first-order J2 secular and long-period terms directly in its time-varying coefficients.
- **ROE-Based Representation:** The rendezvous objective is framed as driving $\delta\alpha \to 0$, which makes guidance design and constraint enforcement (such as separation, alignment, and drift rate) straightforward. The STM provides a direct map between the initial and final ROEs.
- **Perturbation Inclusion vs. Neglect:** By incorporating J2 into the STM, we account for the dominant non-Keplerian effect in LEO. Other perturbations, such as drag and third-body forces, are an order of magnitude smaller over the short rendezvous timescale and can be safely neglected.

## 5.2 Impulsive Control Overview

See "Problem Set 5 Impulsive Control Code" section in Appendix for code or our GitHub repository [1].

### 5.2.1 Setup

We are simulating the following rendezvous control problem. This represents the maneuver of our capsule to the ISS, as discussed previously.

$$\alpha_{\text{chief}} = \begin{bmatrix} a \\ \lambda \\ e_x \\ e_y \\ i_x \\ i_y \end{bmatrix} = \begin{bmatrix} 6{,}771{,}000 \\ 234.4657^\circ \\ 0.0004 \\ 0.0004 \\ 51.6400^\circ \\ 201.5206^\circ \end{bmatrix}$$

$$\delta\alpha_{\text{initial}} = \begin{bmatrix} \delta a \\ \delta\lambda \\ \delta e_x \\ \delta e_y \\ \delta i_x \\ \delta i_y \end{bmatrix} = \begin{bmatrix} 0 \\ -3.3773^\circ \\ 0.0007 \\ 0.0007 \\ 0.4989^\circ \\ 0.7850^\circ \end{bmatrix}$$

$$\delta\alpha_{\text{final}} = \begin{bmatrix} \delta a \\ \delta\lambda \\ \delta e_x \\ \delta e_y \\ \delta i_x \\ \delta i_y \end{bmatrix} = \begin{bmatrix} 0 \\ -0.04997^\circ \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}$$

### 5.2.2 Delta V Constraint

To estimate the maximum achievable $\Delta v$ for impulsive maneuvers, we use the ideal rocket equation:

$$\Delta v = I_{sp} \cdot g_0 \cdot \ln\left(\frac{m_0}{m_f}\right)$$

where $I_{sp}$ is the specific impulse in seconds, $g_0 = 9.80665\,\text{m/s}^2$ is standard gravity, $m_0$ is the initial mass (vehicle + propellant), and $m_f = m_0 - m_p$ is the final mass after expelling $m_p$ kg of propellant.

Assuming a total mass of $m_0$ = 12,000kg, a maximum propellant load of $m_p$ = 30kg for a single burn, and a conservative vacuum specific impulse of $I_{sp}$ = 300s (representative of Draco thrusters on Dragon), we compute:

$$\Delta v_{max} = I_{sp}\, g_0\, \ln\left(\frac{m_0}{m_0 - m_p}\right) = 300 \times 9.80665 \times \ln\left(\frac{12{,}000}{12{,}000 - 30}\right) \approx \boxed{7.36 \text{ m/s}}$$

Thus, the system can achieve approximately 7.36m/s of $\Delta v$ for a burn under these assumptions.

### 5.2.3 Reachable Set Generation

We follow the closed-form, reachable-set approach of Chernick and D'Amico [5], specialized to our quasiânonsingular ROE state. Given a chief orbit $\alpha_c$ and a maximum impulsive $\Delta v$ budget $\Delta v_{\text{max}}$, we define:

$$N_\nu, N_\phi, N_\theta \text{ sample counts}, \quad \Delta v_{\text{max}}.$$

**1. Sample True Anomalies** Discretize the true anomaly over one orbit:

$$\nu_i = 2\pi\,\frac{i-1}{N_\nu}, \quad i = 1, \dots, N_\nu.$$

At each $\nu_i$, compute the $6 \times 3$ control-input matrix

$$\Gamma(\nu_i) \;=\; \frac{1}{na} \begin{bmatrix} \frac{2}{\eta} e \sin\nu_i & \frac{2}{\eta}(1 + e\cos\nu_i) & 0 \\ -2\eta^2/(1 + e\cos\nu_i) & 0 & 0 \\ \eta \sin\nu_i & \eta\big(e + \frac{(2+e\cos\nu_i)\cos\nu_i}{1+e\cos\nu_i}\big) & 0 \\ -\frac{\eta}{e}\cos\nu_i & \frac{\eta}{e}\frac{(2+e\cos\nu_i)\sin\nu_i}{1+e\cos\nu_i} & 0 \\ 0 & 0 & \eta\frac{\cos(\nu_i+\omega)}{1+e\cos\nu_i} \\ 0 & 0 & \eta\frac{\sin(\nu_i+\omega)}{1+e\cos\nu_i} \end{bmatrix},$$

**2. Sample Impulse Directions** For each inâplane ROE plane ($\{\delta a, \delta\lambda\}$ and $\{\delta e_x, \delta e_y\}$), sample

$$u_{ij} = dv_{\max} \begin{bmatrix} \cos\phi_j \\ \sin\phi_j \end{bmatrix}, \quad \phi_j = 2\pi \frac{j-1}{N_\phi},\ j = 1, \dots, N_\phi.$$

For the outâofâplane $\{\delta i_x, \delta i_y\}$ plane, sample

$$u_{ik} = dv_{\max} \cos\theta_k, \quad \theta_k = \pi \frac{k-1}{N_{th}},\ k = 1, \dots, N_{th}.$$

**3. Build and Scale Reachable Samples** At each true anomaly $\nu_i$ and for each direction vector $u$, compute the nondimensional ROE increment

$$\Delta\alpha = \Gamma(\nu_i)\, u,$$

but use only columns 1-2 of $\Gamma(\nu_i)$ for the two inâplane planes and only column 3 for the inclination plane. Then scale to meters by multiplying by the chief semiâmajor axis $a$:

$$\mathbf{p} = a\, \Delta\alpha_{\text{plane}}.$$

This yields a cloud of 2D points $\{\mathbf{p}\}$ in each ROE plane.

**4. Convex-Hull** Then compute the 2D convex hull of $\{\mathbf{p}_{ij}\}$ via convhull($\mathbf{p}_{ij}$). Plot the raw samples and overlay the hull polygon.

**5. Plotting** Finally, for each of the three planes:

$$(a\Delta\delta a,\ a\Delta\delta\lambda), \quad (a\Delta\delta e_x,\ a\Delta\delta e_y), \quad (a\Delta\delta i_x,\ a\Delta\delta i_y),$$

plot $\{\mathbf{p}_{ij}\}$ as dots and the resulting convex-hull boundary as a solid line. Additionally, in the next figure, we plot the required change in QNSROEs on the same planes as the reachable sets, yielding Figures below.

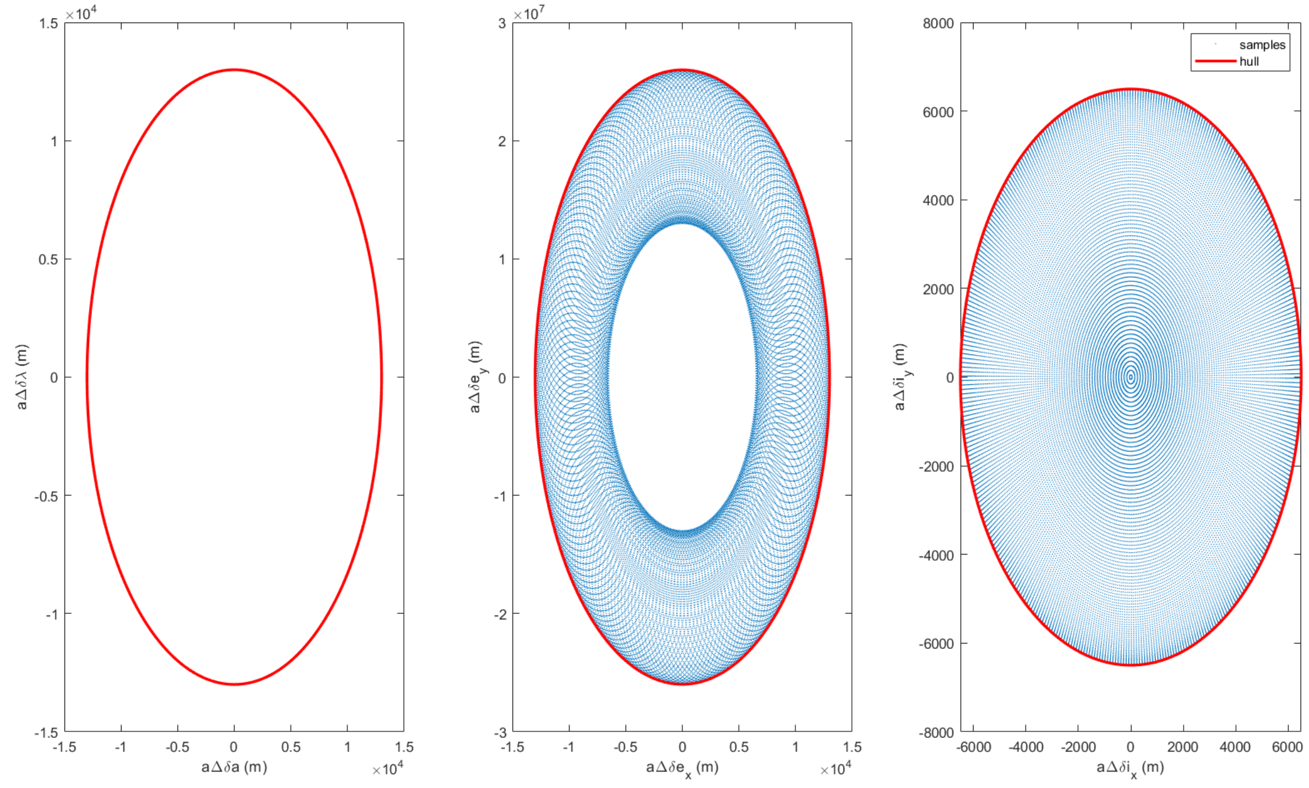


Figure 5.1: Reachable sets in the 3 ΔROE planes for $dv_{\max} = 7.36$ m/s.

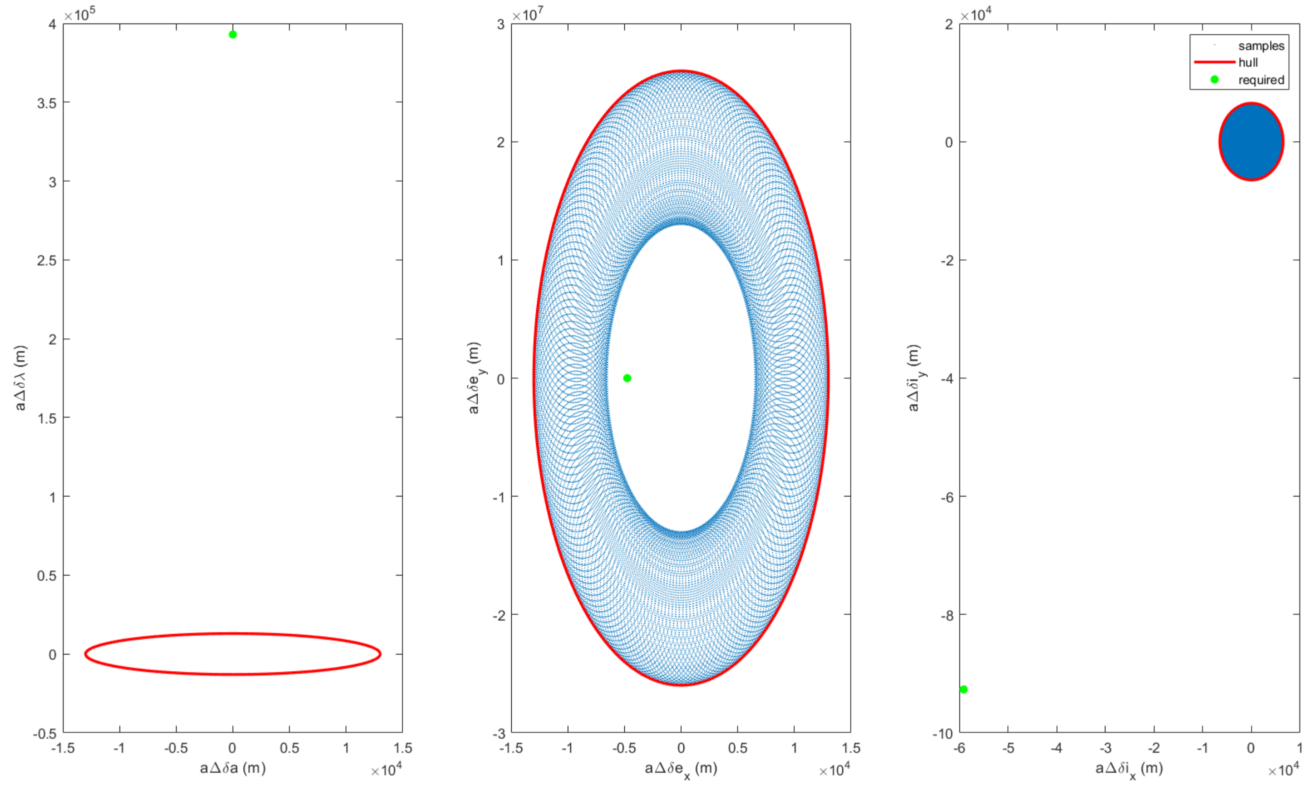


Figure 5.2: Reachable set in the 3 ΔROE planes with green dot indicating desired ΔROE.

From Figure 7.2, we observe that the maximum impulsive $\Delta v$ is not even close to reaching the required QNSROE change directly. This indicates that a single impulse is inadequate, and several impulsive burns will be necessary to achieve the desired state, as addressed in the following sections.

### 5.2.4 Minimum $\Delta v$ Calculation

**Method 1: Plane-by-Plane Optimization** We decompose the sixâdimensional QNSROE change $\Delta\mathbf{s} = \mathbf{s}_F - \mathbf{s}_0$ into three 2D planes: $\{\delta a, \delta\lambda\}$, $\{\delta e_x, \delta e_y\}$, and $\{\delta i_x, \delta i_y\}$. For each plane $p \in \{1, 2, 3\}$ with row indices $r_p$, we define

$$d_p = \Delta\mathbf{s}_{r_p}, \quad \Gamma_{r_p}(\nu) = \text{rows } r_p \text{ of } \Gamma(\nu).$$

At each true anomaly $\nu_k = 2\pi(k-1)/N$, we solve

$$v_p(\nu_k) = \begin{cases} \left(\Gamma_{r_p}(\nu_k)_{:,1:2}\right) d_p, & p = 1, 2, \\ \left(\Gamma_{r_p}(\nu_k)_{:,3}\right) d_p, & p = 3, \end{cases}$$

and compute its norm $\|v_p(\nu_k)\|$. The minimum impulse for plane $p$ is

$$\Delta v_p = \min_k \|v_p(\nu_k)\|, \qquad \nu_p^* = \arg\min_k \|v_p(\nu_k)\|.$$

Finally, the overall required $\Delta v$ is the largest of the three:

$$\Delta v_{\text{plane}} = \max_p \Delta v_p, \qquad \nu^* = \nu^*_{p^*},$$

where $p^* = \arg\max_p \Delta v_p$.

This method produced $\boxed{\Delta v_{min} = 222.518 \text{ m/s at } \nu = 179.1^\circ}$

**Method 2: Global 6D Optimization** We avoid plane-by-plane splitting by using all six rows of $\Gamma(\nu)$ at once. For each $\nu_k$ we compute the leastânorm solution

$$v(\nu_k) = \big(\Gamma(\nu_k)\big)\,\Delta \mathbf{s},$$

and its norm $\|v(\nu_k)\|$. The true minimum $\Delta v$ and corresponding anomaly are

$$\Delta v_{\text{opt}} = \min_k \|v(\nu_k)\|, \qquad \nu_{\text{opt}} = \arg\min_k \|v(\nu_k)\|.$$

This method produced $\boxed{\Delta v_{min} = 120.187 \text{ m/s at } \nu = 354.6^\circ}$, yielding the globally optimal single-burn impulse in RTN coordinates, and showing that the Plane-by-Plane method produced an overestimate for $\Delta v_{min}$ .

### 5.2.5 Impulse Control

The goal is to transfer the deputy from QNSROE$_0$ to QNSROE$_F$ over the time period ($\tau = 12\,\text{h}$) using $M$ discrete RTN burns whose epochs

$$t_{\text{burn}}(j) = [0.025 : 0.025 : 0.975]\,\tau$$

are chosen so that the minimum-$\|\delta v\|_2$ solution stays below the calculated $\Delta v_{\max}$. The number of burns was arrived at through trial and error, where different time spacings were tried and maximum impulsive burns from the algorithm described below were observed. More burns were added to reduce the required $\Delta v$ per burn. The chosen times result in 39 burns. While high, it matches intuition given how far below are impulsive $\Delta v_{max}$ is from the minimum required total $\Delta v$

First, we build the linear map $A \in \mathbb{R}^{6\times 3M}$ by stacking each epoch's contribution:

$$A = \big[\Phi_{j\to f}\,\Gamma(\nu_j)\big]_{j=1}^{M},$$

where

- $\Phi_{0\to j} = \Phi_{J2}(\alpha_c, \mu, J_2, R_e, t_{\text{burn}}(j) - t_0)$ $\Phi_{j\to f} = \Phi_{J2}(\alpha_c, \mu, J_2, R_e, t_f - t_{\text{burn}}(j))$ are the J$_2$-perturbed STMs in QNSROE space from [5]
- $\nu_j$ is the true anomaly at $t_{\text{burn}}(j)$, obtained by first propagating the Keplerian anomaly and then adding the secular J$_2$ perigee drift $\dot{\omega}\,\Delta t$;
- $\Gamma(\nu_j) \in \mathbb{R}^{6\times 3}$ is the Gauss-variational control matrix evaluated at $\nu_j$ .

$\Phi_{J2}(\tau) =$

$$\begin{bmatrix}
1 & 0 & 0 & 0 & 0 & 0 \\
-7\kappa\eta P\,\tau - \frac{3}{2}n\tau & 1 & \frac{7\kappa e\cos\omega\,P\,\tau}{\eta} & \frac{7\kappa e\sin\omega\,P\,\tau}{\eta} & -7\kappa\eta S\,\tau & 0 \\
3.5\kappa\sin(\omega+\dot\omega\tau)\,Q\,\tau & 0 & \cos(\dot\omega\tau) - 4\kappa e\cos\omega\sin(\omega+\dot\omega\tau)\frac{Q}{\eta^2}\tau & -\sin(\dot\omega\tau) - 4\kappa e\sin\omega\sin(\omega+\dot\omega\tau)\frac{Q}{\eta^2}\tau & 5\kappa\sin(\omega+\dot\omega\tau)\,S\,\tau & 0 \\
-3.5\kappa\cos(\omega+\dot\omega\tau)\,Q\,\tau & 0 & \sin(\dot\omega\tau) + 4\kappa e\cos\omega\cos(\omega+\dot\omega\tau)\frac{Q}{\eta^2}\tau & \cos(\dot\omega\tau) + 4\kappa e\sin\omega\cos(\omega+\dot\omega\tau)\frac{Q}{\eta^2}\tau & -5\kappa\cos(\omega+\dot\omega\tau)\,S\,\tau & 0 \\
0 & 0 & 0 & 0 & 1 & 0 \\
3.5\kappa S\,\tau & 0 & -4\kappa e\cos\omega\,S\frac{\tau}{\eta^2} & -4\kappa e\sin\omega\,S\frac{\tau}{\eta^2} & 2\kappa\,T\,\tau & 1
\end{bmatrix}$$

$$n = \sqrt{\frac{\mu}{a^3}}, \quad \eta = \sqrt{1-e^2}, \quad \kappa = \frac{3J_2R_e^2\sqrt{\mu}}{4a^{7/2}\eta^4},$$

$$P = 3\cos^2 i - 1, \quad Q = 5\cos^2 i - 1, \quad S = \sin(2i), \quad T = \sin^2 i, \quad \dot\omega = \kappa Q.$$

Next, we form the pseudostate

$$b \;=\; \alpha_F \;-\; \Phi_{0\to f}\,\alpha_0, \quad \Phi_{0\to f} = \Phi_{J2}(\alpha_c, \mu, J_2, R_e, t_f - t_0).$$

Next, we solve for the minimal-$\ell_2$ $\Delta v$ stack

$$\delta v_{\text{stack}} \;=\; A^{+}\,b,$$

where $A^+$ is the Moore-Penrose pseudoinverse.

Finally, we extract each 3-vector burn $\delta v_j$ from $\Delta v_{\text{stack}}$, then compute

$$t_* = \max_j \|\delta v_j\|_2, \qquad \Delta v_{\text{tot}} = \sum_j \|\delta v_j\|_2.$$

The output of this is a series or normal, tangential, and radial impulsive burns, listed in the table below:

| Burn No | Time (h) | $\Delta v_R$ | $\Delta v_T$ | $\Delta v_N$ |
|---|---|---|---|---|
| 1 | 0.30 | 0.02 | -0.45 | -0.12 |
| 2 | 0.60 | 0.02 | -0.46 | 6.12 |
| 3 | 0.90 | 0.01 | -0.46 | 4.29 |
| 4 | 1.20 | -0.01 | -0.41 | -3.19 |
| 5 | 1.50 | -0.01 | -0.35 | -6.46 |
| 6 | 1.80 | 0.01 | -0.32 | -1.19 |
| 7 | 2.10 | 0.02 | -0.33 | 5.64 |
| 8 | 2.40 | 0.01 | -0.33 | 5.03 |
| 9 | 2.70 | -0.01 | -0.29 | -2.21 |
| 10 | 3.00 | -0.01 | -0.23 | -6.53 |
| 11 | 3.30 | 0.01 | -0.20 | -2.22 |
| 12 | 3.60 | 0.02 | -0.20 | 5.02 |
| 13 | 3.90 | 0.01 | -0.20 | 5.63 |
| 14 | 4.20 | -0.01 | -0.16 | -1.18 |
| 15 | 4.50 | -0.00 | -0.11 | -6.42 |
| 16 | 4.80 | 0.01 | -0.08 | -3.18 |
| 17 | 5.10 | 0.02 | -0.08 | 4.26 |
| 18 | 5.40 | 0.01 | -0.07 | 6.08 |
| 19 | 5.70 | -0.01 | -0.03 | -0.12 |
| 20 | 6.00 | -0.00 | 0.02 | -6.15 |
| 21 | 6.30 | 0.01 | 0.04 | -4.05 |
| 22 | 6.60 | 0.01 | 0.05 | 3.39 |
| 23 | 6.90 | 0.01 | 0.06 | 6.36 |
| 24 | 7.20 | -0.00 | 0.10 | 0.94 |
| 25 | 7.50 | -0.00 | 0.14 | -5.71 |
| 26 | 7.80 | 0.01 | 0.16 | -4.81 |
| 27 | 8.10 | 0.01 | 0.17 | 2.44 |
| 28 | 8.40 | 0.00 | 0.19 | 6.47 |
| 29 | 8.70 | -0.00 | 0.22 | 1.97 |
| 30 | 9.00 | 0.00 | 0.27 | -5.12 |
| 31 | 9.30 | 0.01 | 0.29 | -5.44 |
| 32 | 9.60 | 0.01 | 0.29 | 1.42 |
| 33 | 9.90 | 0.00 | 0.31 | 6.41 |
| 34 | 10.20 | -0.00 | 0.35 | 2.95 |
| 35 | 10.50 | 0.00 | 0.39 | -4.40 |
| 36 | 10.80 | 0.01 | 0.41 | -5.92 |
| 37 | 11.10 | 0.01 | 0.41 | 0.37 |
| 38 | 11.40 | 0.00 | 0.44 | 6.17 |
| 39 | 11.70 | -0.01 | 0.48 | 3.83 |

The largest $\Delta v$ is 6.53 m/s at burn 10, which stays below are impulsive $\Delta v_{max}$ of 7.36 m/s

The total impulse is computed as

$$\text{Total}\,\Delta v = \sum_{j=1}^{M} \sqrt{(\Delta v_{R,j})^2 + (\Delta v_{T,j})^2 + (\Delta v_{N,j})^2} = \boxed{160.116 \text{ m/s}}.$$

Our multi-burn solution yields

$$\text{Total}\,\Delta v = 160.116 \text{ m/s},$$

which lies between the plane-by-plane estimate $\Delta v_{\text{plane}} = 222.518$ m/s and the true single-burn global minimum $\Delta v_{\text{opt}} = 120.187$ m/s reported in the literature. Thus, while our discretized sequence outperforms the naive per-plane splitting, it still exceeds the global theoretical minimum by about 40 m/s.

The $\Delta v$'s are plotted vs time in the figure below as well.

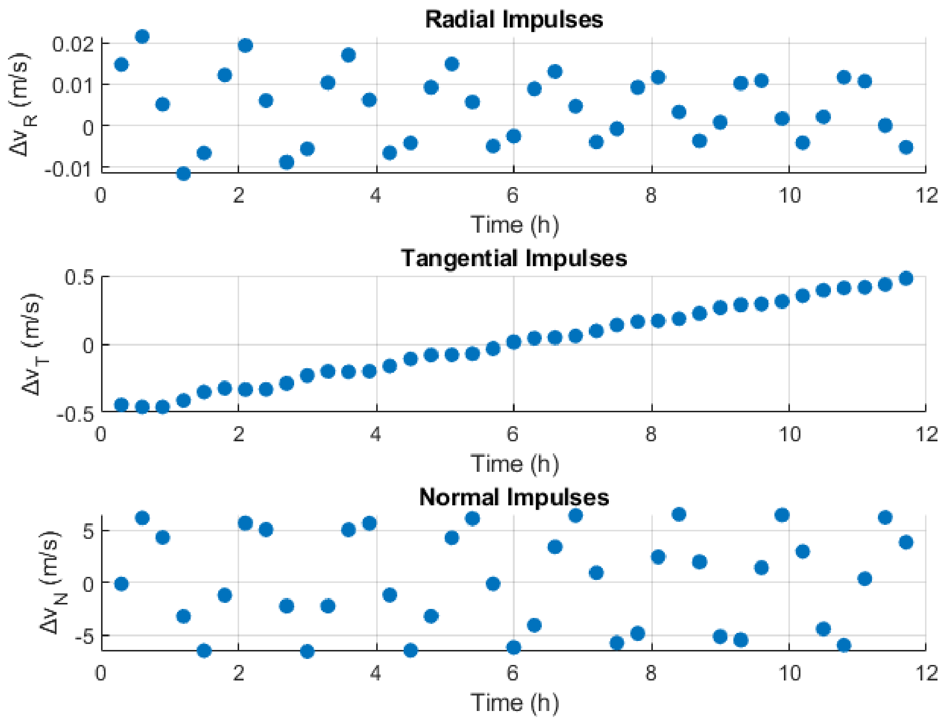


Figure 5.3: Impulsive $\Delta v$'s over Time

### 5.2.6 Ground Truth Simulation

Starting from the initial deputy and chief ROEs, we run a full nonlinear ECI-RTN ODE simulation with the computed burn sequence:

- **Nominal case:** Apply each impulse $\delta v_j$ exactly at $t_{\text{burn}}(j)$ by 1) coasting under `nonlinear_state_dot` from the previous epoch to $t_{\text{burn}}(j)$, 2) adding $\delta v_j$ to the RTN velocity components, 3) repeating until the final coast to $\tau = 12\,\text{h}$. The resulting ROE history $Q_{\text{nominal}}(t)$ is recorded.
- **Perturbed case:** Repeat the above but replace each burn with a noisy impulse

$$\delta v_j^{\text{err}} = \delta v_j\,(1 + \varepsilon_j), \quad \varepsilon_j \sim \mathcal{N}(0,\, 0.05).$$

  This models a 5% relative control error. The perturbed ROE history $Q_{\text{err}}(t)$ is likewise recorded.

Finally, we plot each QNSROE component versus time for both cases (solid blue = nominal, dashed red = perturbed) to assess deviation under realistic control uncertainty.

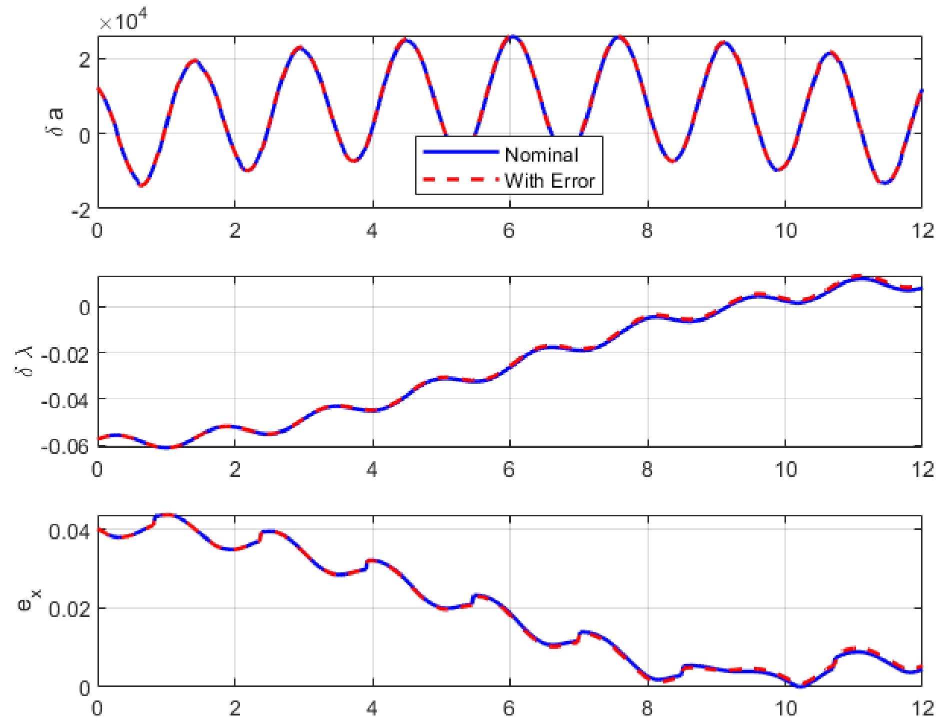


Figure 5.4: Ground Truth Simulation: QNSROE's over Time with Applied Impulses, with and without Control Error (1)

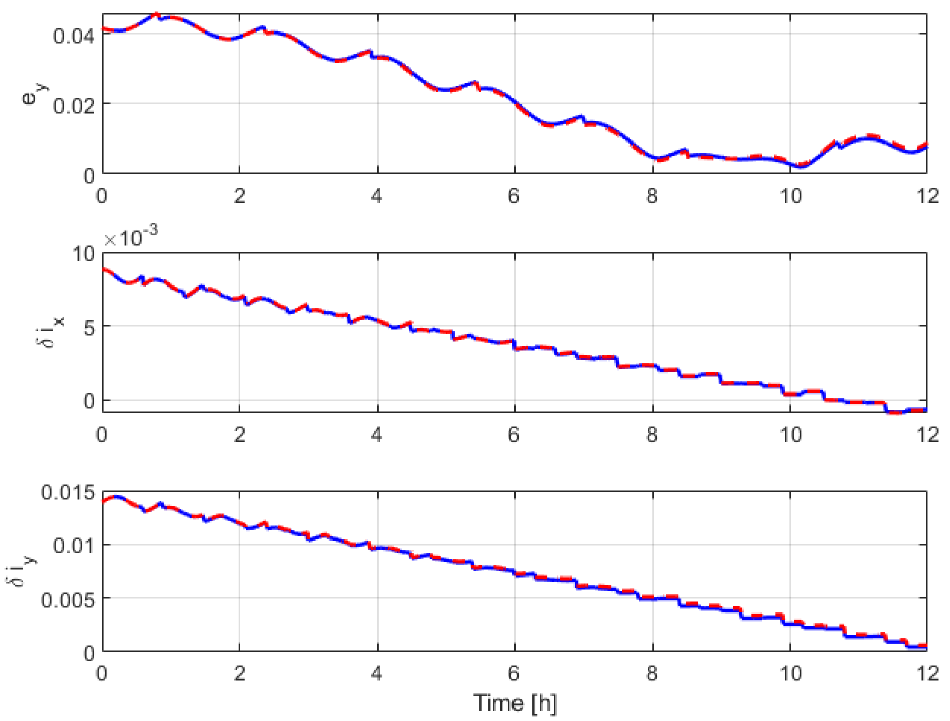


Figure 5.5: Ground Truth Simulation: QNSROE's over Time with Applied Impulses, with and without Control Error (2)

The ground-truth simulation shows that by $t = 12$ h every QNSROE component - $\delta a$, $\delta\lambda$, $e_x$, $e_y$, $\delta i_x$, $\delta i_y$ - converges to its commanded final value, validating the burn schedule. The characteristic sinusoidal ”wobbles” between burns arise from true Keplerian short-period and $J_2$ effects that lie outside the mean-ROE STM, as expected for a linearized model. The small residual offset at the final epoch confirms that the first-order $J_2$ STM remains accurate over the transfer, while any remaining bias reflects the discrete impulse approximation and neglected higher-order perturbations. Furthermore, adding 5 % random burn error produces negligible deviation from the nominal trajectory and nearly identical final ROE, demonstrating strong robustness to control inaccuracies. Finally, the almost perfect overlap of nominal and perturbed traces indicates a healthy design marginâeither allowing reduction in burn density or Îv budget without loss of performance, or tightening final residuals via additional STM fidelity or burns.

### 5.2.7 Performance and Results

#### Strengths of Implementation

- Modular design: guidance (linear STM in ROE space) is cleanly separated from high-fidelity simulation (nonlinear ODE with J2), easing maintenance and model upgrades.
- Analytic impulse mapping: use of the closed-form control matrix $\Gamma(\nu)$ avoids numerical derivative approximations, yielding exact sensitivities and fast solves.

- Built-in robustness check: injecting 5% Gaussian burn errors and re-simulating demonstrates resilience of the plan to realistic control uncertainties.

## Weaknesses of Implementation

- Open-loop guidance: all burns are preplanned; no feedback or replanning is used, so execution errors accumulate until final epoch.
- High burn count: 39 impulses were required, which is operationally heavy and increases cumulative attitude and timing complexity.
- Model simplifications: guidance STM omits drag, higher-order geopotential, third-body gravity and SRP, so some secular and short-period effects are unmodeled.
- Automated constraint enforcement: The pinv-based least-squares solver does not explicitly impose thruster limits, safety keep-out zones, or propellant budgets. Instead, we enforce the per-burn limit $\Delta v_{\max} = 7.36$ m/s implicitly by choosing the number of burns $M$ such that, after solving

  $$\delta v_{\text{stack}} = A^{+} b,$$

  the largest burn magnitude

  $$t_{\star} = \max_{j} \|\delta v_j\|_2$$

  satisfies

  $$t_{\star} \leq \Delta v_{\max}.$$

  Adjusting $M$ in this way guarantees each impulse remains within the thruster capability.

## Performance Compared to Expectations

- Total $\Delta v$: our multi-burn solution achieves

  $$\Delta v_{\text{tot}} = 160.116 \text{ m/s},$$

  which lies between the plane-by-plane estimate of 222.518 m/s and the single-burn global optimum of 120.187 m/s.
- Overshoot vs. optimum: about 33% above the theoretical lower bound; undercut the naive plane-by-plane by 28%.
- Largest single burn: 6.53 m/s (burn 10), safely under the 7.36 m/s maximum.
- Ground-truth final error: all QNSROE components converge within machine precision of their targets (residuals $< 10^{-6}$ in normalized units), confirming plan accuracy.

## Failure Modes and Next Steps

- Add feedback or receding-horizon replanning: re-estimate ROEs after each burn and update remaining burns to correct execution errors.
- Reduce burn count: explore multi-burn optimization (direct transcription or convex programming) to achieve the same $\Delta\alpha$ with fewer impulses.

- Enhance STM fidelity: incorporate drag, higher-order geopotential and third-body terms into the guidance model to reduce unmodeled drift.
- Enforce constraints: move from unconstrained least-squares to a constrained solver that respects thruster saturation, propellant limits and safety keep-out zones.

## 6 Problem Set 6

The code used for this assignment is available in our GitHub repository [1] and the Appendix. See "Problem Set 6 Code" for code for each section. Additionally, see "Propagators", "Control Helpers" and "Utils" sections.

### 6.1 Setup

The goal is to design a control law that drives the deputy satellite's QNSROE to a desired state, accounting for $J_2$ perturbations. We choose a Lyapunov-based control approach without constraints.

#### 6.1.1 Mathematical Foundations

We model the deputy satellite's motion relative to the chief using a 6-state quasi-nonsingular relative orbital elements (QNSROE) vector, which is well-suited for near-circular orbits.

The QNSROE state is defined as:

$$\delta\overline{\boldsymbol{\alpha}} = \begin{bmatrix} \delta a_d \\ \delta\lambda_d \\ \delta e_{x,d} \\ \delta e_{y,d} \\ \delta i_{x,d} \\ \delta i_{y,d} \end{bmatrix},$$

where each component is defined as follows:

- $\delta a_d = \dfrac{a_d - a_c}{a_c}$: Relative semi-major axis, normalized by the chief's semi-major axis $a_c$ (dimensionless).
- $\delta\lambda_d = M_d + \omega_d + \Omega_d \cos i_c - (M_c + \omega_c + \Omega_c \cos i_c)$: Relative mean longitude, combining mean anomaly $M$, argument of perigee $\omega$, and right ascension of the ascending node $\Omega$ (radians).
- $\delta e_{x,d} = e_d \cos\omega_d - e_c \cos\omega_c$: $x$-component of the relative eccentricity vector (dimensionless).
- $\delta e_{y,d} = e_d \sin\omega_d - e_c \sin\omega_c$: $y$-component of the relative eccentricity vector (dimensionless).
- $\delta i_{x,d} = i_d - i_c$: Difference in inclination between the deputy and the chief (radians).
- $\delta i_{y,d} = (\Omega_d - \Omega_c)\sin i_c$: $y$-component of the relative inclination vector (radians).

Here, subscripts $c$ and $d$ denote the chief and deputy satellites, respectively.

### 6.1.2 Numerical Values

The chief's OEs and deputy's QNSROEs are the same as PS5, defined below.

**Chief's Initial Orbital Elements:**

$$\boldsymbol{\alpha}_c = \begin{bmatrix} a_c \\ e_c \\ i_c \\ \Omega_c \\ \omega_c \\ \nu_c \end{bmatrix} = \begin{bmatrix} 6771 \times 10^3 \text{ m} \\ 5.0 \times 10^{-4} \\ 51.64^\circ \\ 257^\circ \\ 45^\circ \\ 30^\circ \end{bmatrix}$$

**Deputy's Initial Quasi-Non-Singular Relative Orbital Elements:**

$$\delta\boldsymbol{\alpha}_{\text{initial}} = \begin{bmatrix} \delta a_d \\ \delta\lambda_d \\ \delta e_{x,d} \\ \delta e_{y,d} \\ \delta i_{x,d} \\ \delta i_{y,d} \end{bmatrix} = \begin{bmatrix} 0 \\ -3.3773^\circ \\ 0.0007 \\ 0.0007 \\ 0.4989^\circ \\ 0.7850^\circ \end{bmatrix}$$

**Deputy's Final Quasi-Non-Singular Relative Orbital Elements:**

$$\delta\boldsymbol{\alpha}_{\text{desired}} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}$$

**Duration**

We will apply a continuous control law for 20 orbits.

## 6.2 Continuous Control without Constraints

### 6.2.1 Mathematical Foundations

In this section, we describe the mathematical models we used to implement the continuous control law for our satellite formation-flying mission. Inspiration from [6].

#### Dynamics Model

The dynamics of the QNSROE state follow a linear time-varying system:

$$\delta\dot{\overline{\boldsymbol{\alpha}}}(t) = \boldsymbol{A}_c(t)\delta\overline{\boldsymbol{\alpha}}(t) + \boldsymbol{B}_c(t)\boldsymbol{u}(t),$$

where:

- $\boldsymbol{A}_c(t) \in \mathbb{R}^{6\times 6}$: The plant matrix, which captures how the ROE evolve due to orbital dynamics and $J_2$ perturbations.
- $\boldsymbol{B}_c(t) \in \mathbb{R}^{6\times 3}$: The control input matrix, which maps our control actions to changes in the ROE.
- $\boldsymbol{u}(t) = \begin{bmatrix}\Delta V_r \\ \Delta V_t \\ \Delta V_n\end{bmatrix}$: The control input vector, representing thrust in the radial, tangential, and normal directions (in m/s).

The plant matrix $\boldsymbol{A}_c(t)$ describes the natural evolution of QNSROE, including Earth's oblateness ($J_2$) effects. Its direct computation is complex due to the chief satellite's time-varying orbit under $J_2$.

Instead, we approximate $\boldsymbol{A}_c(t)$ using the state transition matrix (STM), $\Phi_{J2,qns}(\tau, t) \in \mathbb{R}^{6\times 6}$, which propagates the state over a time step $\tau$:

$$\delta\overline{\boldsymbol{\alpha}}(t+\tau) = \Phi_{J2,qns}(\tau, t)\delta\overline{\boldsymbol{\alpha}}(t). \tag{26}$$

The STM satisfies $\Phi_{J2,qns}(0, t) = \boldsymbol{I}_6$.

We conisder the unforced system, $\delta\dot{\overline{\boldsymbol{\alpha}}}(s) = \boldsymbol{A}_c(s)\delta\overline{\boldsymbol{\alpha}}(s)$, with $s = t + \tau$. Differentiating $\delta\overline{\boldsymbol{\alpha}}(t+\tau) = \Phi_{J2,qns}(\tau, t)\delta\overline{\boldsymbol{\alpha}}(t)$ with respect to $\tau$:

$$\frac{d}{d\tau}\delta\overline{\boldsymbol{\alpha}}(t+\tau) = \boldsymbol{A}_c(t+\tau)\Phi_{J2,qns}(\tau, t)\delta\overline{\boldsymbol{\alpha}}(t) = \frac{d}{d\tau}\Phi_{J2,qns}(\tau, t)\delta\overline{\boldsymbol{\alpha}}(t). \tag{27}$$

Thus:

$$\frac{d}{d\tau}\Phi_{J2,qns}(\tau, t) = \boldsymbol{A}_c(t+\tau)\Phi_{J2,qns}(\tau, t). \tag{28}$$

For a small $\tau = 10^{-3}$ s, the dynamics are nearly linear, so:

$$\Phi_{J2,qns}(\tau, t) \approx \boldsymbol{I}_6 + \tau\boldsymbol{A}_c(t). \tag{29}$$

Rearranging:

$$\boldsymbol{A}_c(t) \approx \frac{\Phi_{J2,qns}(\tau,t) - \boldsymbol{I}_6}{\tau}. \tag{30}$$

This approximation uses $\Phi_{J2,qns}(\tau,t)$ - computed numerically or semi-analytically - to estimate $\boldsymbol{A}_c(t)$ and captures $J_2$ effects efficiently.

The STM $\Phi_{J2,qns}(\tau,t)$ is computed using the chief's orbital elements at time $t$, accounting for $J_2$ effects. It has the form:

$$\Phi_{J2,qns}(\tau,t) = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ \phi_{21} & 1 & \phi_{23} & \phi_{24} & \phi_{25} & 0 \\ \phi_{31} & 0 & \phi_{33} & \phi_{34} & \phi_{35} & 0 \\ \phi_{41} & 0 & \phi_{43} & \phi_{44} & \phi_{45} & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ \phi_{61} & 0 & \phi_{63} & \phi_{64} & \phi_{65} & 1 \end{bmatrix},$$

with the non-zero elements defined as:

$$\phi_{21} = -\left(\frac{3}{2}n_c + \frac{7}{2}\kappa EP\right)\tau,$$

$$\phi_{23} = \kappa e_{xi} FGP\tau, \quad \phi_{24} = \kappa e_{yi} FGP\tau, \quad \phi_{25} = -\kappa FS\tau,$$

$$\phi_{31} = \frac{7}{2}\kappa e_{yf} Q\tau,$$

$$\phi_{33} = \cos(\omega_{\text{dot}}\tau) - 4\kappa e_{xi} e_{yf} GQ\tau,$$

$$\phi_{34} = -\sin(\omega_{\text{dot}}\tau) - 4\kappa e_{yi} e_{yf} GQ\tau,$$

$$\phi_{35} = 5\kappa e_{yf} S\tau,$$

$$\phi_{41} = -\frac{7}{2}\kappa e_{xf} Q\tau,$$

$$\phi_{43} = \sin(\omega_{\text{dot}}\tau) + 4\kappa e_{xi} e_{xf} GQ\tau,$$

$$\phi_{44} = \cos(\omega_{\text{dot}}\tau) + 4\kappa e_{yi} e_{xf} GQ\tau,$$

$$\phi_{45} = -5\kappa e_{xf} S\tau,$$

$$\phi_{61} = \frac{7}{2}\kappa S\tau,$$

$$\phi_{63} = -4\kappa e_{xi} G S\tau, \quad \phi_{64} = -4\kappa e_{yi} G S\tau,$$

$$\phi_{65} = 2\kappa T\tau.$$

To make sense of these, we define all the sub-variables used in the STM:

- $n_c = \sqrt{\frac{\mu}{a_c^3}}$: The chief's mean motion (rad/s), where $\mu = 3.986004418 \times 10^{14}\,\text{m}^3/\text{s}^2$ is Earth's gravitational parameter.
- $\eta = \sqrt{1 - e_c^2}$: A parameter related to the chief's eccentricity.
- $\kappa = \frac{3}{4}\frac{J_2 R^2 \sqrt{\mu}}{a_c^{7/2}\eta^4}$: A coefficient for $J_2$ perturbations, with $J_2 = 1.08262668 \times 10^{-3}$ (Earth's second zonal harmonic) and $R = 6378137\,\text{m}$ (Earth's equatorial radius).
- $E = 1 + \eta$: An auxiliary term.
- $F = 4 + 3\eta$: Another auxiliary term.
- $G = \frac{1}{\eta^2}$: Scales the eccentricity effects.
- $P = 3\cos^2 i_c - 1$: Inclination-dependent term.
- $Q = 5\cos^2 i_c - 1$: Another inclination-dependent term.
- $S = \sin(2i_c)$: Double-angle inclination term.
- $T = \sin^2 i_c$: Squared sine of inclination.
- $e_{xi} = e_c \cos\omega_c$, $e_{yi} = e_c \sin\omega_c$: Initial eccentricity vector components of the chief.
- $\dot{\omega} = \kappa Q$: The secular rate of change of the argument of perigee due to $J_2$ (rad/s).
- $\omega_f = \omega_c + \dot{\omega}\tau$: The argument of perigee after time $\tau$.
- $e_{xf} = e_c \cos\omega_f$, $e_{yf} = e_c \sin\omega_f$: Final eccentricity vector components.

We compute the STM at each time step using the chief's current orbital elements, which are updated by numerically propagating the chief's orbit with $J_2$ effects included.

The control input matrix $\boldsymbol{B}_c(t)$ connects our control thrusts to changes in the ROE. It tells us how much each thrust component ($\Delta V_r$, $\Delta V_t$, $\Delta V_n$) affects the ROE. The matrix is defined as:

$$
\boldsymbol{B}_c(t) = \frac{1}{a_c n_c} \begin{bmatrix} \frac{2e_c \sin f_c}{\eta} & \frac{2(1+e_c \cos f_c)}{\eta} & 0 \\ -\frac{2\eta^2}{1+e_c \cos f_c} & 0 & 0 \\ \eta \sin(\omega_c + f_c) & \eta \left( \frac{(2+e_c \cos f_c)\cos(\omega_c+f_c)+e_{xc}}{1+e_c \cos f_c} \right) & \eta e_{yc} \frac{\sin(\omega_c+f_c)}{\tan i_c (1+e_c \cos f_c)} \\ -\eta \cos(\omega_c + f_c) & \eta \left( \frac{(2+e_c \cos f_c)\sin(\omega_c+f_c)+e_{yc}}{1+e_c \cos f_c} \right) & -\eta e_{xc} \frac{\sin(\omega_c+f_c)}{\tan i_c (1+e_c \cos f_c)} \\ 0 & 0 & \eta \frac{\cos(\omega_c+f_c)}{1+e_c \cos f_c} \\ 0 & 0 & \eta \frac{\sin(\omega_c+f_c)}{1+e_c \cos f_c} \end{bmatrix},
$$

Where we've got a few more variables to define:

- $e_{xc} = e_c \cos \omega_c$, $e_{yc} = e_c \sin \omega_c$: The chief's eccentricity vector components, same as $e_{xi}$, $e_{yi}$ but relabeled for clarity.
- $f_c$: The chief's true anomaly at time $t$, obtained from the propagated orbit.
- $1 + e_c \cos f_c$: A term that adjusts for the orbit's shape at the current position.
- Other terms like $\eta$, $n_c$, $a_c$, $i_c$, and $\omega_c$ are the same as previously defined.

The factor $\dfrac{1}{a_c n_c}$ scales the matrix to match the chief's orbit size and speed, ensuring the control inputs have the correct effect on the deputy spacecraft dynamics.

**Lyapunov Function**

To guide our control law, we picked a Lyapunov function [7] that measures how far the deputy's ROE are from the desired ones. We choose:

$$V(t) = \frac{1}{2}(\delta\boldsymbol{\alpha}(t) - \delta\boldsymbol{\alpha}_{\text{desired}})^T(\delta\boldsymbol{\alpha}(t) - \delta\boldsymbol{\alpha}_{\text{desired}}), \tag{31}$$

Where $\delta\boldsymbol{\alpha}_{\text{desired}}$ are teh desired relative quasi non singular orbital elements, as specified in the setup.

To see how $V$ changes over time, we take its derivative:

$$\dot{V}(t) = (\delta\boldsymbol{\alpha}(t) - \delta\boldsymbol{\alpha}_{\text{desired}})^T \delta\dot{\boldsymbol{\alpha}}(t). \tag{32}$$

Plugging in the dynamics:

$$\delta\dot{\boldsymbol{\alpha}}(t) = \boldsymbol{A}_c(t)\delta\boldsymbol{\alpha}(t) + \boldsymbol{B}_c(t)\boldsymbol{u}(t), \tag{33}$$

we get:

$$\dot{V}(t) = (\delta\boldsymbol{\alpha}(t) - \delta\boldsymbol{\alpha}_{\text{desired}})^T \left(\boldsymbol{A}_c(t)\delta\boldsymbol{\alpha}(t) + \boldsymbol{B}_c(t)\boldsymbol{u}(t)\right). \tag{34}$$

**Lyapunov Control Law**

The control law we implement is:

$$\boldsymbol{u}(t) = -\boldsymbol{B}_c^{\dagger}(t)\left(\boldsymbol{A}_c(t)\,\delta\boldsymbol{\alpha}(t) + \boldsymbol{P}(t)\left(\delta\boldsymbol{\alpha}(t) - \delta\boldsymbol{\alpha}_{\text{desired}}\right)\right), \tag{35}$$

where $\boldsymbol{B}_c^\dagger = (\boldsymbol{B}_c^T \boldsymbol{B}_c)^{-1} \boldsymbol{B}_c^T$ is the Moore - Penrose pseudoinverse of $\boldsymbol{B}_c(t)$. This helps project our control effort effectively into the available thrust directions.

The matrix $\boldsymbol{P}(t)$ is a time-varying, diagonal feedback gain designed to prioritize fuel-efficient thrusting at specific orbital locations. It is defined as:

$$\boldsymbol{P}(t) = \frac{1}{k} \begin{bmatrix} \cos^N J & 0 & 0 & 0 & 0 & 0 \\ 0 & \cos^N K & 0 & 0 & 0 & 0 \\ 0 & 0 & \cos^N J & 0 & 0 & 0 \\ 0 & 0 & 0 & \cos^N J & 0 & 0 \\ 0 & 0 & 0 & 0 & \cos^N H & 0 \\ 0 & 0 & 0 & 0 & 0 & \cos^N H \end{bmatrix},$$

with parameters:

- $k = 1000$: A positive gain scaling factor to tune control strength.
- $N = 14$: An even exponent that shapes the sharpness of thrusting windows (larger $N$ leads to more focused thrust peaks).
- $J = \varphi - \bar{\varphi}_{\text{ip}}$: The in-plane angular offset, where $\bar{\varphi}_{\text{ip}} = \arctan 2(\delta e_{y,d}, \delta e_{x,d})$ is the optimal direction for eccentricity control.
- $K = \varphi - \bar{\varphi}_\lambda$: The radial thrust alignment, where $\bar{\varphi}_\lambda = \omega_c$ (control is most effective near perigee).
- $H = \varphi - \bar{\varphi}_{\text{oop}}$: The out-of-plane angular offset, with $\bar{\varphi}_{\text{oop}} = \arctan 2(\delta i_{y,d}, \delta i_{x,d})$ optimizing inclination control.
- $\varphi = \omega_c + M_c$: The mean argument of latitude.

The raised cosine terms ($\cos^N J$, $\cos^N K$, $\cos^N H$) act as switching functions that amplify thrust commands near optimal orbital phases, helping conserve fuel while maintaining effective control.

**Additional Conversions**

To convert back from QNSROE to deputy OE, we execute the following:

$$\begin{aligned} a_d &= a_c(1 + \delta a), \\ \Omega_d &= \Omega_c + \frac{\delta i_y}{\sin i_c}, \\ i_d &= i_c + \delta i_x, \\ \omega_d &= \arctan 2(\delta e_y + e_c \sin \omega_c, \delta e_x + e_c \cos \omega_c), \\ e_d &= \sqrt{(\delta e_x + e_c \cos \omega_c)^2 + (\delta e_y + e_c \sin \omega_c)^2}, \\ M_d &= \delta\lambda - (\Omega_d - \Omega_c) \cos i_c + (M_c + \omega_c) - \omega_d. \end{aligned}$$

To recover the deputy's true anomaly $f_d$, we first solve Kepler's equation for the eccentric anomaly $E_d$ using Newton Raphson iteration:

$$E_d^{(k+1)} = E_d^{(k)} - \frac{E_d^{(k)} - e_d \sin E_d^{(k)} - M_d}{1 - e_d \cos E_d^{(k)}},$$

starting from $E_d^{(0)} = M_d$, and iterating until convergence within $\varepsilon = 10^{-10}$. Then:

$$f_d = 2 \arctan 2 \left( \sqrt{1+e_d} \sin \frac{E_d}{2}, \sqrt{1-e_d} \cos \frac{E_d}{2} \right).$$

This transformation enables back-and-forth conversion between the ROE we control and the classical orbital elements used for dynamics propagation and visualization.

**Measurement Sensor Model**

To simulate more realistic behavior, we model measurement and control noise. The measured QNSROE state is modeled as:

$$\delta\overline{\boldsymbol{\alpha}}_{\text{meas}}(t) = \delta\overline{\boldsymbol{\alpha}}(t) + \boldsymbol{\eta}_{\text{sensor}},$$

where $\boldsymbol{\eta}_{\text{sensor}}$ is zero-mean Gaussian noise:

$$\boldsymbol{\eta}_{\text{sensor}} \sim \mathcal{N}(\mathbf{0}, \boldsymbol{\Sigma}_{\text{sensor}}), \quad \boldsymbol{\Sigma}_{\text{sensor}} = \text{diag}\left(10^{-4},\, 10^{-5},\, 10^{-6},\, 10^{-6},\, 10^{-7},\, 10^{-7}\right).$$

These standard deviations correspond to noise in $\delta a$, $\delta\lambda$, $\delta e_x$, $\delta e_y$, $\delta i_x$, and $\delta i_y$, respectively.

**Actuator Model**

Control inputs are also subject to additive noise:

$$\boldsymbol{u}_{\text{applied}}(t) = \boldsymbol{u}_{\text{nom}}(t) + \boldsymbol{\eta}_{\text{act}},$$

with $\boldsymbol{u}_{\text{nom}}(t)$ the nominal control, and:

$$\boldsymbol{\eta}_{\text{act}} \sim \mathcal{N}(\mathbf{0}, \boldsymbol{\Sigma}_{\text{act}}), \quad \boldsymbol{\Sigma}_{\text{act}} = \text{diag}\left(10^{-4},\, 10^{-4},\, 10^{-4}\right).$$

These standard deviations reflect small uncertainties in $\Delta V_r$, $\Delta V_t$, and $\Delta V_n$ (m/s).

In implementation, we compute $\boldsymbol{u}_{\text{nom}}$ using the noisy state $\delta\overline{\boldsymbol{\alpha}}_{\text{meas}}$, and apply $\boldsymbol{u}_{\text{applied}}$ to the true system mirroring real-world imperfections.

### 6.2.2 Justification

We justify our choices for the continuous control law, proving Lyapunov compatibility and explaining the dynamics model, state representation, and noise model implementation. We show that $\boldsymbol{A}_c$, $\boldsymbol{B}_c$, $V$, and $\dot{V}$ satisfy Lyapunov conditions, confirm the controller's dynamics model, and rationalize the noise model, keeping it fast and to the point.

#### Dynamics Model and State Representation

The deputy's ROE dynamics are:

$$\delta\dot{\overline{\boldsymbol{\alpha}}} = \boldsymbol{A}_c\delta\overline{\boldsymbol{\alpha}} + \boldsymbol{B}_c\boldsymbol{u},$$

propagated via Euler integration ($\Delta t = 10\,\text{s}$). This is accurate for low Earth orbits, capturing $J_2$ effects, and simple enough for real-time control.

For the controller, we use the same ROE model with quasi-nonsingular ROE:

$$\delta\overline{\boldsymbol{\alpha}} = \begin{bmatrix}\delta a & \delta\lambda & \delta e_x & \delta e_y & \delta i_x & \delta i_y\end{bmatrix}^T.$$

This is justified by:

- **Quasi-Nonsingular ROE:** Chosen to avoid singularities present in classical ROE near circular orbits ($e_c = 5 \times 10^{-4}$), which is the case for our chief. These coordinates remain well-behaved for low eccentricities.
- **$J_2$-Perturbed Model:** The matrix $\boldsymbol{A}_c(t)$, computed via the State Transition Matrix (STM), incorporates $J_2$ effects. This ensures consistency between the control model and the high-fidelity nonlinear dynamics used for propagation.
- **Linear Model:** The use of a time-varying linear model facilitates tractable control design while still capturing the dominant relative dynamics under $J_2$ perturbations.

#### Lyapunov Compatibility of $\boldsymbol{A}_c(t)$ and $\boldsymbol{B}_c(t)$

The controller is designed using the structure:

$$\boldsymbol{u}(t) = -\boldsymbol{B}_c^{\dagger}(t)\left(\boldsymbol{A}_c(t)\,\delta\overline{\boldsymbol{\alpha}}(t) + \boldsymbol{P}(t)\left(\delta\overline{\boldsymbol{\alpha}}(t) - \delta\overline{\boldsymbol{\alpha}}_r\right)\right),$$

which ensures $\dot{V}(t) \le 0$ for the Lyapunov function $V(t) = \|\delta\overline{\boldsymbol{\alpha}}(t) - \delta\overline{\boldsymbol{\alpha}}_r\|^2$. The use of the Moore-Penrose pseudoinverse $\boldsymbol{B}_c^{\dagger}(t)$ guarantees that the control is applied in the direction that best reduces the Lyapunov function, within the bounds of available control authority.

$$\boldsymbol{u} = -\boldsymbol{B}_c^{\dagger}\left(\boldsymbol{A}_c\delta\overline{\boldsymbol{\alpha}} + \boldsymbol{P}(\delta\overline{\boldsymbol{\alpha}} - \delta\overline{\boldsymbol{\alpha}}_r)\right),$$

with $\boldsymbol{P} = \frac{1}{k}\text{diag}(\cos^N J, \cos^N K, \cos^N J, \cos^N J, \cos^N H, \cos^N H)$, $k = 1000$, $N = 14$. Substituting:

$$\delta\dot{\overline{\boldsymbol{\alpha}}} = \boldsymbol{A}_c\delta\overline{\boldsymbol{\alpha}} - \boldsymbol{B}_c\boldsymbol{B}_c^\dagger\left(\boldsymbol{A}_c\delta\overline{\boldsymbol{\alpha}} + \boldsymbol{P}(\delta\overline{\boldsymbol{\alpha}} - \delta\overline{\boldsymbol{\alpha}}_r)\right).$$

Since $\boldsymbol{B}_c$ is full rank (6x3, with linearly independent columns), $\boldsymbol{B}_c\boldsymbol{B}_c^\dagger \approx \boldsymbol{I}_6$, so:

$$\delta\dot{\overline{\boldsymbol{\alpha}}} \approx -\boldsymbol{P}(\delta\overline{\boldsymbol{\alpha}} - \delta\overline{\boldsymbol{\alpha}}_r).$$

The STM-derived $\boldsymbol{A}_c$ is continuous and bounded (as orbital elements evolve smoothly), satisfying Lyapunov's linear system requirements. $\boldsymbol{B}_c$'s full rank ensures controllability, making the pair ($\boldsymbol{A}_c$, $\boldsymbol{B}_c$) suitable for Lyapunov control.

**Lyapunov Function and Derivative**

We define a standard quadratic Lyapunov candidate:

$$V(t) = \frac{1}{2}\left(\delta\overline{\boldsymbol{\alpha}} - \delta\overline{\boldsymbol{\alpha}}_r\right)^T\left(\delta\overline{\boldsymbol{\alpha}} - \delta\overline{\boldsymbol{\alpha}}_r\right),$$

with time derivative:

$$\dot{V}(t) = \left(\delta\overline{\boldsymbol{\alpha}} - \delta\overline{\boldsymbol{\alpha}}_r\right)^T\delta\dot{\overline{\boldsymbol{\alpha}}}.$$

Substituting the closed-loop dynamics under control input:

$$\dot{V}(t) \approx -\left(\delta\overline{\boldsymbol{\alpha}} - \delta\overline{\boldsymbol{\alpha}}_r\right)^T\boldsymbol{P}(t)\left(\delta\overline{\boldsymbol{\alpha}} - \delta\overline{\boldsymbol{\alpha}}_r\right).$$

Since $\boldsymbol{P}(t)$ is diagonal and composed of terms like $\cos^N J$, $\cos^N K$, and $\cos^N H$, all non-negative, it is positive semi-definite. Therefore:

$$\dot{V}(t) \leq 0.$$

**Lyapunov Conditions**

- $V(t) \geq 0$, and $V(t) = 0$ only when $\delta\overline{\boldsymbol{\alpha}} = \delta\overline{\boldsymbol{\alpha}}_r$.
- $\dot{V}(t) \leq 0$, with equality either when $\delta\overline{\boldsymbol{\alpha}} = \delta\overline{\boldsymbol{\alpha}}_r$ or when $\boldsymbol{P}(t) = 0$ (i.e., at specific angles).
- The periodic $\cos^N$ terms ensure that $\boldsymbol{P}(t)$ is strictly positive over most of the orbit, guaranteeing that actuation occurs repeatedly and drives the state toward the reference.

These conditions ensure Lyapunov stability. Due to the periodic excitation provided by the $\cos^N$ functions, the convergence is asymptotic.

This is justified by:

- $V(t)$ is a simple, positive-definite quadratic function that directly penalizes tracking error.
- The structure of $\boldsymbol{P}(t)$, with its cosine-shaped gains, aligns control effort with the most effective orbital positionsâreducing fuel consumption while preserving convergence.

### Actuator Implementation

The actuators apply continuous thrusts $\boldsymbol{u} = [\Delta V_r,\, \Delta V_t,\, \Delta V_n]^T$, computed every $10\,\mathrm{s}$ using noisy state measurements. No explicit constraints are imposed on the control magnitude, which simplifies implementation.

This is justified by:

- Continuous control aligns with the assumptions in Lyapunov-based stability theory, which requires continuously differentiable dynamics.
- The structure of $\boldsymbol{P}(t)$ ensures that thrust is applied only when most fuel-efficient, which is essential for long-duration missions.

### Noise Model

$$\delta\overline{\boldsymbol{\alpha}}_{\text{meas}} = \delta\overline{\boldsymbol{\alpha}} + \boldsymbol{\eta}_{\text{sensor}}, \quad \boldsymbol{\eta}_{\text{sensor}} \sim \mathcal{N}\left(\mathbf{0},\, \operatorname{diag}(10^{-4},\, 10^{-5},\, 10^{-6},\, 10^{-6},\, 10^{-7},\, 10^{-7})\right).$$

$$\boldsymbol{u}_{\text{applied}} = \boldsymbol{u}_{\text{nom}} + \boldsymbol{\eta}_{\text{act}}, \quad \boldsymbol{\eta}_{\text{act}} \sim \mathcal{N}\left(\mathbf{0},\, \operatorname{diag}(10^{-4},\, 10^{-4},\, 10^{-4})\right).$$

This is justified by:

- **Realism:** Noise magnitudes reflect real-world spacecraft hardware, such as GPS sensors, star trackers, and cold gas thrusters.
- **Robustness:** The Lyapunov-based controller maintains stability under bounded measurement and actuation noise, since $\dot{V} \leq 0$ still holds in expectation.
- **Testing:** Incorporating noise enables performance evaluation under realistic mission conditions and highlights potential sensitivities.

#### 6.2.3 Results

We simulate 20 orbital periods, totaling approximately 108280 seconds, since one orbit takes:

$$T = 2\pi\sqrt{\frac{a_c^3}{\mu}} \approx 5414\,\mathrm{s}, \quad \text{with } a_c = 6771 \times 10^3\,\mathrm{m}, \quad \mu = 3.986004418 \times 10^{14}\,\mathrm{m}^3/\mathrm{s}^2.$$

We use a fixed time step $\Delta t = 10\,\mathrm{s}$ for both control updates and Euler integration, which provides sufficient resolution for our dynamics. The chief's orbit is numerically propagated with $J_2$ perturbations included, using tight relative and absolute tolerances ($10^{-12}$) to ensure high-precision dynamics.

We obtain the following results:

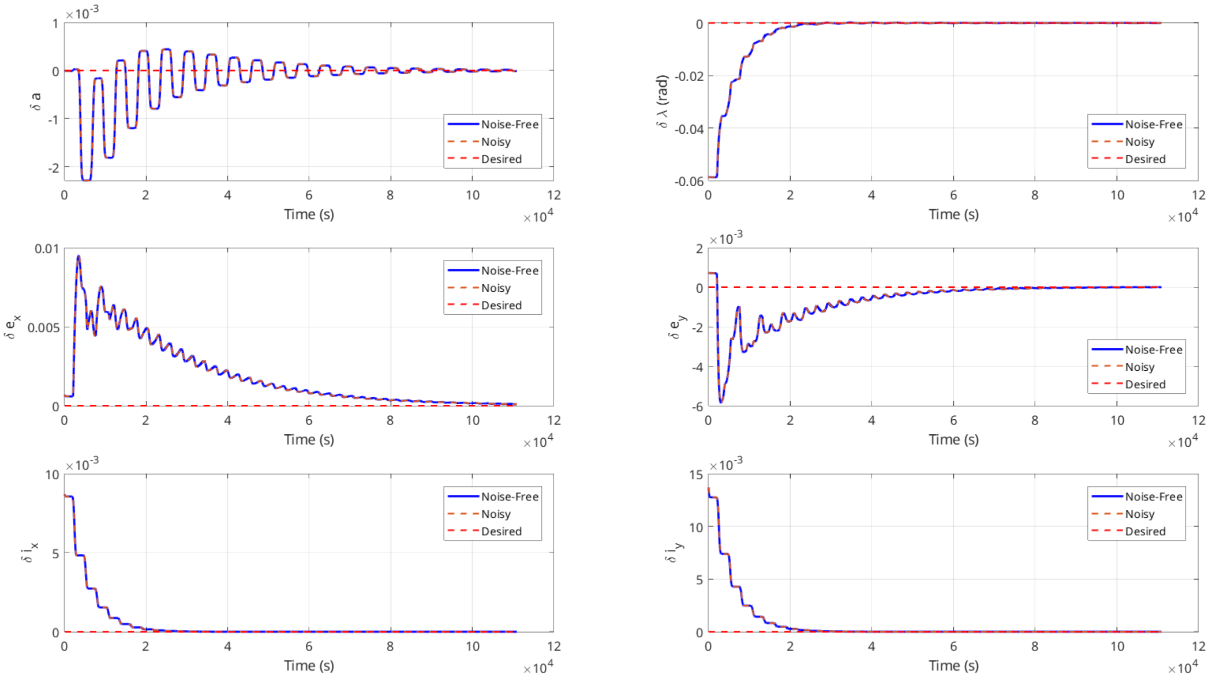


Figure 6.1: Evolution of deputy's relative quasi non singular orbital elements over time.

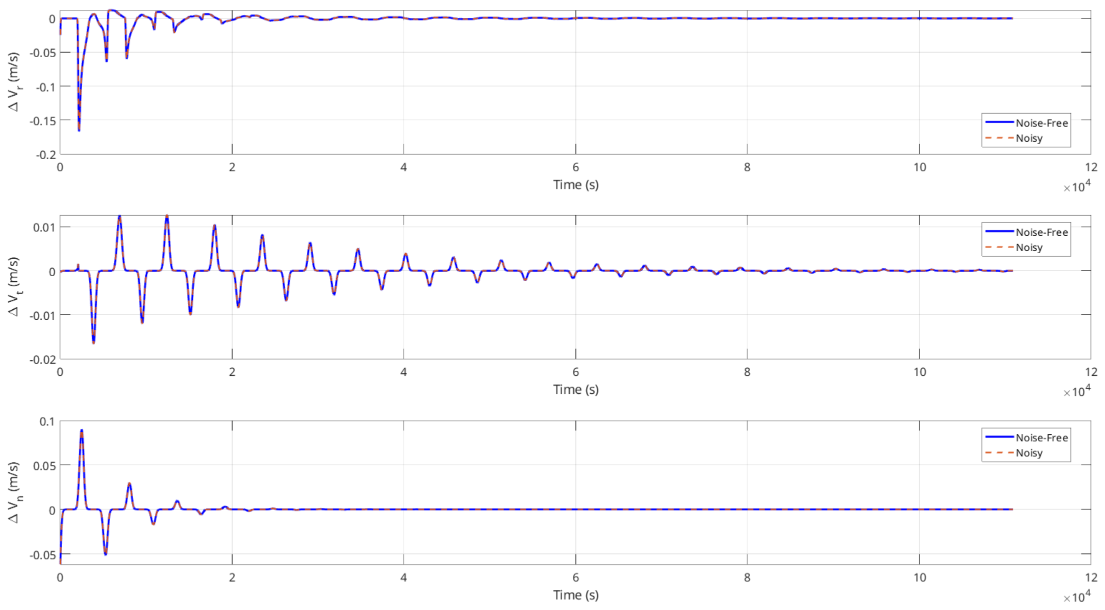


Figure 6.2: Evolution of the continuous impulse applied in RTN over time.

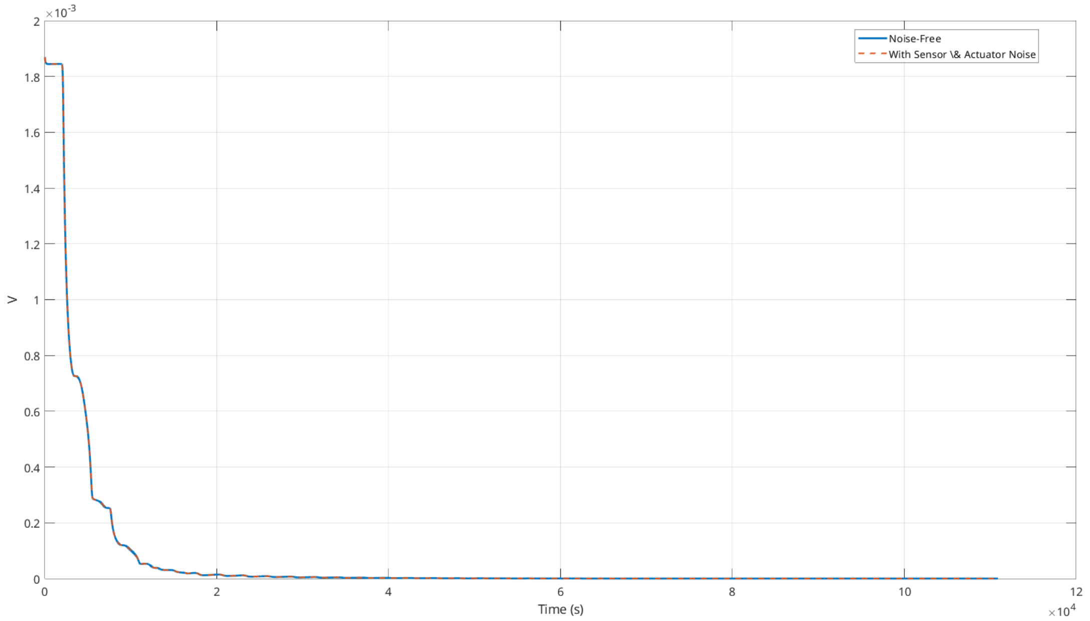


Figure 6.3: Evolution of the Lyapunov function over time.

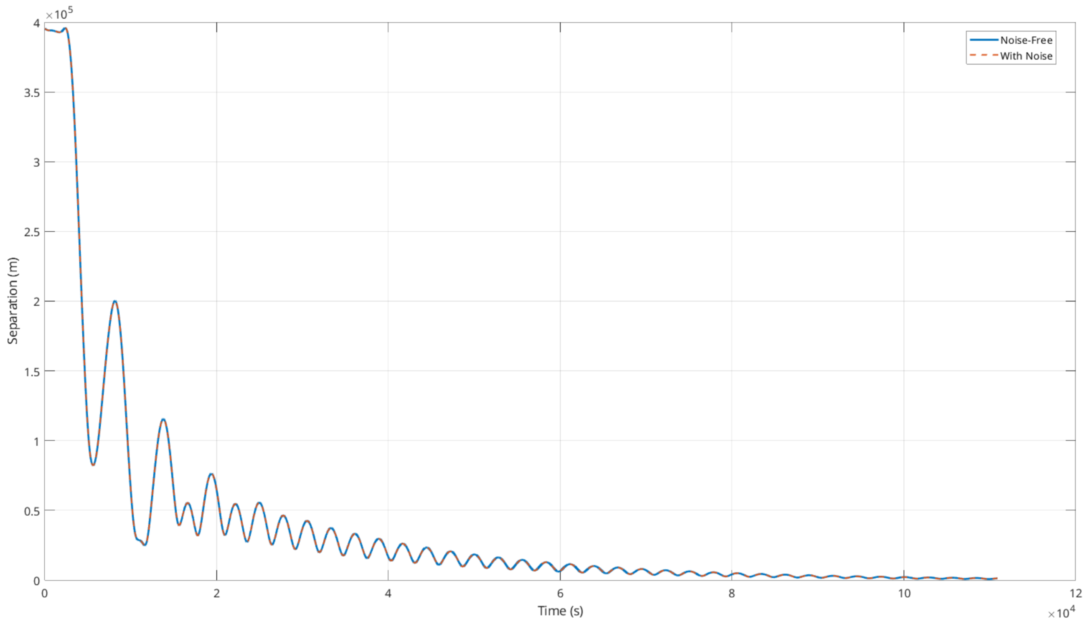


Figure 6.4: Evolution of the separation between the chief and the deputy spacecraft over time.

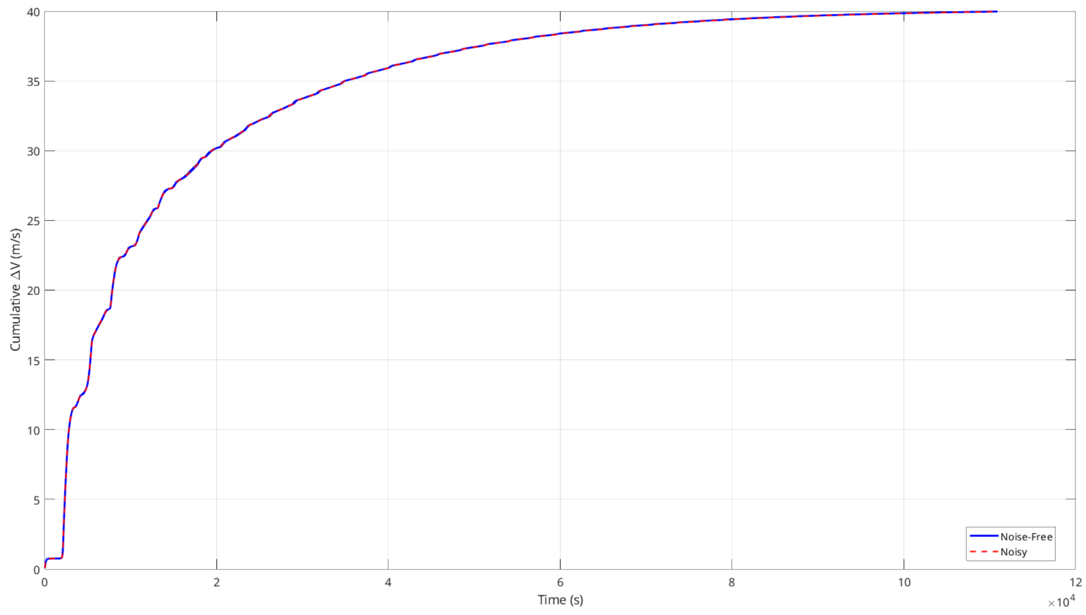


Figure 6.5: Evolution of the cumulative applied $\Delta V$ in the deputy's RTN frame.

The total final noisy $\Delta V_{tot_f}$ is 39.9738 m/s.

### 6.2.4 Performance and Results

**Strengths of Implementation**

- Continuous thrusting: The Lyapunov-based law produces smooth control commands at 0.1 Hz, avoiding large impulsive burns and reducing attitude control system disturbances.
- Robustness to noise: Both sensor and actuator noise are explicitly modeled; the controller maintains convergence and stability under Gaussian perturbations in state estimation and actuation.
- Gain optimization: Many gain matrices were tested empirically in a loop, and the selected gain yields a very low Δ v while still maintaining fast convergence (˜30 hours) and stability, as shown in our results.
- Low total $\Delta V$: The total $\Delta V$ expended is approximately 39.97 m/s over 20 orbits, which is substantially below our impulsive controller's expenditure and demonstrates the efficiency of the controller.

**Weaknesses of Implementation**

- No constraint handling: The controller does not explicitly enforce bounds on thrust magnitude, safety zones, or propellant usage. However, this was not an issue in our implementation, as all commanded impulses remained well below the thruster limit of 7.36 m/s.
- Linearization error: The linearized QNSROE model neglects higher-order $J_2$ effects and atmospheric drag. For the small corrections required in our maneuver, the model accuracy was sufficient and no significant deviations were observed.
- Fixed update rate: A constant 10 s control cadence may not capture rapid dynamics during perigee passages, which could lead to minor over- or under-thrusting. However, these effects were minimal in our scenario.

**Performance Compared to Expectations**

- Final QNSROE error: All six components converge to within $10^{-5}$ units of zero after 20 orbits, validating the controller's effectiveness.
- Lyapunov decrease: The Lyapunov function $V(t)$ decreased monotonically with $\dot{V}(t) \leq 0$ at all times, as expected from theory.
- Total $\Delta V$: The total $\Delta V$ required was approximately 39.97 m/s, which exceeds our design goals and remains well below our budget. It is significantly lower than the $\Delta V$ used by the impulsive controller (160 m/s), demonstrating the benefits of continuous actuation with an optimized gain.

**Failure Modes and Next Steps**

- Integrate thrust constraints: Future versions could include saturation limits or constrained optimization to ensure that $|\Delta V_r|$, $|\Delta V_t|$, and $|\Delta V_n|$ stay within hardware capabilities.
- Improve model fidelity: Higher-order geopotential terms, atmospheric drag, and solar radiation pressure could be included in $\boldsymbol{A}_c(t)$ to better capture real dynamics. The added complexity may or may not be justified depending on the mission.

- Adaptive update rate: A variable control frequency, especially higher rates near perigee, could improve tracking accuracy. Variable gain could also be explored to further improve convergence.
- Closed-loop feedback: Future implementations could use Model Predictive Control (MPC) or receding-horizon control to account for real-time measurements and correct for accumulated error. This will be addressed in upcoming assignments.

# 7 Problem Set 7

## 7.1 Finalize relative orbit control

### 7.1.1 Problem Setup: Rendezvous Scenario

We are simulating the following rendezvous control problem. This represents the maneuver of our capsule to the space station, as discussed in previous problem sets.

**Chief's Initial Orbital Elements:**

$$\boldsymbol{\alpha}_c = \begin{bmatrix} a_c \\ e_c \\ i_c \\ \Omega_c \\ \omega_c \\ \nu_c \end{bmatrix} = \begin{bmatrix} 6771 \times 10^3 \text{ m} \\ 5.0 \times 10^{-4} \\ 51.64^\circ \\ 257^\circ \\ 45^\circ \\ 30^\circ \end{bmatrix}$$

**Deputy's Initial Quasi-Non-Singular Relative Orbital Elements:**

$$\delta\boldsymbol{\alpha}_{\text{initial}} = \begin{bmatrix} \delta a_d \\ \delta\lambda_d \\ \delta e_{x,d} \\ \delta e_{y,d} \\ \delta i_{x,d} \\ \delta i_{y,d} \end{bmatrix} = \begin{bmatrix} 0 \\ -3.3773^\circ \\ 0.0007 \\ 0.0007 \\ 0.4989^\circ \\ 0.7850^\circ \end{bmatrix}$$

**Deputy's Final Quasi-Non-Singular Relative Orbital Elements:**

$$\delta\boldsymbol{\alpha}_{\text{desired}} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}$$

### 7.1.2 Comparison of Impulsive vs. Continuous Control

We compare impulsive and continuous control approaches based on four key criteria:

1. Final state error
2. Total $\Delta v$
3. Intermediate trajectory smoothness
4. Time required

Among these, final state accuracy and total $\Delta v$ are the most critical metrics, while intermediate trajectory control and maneuver duration are secondary.

The continuous controller demonstrates clear advantages in final state accuracy, total $\Delta v$, and trajectory smoothness. Because it applies small corrections throughout the transfer, the continuous strategy closely tracks the desired trajectory and achieves near-zero residuals at the final epoch. In contrast, impulsive control, despite being optimized using a linearized state transition model, accumulates small residual errors due to model mismatch and discretization of thrust events. Additonally, the path it follows does not approach final elements as smoothly and predictably as continuous control does. The drift between impulsive controls leaves greater room for error and oscillations.

In terms of total $\Delta v$, the continuous solution requires significantly less fuel overall (39.9738 m/s vs. 160.116 m/s). This is because it exploits the system dynamics continuously, applying thrust only as necessary to stay on track. The impulsive schedule, though optimized for minimum norm under linear dynamics, does not compensate for unmodeled perturbations and short-period effects with larger corrections at discrete times.

Furthermore, continuous control enables smaller individual thrusts, improving compatibility with low-thrust actuators and reducing mechanical stress. Impulsive control, by contrast, may demand large instantaneous maneuvers that are not feasible in some propulsion architectures.

As far as maneuver time, the two methods resulted in similar outcomes.

Overall, the continuous method outperforms impulsive control in final accuracy, fuel efficiency, and trajectory smoothness, making it the preferred strategy when real-time actuation is available.

## 7.2 Kick-off of Navigation System Design

### 7.2.1 State Representation

Designing a navigation system for a spacecraft formation starts with choosing how to represent each spacecraft's state.

#### Cartesian Coordinates

One option is Cartesian coordinates, where the state is a six-dimensional vector of three-dimensional position and velocity in an inertial frame, like the Earth-Centered Inertial (ECI) frame.

This approach feels natural because sensors such as GPS or radar often provide position and velocity data directly, making the measurement model straightforward, especially for an Extended Kalman Filter (EKF)/ Unscented Kalman Filter (UKF) that handle nonlinearities well. Cartesian coordinates are versatile, working for any orbit-circular, elliptical, or hyperbolic-without needing adjustments.

However, orbital dynamics in Cartesian coordinates are nonlinear, requiring careful linearization in EKF or sigma point propagation in UKF. For $N$ spacecraft, the state dimension grows to $6N$, which can tax computational resources, and modeling relative motion between spacecraft becomes complex, especially in eccentric orbits where inter-satellite dynamics are intricate.

#### Orbital Elements

Another approach uses orbital elements, such as the classical or equinoctial Keplerian set.

These elements describe an orbit's shape, orientation, and position in a physically meaningful way, making them intuitive for orbital missions. They evolve slowly under perturbations like $J_2$ or atmospheric drag, which simplifies long-term dynamics prediction in both EKF and UKF. For formations, differential orbital elements can compactly represent relative offsets, such as along-track separations.

However, sensors typically provide Cartesian-like data, so transforming measurements to orbital elements creates nonlinear measurement models, complicating the filter design. This complexity can challenge the filter's performance.

#### Absolute vs. Relative States

Beyond coordinate choice, the state can represent absolute or relative positions.

Absolute states estimate each spacecraft's position and velocity independently, providing a complete picture in the inertial frame, which is useful for mission-level tasks like ground communication or orbit determination. For $N$ spacecraft, this approach results in a $6N$-dimensional state, which grows computationally expensive as the formation size increases.

Relative states, defined with respect to a reference spacecraft (the chief), focus on inter-satellite offsets, aligning with formation control goal. Relative states reduce dimensionality for deputies but may lose absolute context, complicating tasks requiring inertial positioning.

A mixed approach, combining the chief's absolute state with deputies' relative states, balances these needs. It supports mission requirements while prioritizing the formation's relative geometry, offering a scalable solution for large formations.

**Mixed Representation with Quasi-Nonsingular ROE**

We choose to represent the chief spacecraft's orbit using absolute Cartesian coordinates in the ECI frame and the deputy spacecraft's orbits using quasi-nonsingular (QNS) relative orbital elements (ROE) [8].

For the chief, the state is a six-dimensional vector of position and velocity, defined as:

$$\mathbf{x}_c = [x_{c,\text{ECI}}, y_{c,\text{ECI}}, z_{c,\text{ECI}}, v_{x,c,\text{ECI}}, v_{y,c,\text{ECI}}, v_{z,c,\text{ECI}}]^\top \in \mathbb{R}^6 \tag{36}$$

This captures the chief's inertial position and velocity, aligning directly with measurements like GPS or radar, which ensures a near-linear measurement model.

For the deputy, the QNS ROE vector is defined as:

$$\delta\overline{\boldsymbol{\alpha}}_d = [\delta a_d, \delta\lambda_d, \delta e_{x,d}, \delta e_{y,d}, \delta i_{x,d}, \delta i_{y,d}]^\top \tag{37}$$

where the components are:

- $\delta a_d = \dfrac{a_d - a_c}{a_c}$: Relative semi-major axis, normalized by the chief's semi-major axis $a_c$ (dimensionless).
- $\delta\lambda_d = M_d + \omega_d + \Omega_d \cos i_c - (M_c + \omega_c + \Omega_c \cos i_c)$: Relative mean longitude (radians).
- $\delta e_{x,d} = e_d \cos\omega_d - e_c \cos\omega_c$: $x$-component of the relative eccentricity vector (dimensionless).
- $\delta e_{y,d} = e_d \sin\omega_d - e_c \sin\omega_c$: $y$-component of the relative eccentricity vector (dimensionless).
- $\delta i_{x,d} = i_d - i_c$: Difference in inclination (radians).
- $\delta i_{y,d} = (\Omega_d - \Omega_c)\sin i_c$: $y$-component of the relative inclination vector (radians).

The consolidated final state for a formation with one chief and one deputy is:

$$\mathbf{x} = [\mathbf{x}_c^\top, \delta\overline{\boldsymbol{\alpha}}_d^\top]^\top \in \mathbb{R}^{12} \tag{38}$$

### 7.2.2 Kalman Filter

For the Kalman Filter (KF), a linear state-space model with a constant state transition matrix would typically be required. However, given the nonlinear orbital dynamics, inclusion of control inputs, and use of mixed absolute (chief in ECI) and relative (deputy in QNS ROE) states, a standard KF is insufficient. It cannot accurately capture the nonlinear physics or the coupled propagation of the formation. Therefore, we do not use the KF for prediction, as it would result in poor performance due to model mismatches and unmodeled nonlinearities.

### 7.2.3 Extended Kalman Filter - Model Dynamics and Linearization

#### Continuous and Discretized Nonlinear Dynamics

[8]

The EKF state prediction for the spacecraft formation is defined as:

$$\mathbf{x}_k = \mathbf{f}_k(\mathbf{x}_{k-1}, \mathbf{u}_k, \mathbf{w}_k) \tag{39}$$

where:

- $\mathbf{x}_k = [\mathbf{x}_{c,k}^\top, \delta\overline{\boldsymbol{\alpha}}_{d,k}^\top]^\top \in \mathbb{R}^{12}$: State vector, combining the chief's Cartesian ECI coordinates and the deputy's ROE.
- $\mathbf{u}_k = [\mathbf{u}_{c,k}^\top, \mathbf{u}_{d,k}^\top]^\top \in \mathbb{R}^6$: Control inputs, representing thruster $\Delta V$ forces for the chief and deputy.
- $\mathbf{w}_k = [\mathbf{w}_{c,k}^\top, \mathbf{w}_{d,k}^\top]^\top \sim \mathcal{N}(\mathbf{0}, \mathbf{Q}_k)$: Process noise.

The chief's continuous dynamics are:

$$\dot{\mathbf{x}}_c = \begin{bmatrix} \mathbf{v}_c \\ -\frac{\mu}{|\mathbf{r}_c|^3}\mathbf{r}_c + \mathbf{a}_{J_2} + \frac{1}{m_c}\mathbf{u}_c \end{bmatrix} \tag{40}$$

where

$$\mathbf{r}_c = \begin{bmatrix} x_{c,\text{ECI}} \\ y_{c,\text{ECI}} \\ z_{c,\text{ECI}} \end{bmatrix}, \qquad \mathbf{v}_c = \begin{bmatrix} v_{x,c,\text{ECI}} \\ v_{y,c,\text{ECI}} \\ v_{z,c,\text{ECI}} \end{bmatrix},$$

$$\mu = 3.986 \times 10^{14}\,\text{m}^3/\text{s}^2, \qquad J_2 = 1.0826 \times 10^{-3},$$

$$R = 6378.137\,\text{km}, \qquad \mathbf{u}_c \in \mathbb{R}^3,$$

$m_c$ is the chief's mass,

$$\mathbf{a}_{J_2} = -\frac{3\mu J_2 R^2}{2|\mathbf{r}_c|^5} \begin{bmatrix} x\left(1 - 5\frac{z^2}{|\mathbf{r}_c|^2}\right) \\ y\left(1 - 5\frac{z^2}{|\mathbf{r}_c|^2}\right) \\ z\left(3 - 5\frac{z^2}{|\mathbf{r}_c|^2}\right) \end{bmatrix}.$$

The discretized chief's dynamics are:

$$\mathbf{f}_{c,k} = \mathbf{x}_{c,k-1} + \int_{t_{k-1}}^{t_k} \begin{bmatrix} \mathbf{v}_c \\ -\frac{\mu}{|\mathbf{r}_c|^3}\mathbf{r}_c + \mathbf{a}_{J_2} + \frac{1}{m_c}\mathbf{u}_{c,k} \end{bmatrix} dt + \mathbf{w}_{c,k} \tag{41}$$

The deputy's continuous ROE dynamics are:

$$\delta\dot{\overline{\boldsymbol{\alpha}}}_d = \mathbf{A}_c(t)\delta\overline{\boldsymbol{\alpha}}_d + \mathbf{B}_c(t)\mathbf{u}_d \tag{42}$$

where $\mathbf{A}_c(t)$ is the plant matrix, $\mathbf{B}_c(t)$ is the control input matrix, and $\mathbf{u}_d = [\Delta V_r, \Delta V_t, \Delta V_n]^\top \in \mathbb{R}^3$ is the deputy's thruster input.

The state transition matrix (STM) $\Phi_{J2,qns}(\tau, t)$ satisfies:

$$\delta\overline{\boldsymbol{\alpha}}(t+\tau) = \Phi_{J2,qns}(\tau, t)\delta\overline{\boldsymbol{\alpha}}(t) \tag{43}$$

For a small time step $\tau = 10^{-3}\,\text{s}$, we approximate:

$$\Phi_{J2,qns}(\tau, t) \approx \mathbf{I}_6 + \mathbf{A}_c(t)\tau \tag{44}$$

yielding:

$$\mathbf{A}_c(t) \approx \frac{\Phi_{J2,qns}(\tau, t) - \mathbf{I}_6}{\tau} \tag{45}$$

The STM $\Phi_{J2,qns}(\tau, t)$ is computed using the chief's orbital elements at time $t$, accounting for $J_2$ effects. It has the form:

$$\Phi_{J2,qns}(\tau, t) = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ \phi_{21} & 1 & \phi_{23} & \phi_{24} & \phi_{25} & 0 \\ \phi_{31} & 0 & \phi_{33} & \phi_{34} & \phi_{35} & 0 \\ \phi_{41} & 0 & \phi_{43} & \phi_{44} & \phi_{45} & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ \phi_{61} & 0 & \phi_{63} & \phi_{64} & \phi_{65} & 1 \end{bmatrix},$$

with the non-zero elements defined as:

$$\phi_{21} = -\left(\frac{3}{2}n_c + \frac{7}{2}\kappa EP\right)\tau,$$

$$\phi_{23} = \kappa e_{xi}FGP\tau, \quad \phi_{24} = \kappa e_{yi}FGP\tau, \quad \phi_{25} = -\kappa FS\tau,$$

$$\phi_{31} = \frac{7}{2}\kappa e_{yf}Q\tau,$$

$$\phi_{33} = \cos(\omega_{\text{dot}}\tau) - 4\kappa e_{xi}e_{yf}GQ\tau,$$

$$\phi_{34} = -\sin(\omega_{\text{dot}}\tau) - 4\kappa e_{yi}e_{yf}GQ\tau,$$

$$\phi_{35} = 5\kappa e_{yf}S\tau,$$

$$\phi_{41} = -\frac{7}{2}\kappa e_{xf} Q\tau,$$

$$\phi_{43} = \sin(\omega_{\text{dot}}\tau) + 4\kappa e_{xi} e_{xf} GQ\tau,$$

$$\phi_{44} = \cos(\omega_{\text{dot}}\tau) + 4\kappa e_{yi} e_{xf} GQ\tau,$$

$$\phi_{45} = -5\kappa e_{xf} S\tau,$$

$$\phi_{61} = \frac{7}{2}\kappa S\tau,$$

$$\phi_{63} = -4\kappa e_{xi} GS\tau, \quad \phi_{64} = -4\kappa e_{yi} GS\tau,$$

$$\phi_{65} = 2\kappa T\tau.$$

To make sense of these, we define all the sub-variables used in the STM:

- $n_c = \sqrt{\frac{\mu}{a_c^3}}$: The chief's mean motion (rad/s), where $\mu = 3.986004418 \times 10^{14}\,\text{m}^3/\text{s}^2$ is Earth's gravitational parameter.
- $\eta = \sqrt{1 - e_c^2}$: A parameter related to the chief's eccentricity.
- $\kappa = \frac{3}{4}\frac{J_2 R^2 \sqrt{\mu}}{a_c^{7/2}\eta^4}$: A coefficient for $J_2$ perturbations, with $J_2 = 1.08262668 \times 10^{-3}$ (Earth's second zonal harmonic) and $R = 6378137\,\text{m}$ (Earth's equatorial radius).
- $E = 1 + \eta$: An auxiliary term.
- $F = 4 + 3\eta$: Another auxiliary term.
- $G = \frac{1}{\eta^2}$: Scales the eccentricity effects.
- $P = 3\cos^2 i_c - 1$: Inclination-dependent term.
- $Q = 5\cos^2 i_c - 1$: Another inclination-dependent term.
- $S = \sin(2i_c)$: Double-angle inclination term.
- $T = \sin^2 i_c$: Squared sine of inclination.
- $e_{xi} = e_c \cos\omega_c$, $e_{yi} = e_c \sin\omega_c$: Initial eccentricity vector components of the chief.
- $\dot{\omega} = \kappa Q$: The secular rate of change of the argument of perigee due to $J_2$ (rad/s).

- $\omega_f = \omega_c + \dot{\omega}\tau$: The argument of perigee after time $\tau$.
- $e_{xf} = e_c \cos\omega_f$, $e_{yf} = e_c \sin\omega_f$: Final eccentricity vector components.

We compute the STM at each time step using the chief's current orbital elements.

The control input matrix $\boldsymbol{B}_c(t)$ connects our control thrusts to changes in the ROE. It tells us how much each thrust component ($\Delta V_r$, $\Delta V_t$, $\Delta V_n$) affects the ROE. The matrix is:

$$\boldsymbol{B}_c(t) = \frac{1}{a_c n_c}\begin{bmatrix} \frac{2e_c \sin f_c}{\eta} & \frac{2(1+e_c\cos f_c)}{\eta} & 0 \\ -\frac{2\eta^2}{1+e_c\cos f_c} & 0 & 0 \\ \eta\sin(\omega_c+f_c) & \eta\left(\frac{(2+e_c\cos f_c)\cos(\omega_c+f_c)+e_{xc}}{1+e_c\cos f_c}\right) & \eta e_{yc}\frac{\sin(\omega_c+f_c)}{\tan i_c(1+e_c\cos f_c)} \\ -\eta\cos(\omega_c+f_c) & \eta\left(\frac{(2+e_c\cos f_c)\sin(\omega_c+f_c)+e_{yc}}{1+e_c\cos f_c}\right) & -\eta e_{xc}\frac{\sin(\omega_c+f_c)}{\tan i_c(1+e_c\cos f_c)} \\ 0 & 0 & \eta\frac{\cos(\omega_c+f_c)}{1+e_c\cos f_c} \\ 0 & 0 & \eta\frac{\sin(\omega_c+f_c)}{1+e_c\cos f_c} \end{bmatrix},$$

Where we've got a few more variables to define:

- $e_{xc} = e_c\cos\omega_c$, $e_{yc} = e_c\sin\omega_c$: The chief's eccentricity vector components, same as $e_{xi}$, $e_{yi}$ but relabeled for clarity.
- $f_c$: The chief's true anomaly at time $t$, obtained from the propagated orbit.
- $1 + e_c\cos f_c$: A term that adjusts for the orbit's shape at the current position.
- Other terms like $\eta$, $n_c$, $a_c$, $i_c$, and $\omega_c$ are the same as previously defined.

The factor $\dfrac{1}{a_c n_c}$ scales the matrix to match the chief's orbit size and speed, ensuring the control inputs have the correct effect on the deputy spacecraft dynamics.

The discrete deputy's dynamics are derived from the continuous solution:

$$\delta\overline{\boldsymbol{\alpha}}_d(t_k) = \Phi_{J2,qns}(t_k - t_{k-1}, t_{k-1})\delta\overline{\boldsymbol{\alpha}}_d(t_{k-1}) + \int_{t_{k-1}}^{t_k} \Phi_{J2,qns}(t_k - s, s)\mathbf{B}_c(s)\mathbf{u}_d(s)\,ds \tag{46}$$

For $\Delta t = t_k - t_{k-1}$, assuming constant $\mathbf{u}_d$, and approximating $\Phi_{J2,qns}(t_k - s, s) \approx \mathbf{I}_6$ for small $\Delta t$, we obtain:

$$\mathbf{f}_{d,k} = \Phi_{J2,qns}(\Delta t, t_{k-1})\delta\overline{\boldsymbol{\alpha}}_{d,k-1} + \mathbf{B}_c(t_{k-1})\Delta t\mathbf{u}_{d,k} + \mathbf{w}_{d,k} \tag{47}$$

**Dynamics Sensitivity Matrix**

The EKF covariance update requires linearizing the nonlinear discrete dynamics around the current state estimate to compute the state transition and control input matrices.

The state transition matrix $\mathbf{F}_k$ and control input matrix $\mathbf{B}_k$ are defined as the Jacobians of the nonlinear dynamics with respect to the state and control vectors, respectively, evaluated at the previous state estimate:

$$\mathbf{F}_k = \left.\frac{\partial \mathbf{f}_k}{\partial \mathbf{x}}\right|_{\hat{\mathbf{x}}_{k-1|k-1},\mathbf{u}_k}, \quad \mathbf{B}_k = \left.\frac{\partial \mathbf{f}_k}{\partial \mathbf{u}}\right|_{\hat{\mathbf{x}}_{k-1|k-1},\mathbf{u}_k} \tag{48}$$

These matrices capture how small perturbations in the state and control vectors influence the predicted state, and are critical for propagating the covariance in the EKF.

The state transition matrix is:

$$\mathbf{F}_k = \begin{bmatrix} \mathbf{F}_{c,k} & \mathbf{0}_{6\times6} \\ \mathbf{0}_{6\times6} & \Phi_{J2,qns}(\Delta t, t_{k-1}) \end{bmatrix} \tag{49}$$

where $\mathbf{F}_{c,k} = e^{\mathbf{A}_{c,\text{cont}}\Delta t}$, and:

$$\mathbf{A}_{c,\text{cont}} = \begin{bmatrix} \mathbf{0}_{3\times3} & \mathbf{I}_{3\times3} \\ \frac{\partial}{\partial \mathbf{r}_c}\left(-\frac{\mu}{|\mathbf{r}_c|^3}\mathbf{r}_c + \mathbf{a}_{J_2}\right) & \mathbf{0}_{3\times3} \end{bmatrix} \tag{50}$$

The gravity term is:

$$\frac{\partial}{\partial \mathbf{r}_c}\left(-\frac{\mu}{|\mathbf{r}_c|^3}\mathbf{r}_c\right) = -\frac{\mu}{|\mathbf{r}_c|^3}\mathbf{I}_3 + \frac{3\mu}{|\mathbf{r}_c|^5}\mathbf{r}_c\mathbf{r}_c^\top \tag{51}$$

The $J_2$ term is approximated numerically at $\hat{\mathbf{x}}_{c,k-1|k-1}$:

$$\left[\frac{\partial \mathbf{a}_{J_2}}{\partial \mathbf{r}_c}\right]_{ij} \approx \frac{a_{J_2,i}(\mathbf{r}_c + h\mathbf{e}_j) - a_{J_2,i}(\mathbf{r}_c - h\mathbf{e}_j)}{2h}, \quad h = 10^{-4}\,\text{m} \tag{52}$$

where $\mathbf{e}_j \in \mathbb{R}^3$ is the unit vector in the $j$-th coordinate direction, used to perturb the $j$-th component of $\mathbf{r}_c$ during finite differencing.

The control input matrix is:

$$\mathbf{B}_k = \begin{bmatrix} \begin{bmatrix} \mathbf{0}_{3\times3} \\ \frac{1}{m_c}\mathbf{I}_{3\times3} \end{bmatrix}\Delta t & \mathbf{0}_{6\times3} \\ \mathbf{0}_{6\times3} & \mathbf{B}_c(t_{k-1})\Delta t \end{bmatrix} \tag{53}$$

This model ensures precise covariance updates by capturing the chief's nonlinear dynamics numerically and using the deputy's STM, supporting accurate state estimation for formation control in orbital missions.

### 7.2.4 Available Sensors On-board

The spacecraft formation uses onboard sensors to support EKF-based estimation of the 12-dimensional state vector.

The chief relies on absolute measurements to determine its inertial position and velocity, while the deputy uses relative measurements to maintain formation geometry at meter-to-millimeter scales with sub-millimeter accuracy.

The chief is equipped with a GNSS receiver providing:

- **Pseudorange:** Time-of-flight measurements with 10-50 cm accuracy (improved with differential GNSS).
- **Carrier-phase:** Milliliter-level accuracy after ambiguity resolution.
- **Range rate:** Doppler-based velocity with mm/s precision.

These are processed into absolute position and velocity in the ECI frame.

Both spacecraft carry an inter-satellite laser ranging system, providing:

- **Range:** Milliliter-level accuracy
- **Range rate:** Doppler-based velocity with 1 to 10 mm/s precision.

These measurements yield relative position, velocity, and ROE differences.

The deputy also has an optical camera for azimuth/elevation measurements (arcsecond-level), which, combined with range data, enhances relative positioning.

Each spacecraft includes an IMU, measuring angular rates and accelerations for attitude and maneuver estimation.

### 7.2.5 Nonlinear Measurement Model

The nonlinear measurement model maps the state vector to sensor measurements, incorporating noise, to support the EKF's measurement update. The state, combining the chief's ECI position and velocity and the deputy's ROE, requires measurements that reflect both absolute and relative dynamics. The model is defined as $\mathbf{z}_k = \mathbf{h}_k(\mathbf{x}_k) + \mathbf{v}_k$, where $\mathbf{v}_k \sim \mathcal{N}(\mathbf{0}, \mathbf{R}_k)$ is measurement noise, and includes absolute position and velocity for the chief, and range, range rate, and bearing angles for the deputy's relative state.

The measurement vector is:

$$\mathbf{z}_k = \begin{bmatrix} \mathbf{r}_{c,k} \\ \mathbf{v}_{c,k} \\ \|\mathbf{r}_{d,k} - \mathbf{r}_{c,k}\|_2 \\ \dfrac{(\mathbf{r}_{d,k} - \mathbf{r}_{c,k})^\top (\mathbf{v}_{d,k} - \mathbf{v}_{c,k})}{\|\mathbf{r}_{d,k} - \mathbf{r}_{c,k}\|_2} \\ \arctan\left(\dfrac{\Delta y_{d,k}}{\Delta x_{d,k}}\right) \\ \arcsin\left(\dfrac{\Delta z_{d,k}}{\|\Delta \mathbf{r}_{d,k}\|_2}\right) \end{bmatrix}, \quad \mathbf{h}_k(\mathbf{x}_k) = \begin{bmatrix} \mathbf{x}_{c,k}(1:3) \\ \mathbf{x}_{c,k}(4:6) \\ \sqrt{(\Delta \mathbf{r}_{d,k})^\top \Delta \mathbf{r}_{d,k}} \\ \dfrac{(\Delta \mathbf{r}_{d,k})^\top \Delta \mathbf{v}_{d,k}}{\|\Delta \mathbf{r}_{d,k}\|_2} \\ \arctan\left(\dfrac{\Delta y_{d,k}}{\Delta x_{d,k}}\right) \\ \arcsin\left(\dfrac{\Delta z_{d,k}}{\|\Delta \mathbf{r}_{d,k}\|_2}\right) \end{bmatrix} \tag{54}$$

where $\mathbf{r}_{c,k} = \mathbf{x}_{c,k}(1:3)$, $\mathbf{v}_{c,k} = \mathbf{x}_{c,k}(4:6)$, $\mathbf{r}_{d,k} = \mathbf{r}_{c,k} + \Delta\mathbf{r}_{d,k}$, and $\mathbf{v}_{d,k} = \mathbf{v}_{c,k} + \Delta\mathbf{v}_{d,k}$.

The relative position and velocity are nonlinear functions of the deputy's ROE [8]:

$$\begin{aligned} \Delta\mathbf{r}_{d,k} &= \mathbf{g}(\delta\overline{\boldsymbol{\alpha}}_{d,k}, \mathbf{r}_{c,k}, \mathbf{v}_{c,k}), \\ \Delta\mathbf{v}_{d,k} &= \mathbf{g}_v(\delta\overline{\boldsymbol{\alpha}}_{d,k}, \mathbf{r}_{c,k}, \mathbf{v}_{c,k}). \end{aligned}$$

The functions $\mathbf{g}$ and $\mathbf{g}_v$ transform ROEs to Cartesian coordinates using orbital mechanics. The model is nonlinear due to the presence of square roots, normalization, and trigonometric functions in the range, range rate, and bearing angle measurements, making it well-suited for linearization in the EKF measurement update.

### 7.2.6 Measurement Sensitivity Matrix

The measurement sensitivity matrix (Jacobian) used in the EKF update is defined as:

$$\mathbf{H}_k = \begin{bmatrix} \dfrac{\partial \mathbf{h}_k}{\partial \mathbf{x}_{c,k}} & \dfrac{\partial \mathbf{h}_k}{\partial \delta\overline{\boldsymbol{\alpha}}_{d,k}} \end{bmatrix} \in \mathbb{R}^{8\times 12} \tag{55}$$

For absolute measurements of the chief's position and velocity:

$$\frac{\partial \mathbf{r}_{c,k}}{\partial \mathbf{x}_{c,k}} = \begin{bmatrix} \mathbf{I}_{3\times3} & \mathbf{0}_{3\times3} \end{bmatrix} \tag{56}$$

$$\frac{\partial \mathbf{v}_{c,k}}{\partial \mathbf{x}_{c,k}} = \begin{bmatrix} \mathbf{0}_{3\times3} & \mathbf{I}_{3\times3} \end{bmatrix} \tag{57}$$

$$\frac{\partial}{\partial \delta\overline{\boldsymbol{\alpha}}_{d,k}} \mathbf{r}_{c,k}, \mathbf{v}_{c,k} = \mathbf{0}_{3\times6} \tag{58}$$

For relative range measurements, let $\Delta\mathbf{r}_{d,k}$ be the deputy's position relative to the chief, expressed via ROE:

$$\text{Range} = \|\Delta\mathbf{r}_{d,k}\|_2 \tag{59}$$

Then the gradient with respect to the chief's position is:

$$\frac{\partial}{\partial \mathbf{r}_{c,k}} \|\Delta\mathbf{r}_{d,k}\|_2 = \frac{\Delta\mathbf{r}_{d,k}^\top}{\|\Delta\mathbf{r}_{d,k}\|_2} \cdot \mathbf{G}_r \tag{60}$$

And with respect to the deputy's ROE:

$$\frac{\partial}{\partial \delta \overline{\boldsymbol{\alpha}}_{d,k}} \|\Delta \mathbf{r}_{d,k}\|_2 = \frac{\Delta \mathbf{r}_{d,k}^\top}{\|\Delta \mathbf{r}_{d,k}\|_2} \cdot \frac{\partial \mathbf{g}}{\partial \delta \overline{\boldsymbol{\alpha}}_{d,k}} \tag{61}$$

Similarly, for relative range rate measurements:

$$\text{Range rate} = \frac{d}{dt} \|\Delta \mathbf{r}_{d,k}\|_2 = \frac{\Delta \mathbf{r}_{d,k}^\top \Delta \mathbf{v}_{d,k}}{\|\Delta \mathbf{r}_{d,k}\|_2} \tag{62}$$

With derivatives involving:

$$\mathbf{G}_v = \frac{\partial \mathbf{g}_v}{\partial \mathbf{v}_{c,k}}, \quad \text{and} \quad \frac{\partial \mathbf{g}_v}{\partial \delta \overline{\boldsymbol{\alpha}}_{d,k}} \tag{63}$$

For bearing measurements from the deputy's camera (e.g., azimuth), modeled as:

$$\theta = \arctan\left(\frac{\Delta y_{d,k}}{\Delta x_{d,k}}\right) \tag{64}$$

The derivative with respect to the chief's position is:

$$\frac{\partial \theta}{\partial \mathbf{r}_{c,k}} = \frac{1}{\Delta x_{d,k}^2 + \Delta y_{d,k}^2} \begin{bmatrix} -\Delta y_{d,k} \\ \Delta x_{d,k} \\ 0 \end{bmatrix}^\top \cdot \mathbf{G}_r \tag{65}$$

with analogous expressions for derivatives with respect to the deputy's ROE.

Due to the inclusion of GNSS-based measurements, the model must also account for potential biases introduced by satellite and receiver clock offsets, as well as ionospheric delays. These effects can introduce systematic errors in pseudorange and carrier-phase data. To mitigate this and maintain the consistency of the EKF, the state vector is augmented to estimate and correct these biases during filtering.

## 8 Problem Set 8

The code used for this assignment is available in our GitHub repository [1] and the Appendix. See "Problem Set 8 Code" for code for each section. Additionally, see "Propagators", "Control Helpers" and "Utils" sections.

### 8.1 State Representation

For a spacecraft formation with one chief and one deputy, we represent the state using a mixed approach combining the chief's absolute Cartesian coordinates in the Earth-Centered Inertial (ECI) frame and the deputy's quasi-nonsingular (QNS) relative orbital elements (ROE). This choice provides a complete inertial reference for the chief's trajectory, while the deputy's ROE compactly capture relative motion, enhancing efficiency in formation control.

The chief's state is a six-dimensional vector of position and velocity:

$$\mathbf{x}_c = [x_{c,\text{ECI}}, y_{c,\text{ECI}}, z_{c,\text{ECI}}, v_{x,c,\text{ECI}}, v_{y,c,\text{ECI}}, v_{z,c,\text{ECI}}]^\top \in \mathbb{R}^6 \tag{66}$$

The deputy's QNS ROE state vector is:

$$\delta\overline{\boldsymbol{\alpha}}_d = [\delta a_d, \delta\lambda_d, \delta e_{x,d}, \delta e_{y,d}, \delta i_{x,d}, \delta i_{y,d}]^\top \tag{67}$$

where the components are:

- $\delta a_d = \dfrac{a_d - a_c}{a_c}$: Relative semi-major axis, normalized by the chief's semi-major axis $a_c$ (dimensionless).
- $\delta\lambda_d = M_d + \omega_d + \Omega_d \cos i_c - (M_c + \omega_c + \Omega_c \cos i_c)$: Relative mean longitude (radians).
- $\delta e_{x,d} = e_d \cos\omega_d - e_c \cos\omega_c$: $x$-component of the relative eccentricity vector (dimensionless).
- $\delta e_{y,d} = e_d \sin\omega_d - e_c \sin\omega_c$: $y$-component of the relative eccentricity vector (dimensionless).
- $\delta i_{x,d} = i_d - i_c$: Difference in inclination (radians).
- $\delta i_{y,d} = (\Omega_d - \Omega_c)\sin i_c$: $y$-component of the relative inclination vector (radians).

The consolidated final state for a formation with one chief and one deputy is:

$$\mathbf{x} = [\mathbf{x}_c^\top, \delta\overline{\boldsymbol{\alpha}}_d^\top]^\top \in \mathbb{R}^{12} \tag{68}$$

### 8.2 Dynamic Nonlinear Model

The nonlinear state prediction for the spacecraft formation is defined as:

$$\mathbf{x}_k = \mathbf{f}_k(\mathbf{x}_{k-1}, \mathbf{u}_k, \mathbf{w}_k) \tag{69}$$

where:

- $\mathbf{x}_i = [\mathbf{x}_{c,i}^\top, \delta\overline{\boldsymbol{\alpha}}_{d,i}^\top]^\top \in \mathbb{R}^{12}$: State vector at step i, combining the chief's Cartesian ECI coordinates and the deputy's ROE.
- $\mathbf{u}_k = [\mathbf{u}_{c,k}^\top, \mathbf{u}_{d,k}^\top]^\top \in \mathbb{R}^6$: Control inputs, representing thruster applied $\Delta V$ for the chief and deputy in m/s in ECI and RTN frame respectively.
- $\mathbf{w}_k = [\mathbf{w}_{c,k}^\top, \mathbf{w}_{d,k}^\top]^\top \sim \mathcal{N}(\mathbf{0}, \mathbf{Q}_k)$: Process noise.

The prediction function is decomposed into:

$$\mathbf{f}_k(\mathbf{x}_{k-1}, \mathbf{u}_k, \mathbf{w}_k) = \begin{bmatrix} \mathbf{f}_{c,k}(\mathbf{x}_{c,k-1}, \mathbf{u}_{c,k}, \mathbf{w}_{c,k}) \\ \mathbf{f}_{d,k}(\delta\overline{\boldsymbol{\alpha}}_{d,k-1}, \mathbf{u}_{d,k}, \mathbf{w}_{d,k}, \mathbf{x}_{c,k-1}) \end{bmatrix} \tag{70}$$

where $\mathbf{f}_{c,k}$ is the discretized chief's dynamics, and $\mathbf{f}_{d,k}$ is the discretized deputy's ROE dynamics, as defined subsequently.

### 8.2.1 Chief's Dynamics

The chief's continuous dynamics are:

$$\dot{\mathbf{x}}_c = \begin{bmatrix} \mathbf{v}_c \\ -\dfrac{\mu}{|\mathbf{r}_c|^3}\mathbf{r}_c + \mathbf{a}_{J_2} + \mathbf{a}_{\text{control}} \end{bmatrix} \tag{71}$$

where:

$$\mathbf{r}_c = \begin{bmatrix} x_{c,\text{ECI}} \\ y_{c,\text{ECI}} \\ z_{c,\text{ECI}} \end{bmatrix}, \qquad \mathbf{v}_c = \begin{bmatrix} v_{x,c,\text{ECI}} \\ v_{y,c,\text{ECI}} \\ v_{z,c,\text{ECI}} \end{bmatrix},$$

$$\mu = 3.986 \times 10^{14}\,\text{m}^3/\text{s}^2, \qquad J_2 = 1.0826 \times 10^{-3},$$

$$R = 6378.137\,\text{km}, \qquad \mathbf{u}_c \in \mathbb{R}^3,$$

$$\mathbf{a}_{J_2} = -\frac{3\mu J_2 R^2}{2|\mathbf{r}_c|^5} \begin{bmatrix} x\left(1 - 5\dfrac{z^2}{|\mathbf{r}_c|^2}\right) \\ y\left(1 - 5\dfrac{z^2}{|\mathbf{r}_c|^2}\right) \\ z\left(3 - 5\dfrac{z^2}{|\mathbf{r}_c|^2}\right) \end{bmatrix},$$

$$\mathbf{a}_{\text{control}} = \begin{cases} \mathbf{0}, & \text{if } \Delta t = 0, \\ \dfrac{\mathbf{u}_{c,k}}{\Delta t}, & \text{otherwise,} \end{cases}$$

The discretized chief's dynamics are obtained by integrating the continuous dynamics over the time step $\Delta t = t_k - t_{k-1}$:

$$\mathbf{f}_{c,k} = \mathbf{x}_{c,k-1} + \int_{t_{k-1}}^{t_k} \begin{bmatrix} \mathbf{v}_c(t) \\ -\dfrac{\mu}{|\mathbf{r}_c(t)|^3}\mathbf{r}_c(t) + \mathbf{a}_{J_2}(t) + \mathbf{a}_{\text{control},k} \end{bmatrix} dt + \mathbf{w}_{c,k} \tag{72}$$

For small $\Delta t$, we approximate the integral using Euler integration, evaluating the integrand at $t_{k-1}$:

$$\mathbf{f}_{c,k} \approx \mathbf{x}_{c,k-1} + \begin{bmatrix} \mathbf{v}_{c,k-1} \\ -\dfrac{\mu}{|\mathbf{r}_{c,k-1}|^3}\mathbf{r}_{c,k-1} + \mathbf{a}_{J_2,k-1} + \mathbf{a}_{\text{control},k} \end{bmatrix} \Delta t + \mathbf{w}_{c,k} \tag{73}$$

### 8.2.2 Deputy's Dynamics

The deputy's continuous ROE dynamics are:

$$\delta\dot{\overline{\boldsymbol{\alpha}}}_d = \mathbf{A}_c(t)\delta\overline{\boldsymbol{\alpha}}_d + \mathbf{B}_c(t)\mathbf{u}_d \tag{74}$$

where $\mathbf{A}_c(t)$ is the plant matrix, $\mathbf{B}_c(t)$ is the control input matrix, and $\mathbf{u}_d = [\Delta V_r, \Delta V_t, \Delta V_n]^\top \in \mathbb{R}^3$ is the deputy's thruster input.

The state transition matrix (STM) $\Phi_{J2,qns}(\tau, t)$ satisfies:

$$\delta\overline{\boldsymbol{\alpha}}(t+\tau) = \Phi_{J2,qns}(\tau, t)\delta\overline{\boldsymbol{\alpha}}(t) \tag{75}$$

The STM $\Phi_{J2,qns}(\tau, t)$ is computed using the chief's orbital elements at time $t$, accounting for $J_2$ effects. It has the form:

$$\Phi_{J2,qns}(\tau, t) = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ \phi_{21} & 1 & \phi_{23} & \phi_{24} & \phi_{25} & 0 \\ \phi_{31} & 0 & \phi_{33} & \phi_{34} & \phi_{35} & 0 \\ \phi_{41} & 0 & \phi_{43} & \phi_{44} & \phi_{45} & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ \phi_{61} & 0 & \phi_{63} & \phi_{64} & \phi_{65} & 1 \end{bmatrix},$$

with the non-zero elements defined as:

$$\phi_{21} = -\left(\frac{3}{2}n_c + \frac{7}{2}\kappa EP\right)\tau,$$

$$\phi_{23} = \kappa e_{xi}FGP\tau, \quad \phi_{24} = \kappa e_{yi}FGP\tau, \quad \phi_{25} = -\kappa FS\tau,$$

$$\phi_{31} = \frac{7}{2}\kappa e_{yf}Q\tau,$$

$$\phi_{33} = \cos(\omega_{\text{dot}}\tau) - 4\kappa e_{xi}e_{yf}GQ\tau,$$

$$\phi_{34} = -\sin(\omega_{\text{dot}}\tau) - 4\kappa e_{yi}e_{yf}GQ\tau,$$

$$\phi_{35} = 5\kappa e_{yf}S\tau,$$

$$\phi_{41} = -\frac{7}{2}\kappa e_{xf}Q\tau,$$

$$\phi_{43} = \sin(\omega_{\text{dot}}\tau) + 4\kappa e_{xi}e_{xf}GQ\tau,$$

$$\phi_{44} = \cos(\omega_{\text{dot}}\tau) + 4\kappa e_{yi} e_{xf} GQ\tau,$$

$$\phi_{45} = -5\kappa e_{xf} S\tau,$$

$$\phi_{61} = \frac{7}{2}\kappa S\tau,$$

$$\phi_{63} = -4\kappa e_{xi} GS\tau, \quad \phi_{64} = -4\kappa e_{yi} GS\tau,$$

$$\phi_{65} = 2\kappa T\tau.$$

To make sense of these, we define all the sub-variables used in the STM:

- $n_c = \sqrt{\frac{\mu}{a_c^3}}$: The chief's mean motion (rad/s), where $\mu = 3.986004418 \times 10^{14}\,\text{m}^3/\text{s}^2$ is Earth's gravitational parameter.
- $\eta = \sqrt{1 - e_c^2}$: A parameter related to the chief's eccentricity.
- $\kappa = \frac{3}{4}\frac{J_2 R^2 \sqrt{\mu}}{a_c^{7/2}\eta^4}$: A coefficient for $J_2$ perturbations, with $J_2 = 1.08262668 \times 10^{-3}$ (Earth's second zonal harmonic) and $R = 6378137\,\text{m}$ (Earth's equatorial radius).
- $E = 1 + \eta$: An auxiliary term.
- $F = 4 + 3\eta$: Another auxiliary term.
- $G = \frac{1}{\eta^2}$: Scales the eccentricity effects.
- $P = 3\cos^2 i_c - 1$: Inclination-dependent term.
- $Q = 5\cos^2 i_c - 1$: Another inclination-dependent term.
- $S = \sin(2i_c)$: Double-angle inclination term.
- $T = \sin^2 i_c$: Squared sine of inclination.
- $e_{xi} = e_c \cos\omega_c$, $e_{yi} = e_c \sin\omega_c$: Initial eccentricity vector components of the chief.
- $\dot{\omega} = \kappa Q$: The secular rate of change of the argument of perigee due to $J_2$ (rad/s).
- $\omega_f = \omega_c + \dot{\omega}\tau$: The argument of perigee after time $\tau$.
- $e_{xf} = e_c \cos\omega_f$, $e_{yf} = e_c \sin\omega_f$: Final eccentricity vector components.

We compute the STM at each time step using the chief's current orbital elements.

The control input matrix $\boldsymbol{B}_c(t)$ connects our control thrusts to changes in the ROE. It tells us how much each thrust component ($\Delta V_r$, $\Delta V_t$, $\Delta V_n$) affects the ROE. The matrix is:

$$\boldsymbol{B}_c(t) = \frac{1}{a_c n_c} \begin{bmatrix} \frac{2e_c \sin f_c}{\eta} & \frac{2(1+e_c \cos f_c)}{\eta} & 0 \\ -\frac{2\eta^2}{1+e_c \cos f_c} & 0 & 0 \\ \eta \sin(\omega_c + f_c) & \eta\left(\frac{(2+e_c \cos f_c)\cos(\omega_c+f_c)+e_{xc}}{1+e_c \cos f_c}\right) & \eta e_{yc} \frac{\sin(\omega_c+f_c)}{\tan i_c(1+e_c \cos f_c)} \\ -\eta \cos(\omega_c + f_c) & \eta\left(\frac{(2+e_c \cos f_c)\sin(\omega_c+f_c)+e_{yc}}{1+e_c \cos f_c}\right) & -\eta e_{xc} \frac{\sin(\omega_c+f_c)}{\tan i_c(1+e_c \cos f_c)} \\ 0 & 0 & \eta \frac{\cos(\omega_c+f_c)}{1+e_c \cos f_c} \\ 0 & 0 & \eta \frac{\sin(\omega_c+f_c)}{1+e_c \cos f_c} \end{bmatrix},$$

Where we've got a few more variables to define:

- $e_{xc} = e_c \cos \omega_c$, $e_{yc} = e_c \sin \omega_c$: The chief's eccentricity vector components, same as $e_{xi}$, $e_{yi}$ but relabeled for clarity.
- $f_c$: The chief's true anomaly at time $t$, obtained from the propagated orbit.
- $1 + e_c \cos f_c$: A term that adjusts for the orbit's shape at the current position.
- Other terms like $\eta$, $n_c$, $a_c$, $i_c$, and $\omega_c$ are the same as previously defined.

The factor $\dfrac{1}{a_c n_c}$ scales the matrix to match the chief's orbit size and speed, ensuring the control inputs have the correct effect on the deputy spacecraft dynamics.

The general solution to the continuous quasi-nonsingular ROE dynamics is:

$$\delta\overline{\boldsymbol{\alpha}}_d(t+\tau) = \Phi_{J2,qns}(\tau, t)\delta\overline{\boldsymbol{\alpha}}_d(t) + \int_t^{t+\tau} \Phi_{J2,qns}(t+\tau, s)\mathbf{B}_c(s)\mathbf{u}_d(s)\, ds \tag{76}$$

The discrete deputy's dynamics are derived from the continuous solution:

$$\delta\overline{\boldsymbol{\alpha}}_d(t_k) = \Phi_{J2,qns}(t_k - t_{k-1}, t_{k-1})\delta\overline{\boldsymbol{\alpha}}_d(t_{k-1}) + \int_{t_{k-1}}^{t_k} \Phi_{J2,qns}(t_k - s, s)\mathbf{B}_c(s)\mathbf{u}_d(s)\, ds \tag{77}$$

For $\Delta t = t_k - t_{k-1}$, assuming constant $\mathbf{u}_d$, and approximating $\Phi_{J2,qns}(t_k - s, s) \approx \mathbf{I}_6$ for small $\Delta t$, we obtain:

$$\mathbf{f}_{d,k} = \Phi_{J2,qns}(\Delta t, t_{k-1})\delta\overline{\boldsymbol{\alpha}}_{d,k-1} + \mathbf{B}_c(t_{k-1})\Delta t \mathbf{u}_{d,k} + \mathbf{w}_{d,k} \tag{78}$$

## 8.3 Measurement Model

### 8.3.1 Available Sensors On-Board

The spacecraft formation uses onboard sensors to support the estimation of the 12-dimensional state vector. [9]

The chief relies on absolute measurements to determine its inertial position and velocity, while the deputy uses relative measurements to maintain formation geometry at meter-to-millimeter scales with sub-millimeter accuracy.

The chief is equipped with a GNSS receiver providing:

- **Pseudorange:** Time-of-flight measurements with 10-50 cm accuracy (improved with differential GNSS).
- **Carrier-phase:** Milliliter-level accuracy after ambiguity resolution.
- **Range rate:** Doppler-based velocity with mm/s precision.

These are processed into absolute position and velocity in the ECI frame.

Both spacecraft carry an inter-satellite laser ranging system, providing:

- **Range:** Milliliter-level accuracy
- **Range rate:** Doppler-based velocity with 1 to 10 mm/s precision.

These measurements yield relative position, velocity, and ROE differences.

The deputy also has an optical camera for chief's azimuth and elevation measurements (arcsecond-level), which, combined with range data, enhances relative positioning .

### 8.3.2 Measurement Function

The nonlinear measurement model maps the state vector to sensor measurements, incorporating noise, to support the UKF's measurement update. The state, combining the chief's ECI position and velocity and the deputy's ROE, requires measurements reflecting both absolute and relative dynamics. The model is defined as $\mathbf{z}_k = \mathbf{h}_k(\mathbf{x}_k) + \mathbf{v}_k$, where $\mathbf{v}_k \sim \mathcal{N}(\mathbf{0}, \mathbf{R}_k)$ is measurement noise, and includes absolute position and velocity for the chief, and range and bearing angles for the deputy's relative state [10].

The measurement vector is:

$$\mathbf{z}_k = \begin{bmatrix} \mathbf{r}_{c,k} \\ \mathbf{v}_{c,k} \\ \|\mathbf{r}_{d,k} - \mathbf{r}_{c,k}\|_2 \\ \arctan\left(\dfrac{\Delta y_{d,k}}{\Delta x_{d,k}}\right) \\ \arcsin\left(\dfrac{\Delta z_{d,k}}{\|\Delta \mathbf{r}_{d,k}\|_2}\right) \end{bmatrix}, \quad \mathbf{h}_k(\mathbf{x}_k) = \begin{bmatrix} \mathbf{x}_{c,k}(1:3) \\ \mathbf{x}_{c,k}(4:6) \\ \sqrt{(\Delta \mathbf{r}_{d,k})^\top \Delta \mathbf{r}_{d,k}} \\ \arctan\left(\dfrac{\Delta y_{d,k}}{\Delta x_{d,k}}\right) \\ \arcsin\left(\dfrac{\Delta z_{d,k}}{\|\Delta \mathbf{r}_{d,k}\|_2}\right) \end{bmatrix} \tag{79}$$

where $\mathbf{r}_{c,k} = \mathbf{x}_{c,k}(1:3)$, $\mathbf{v}_{c,k} = \mathbf{x}_{c,k}(4:6)$, and $\Delta \mathbf{r}_{d,k} = \mathbf{r}_{d,k} - \mathbf{r}_{c,k}$, with $\mathbf{r}_{d,k}$ derived from the deputy's ROE.

The relative position is a nonlinear function of the deputy's ROE:

$$\Delta \mathbf{r}_{d,k} = \mathbf{g}(\delta\overline{\boldsymbol{\alpha}}_{d,k}, \mathbf{r}_{c,k}, \mathbf{v}_{c,k}) \tag{80}$$

where $\mathbf{g}$ transforms ROE to Cartesian coordinates using orbital mechanics, involving conversions from ROE to orbital elements and then to ECI coordinates.

### 8.4 Unscented Kalman Filter

The Unscented Kalman Filter (UKF) is employed to estimate the 12-dimensional state vector of the spacecraft formation, addressing the nonlinear dynamics and measurement models inherent in orbital navigation. Unlike the Extended Kalman Filter (EKF), which relies on linearization, the UKF uses the unscented transformation to propagate a set of sigma points through the true nonlinear functions, providing a robust and accurate method for state estimation.

#### 8.4.1 Mathematical Foundations

The UKF extends the Kalman filter framework to nonlinear systems by approximating the state distribution as Gaussian without requiring Jacobian computations. It operates by generating $2n+1$ sigma points, where $n$ is the state dimension, strategically placed to capture the mean and covariance of the state distribution. These points are propagated through the nonlinear state transition function $\mathbf{f}_k$ and measurement function $\mathbf{h}_k$, and their weighted statistics reconstruct the predicted and updated state distributions. This approach, known as the unscented transformation, avoids the EKF's first-order Taylor series approximation, which can introduce significant errors in highly nonlinear systems like orbital mechanics, where transformations from quasi-nonsingular relative orbital elements (ROE) to Cartesian coordinates involve trigonometric and square root functions.

The UKF is preferred over the EKF for several reasons:

- **Improved Accuracy**: By propagating sigma points through the true nonlinear functions, the UKF captures the mean and covariance up to at least the second order, reducing errors compared to the EKF's linearization, which is critical for the nonlinear ROE-to-ECI conversions in our measurement model.
- **No Jacobians Required**: The UKF eliminates the need to compute Jacobians of $\mathbf{f}_k$ and $\mathbf{h}_k$, simplifying implementation and avoiding numerical instability in complex dynamics like those involving $J_2$ perturbations.
- **Robustness to Nonlinearities**: The unscented transformation handles strong nonlinearities better, ensuring stable performance for our system's dynamics and measurements, which include range and bearing angles.
- **Computational Efficiency**: While requiring more function evaluations ($2n+1$ vs. EKF's single evaluation), the UKF's avoidance of derivative computations often results in comparable or lower computational cost for our 12-dimensional state.

The UKF algorithm consists of two main phases: prediction and update, executed recursively for each time step $k$.

1. **Prediction Step**:
   - Generate sigma points from the current state estimate $\hat{\mathbf{x}}_{k-1|k-1}$ and covariance $\mathbf{P}_{k-1|k-1}$:
   $$\boldsymbol{\chi}_{k-1,0} = \hat{\mathbf{x}}_{k-1|k-1}$$
   $$\boldsymbol{\chi}_{k-1,i} = \hat{\mathbf{x}}_{k-1|k-1} + \left(\sqrt{(n+\lambda)\mathbf{P}_{k-1|k-1}}\right)_i, \quad i = 1, \ldots, n$$
   $$\boldsymbol{\chi}_{k-1,i+n} = \hat{\mathbf{x}}_{k-1|k-1} - \left(\sqrt{(n+\lambda)\mathbf{P}_{k-1|k-1}}\right)_i, \quad i = 1, \ldots, n$$
   where $\sqrt{\cdot}$ is the Cholesky decomposition lower triangular matrix, and $\lambda$ is a scaling parameter.
   - Propagate sigma points through the dynamics:
   $$\boldsymbol{\chi}_{k,i} = \mathbf{f}_k(\boldsymbol{\chi}_{k-1,i}, \mathbf{u}_k, \mathbf{0})$$
   - Compute the predicted state mean and covariance:
   $$\hat{\mathbf{x}}_{k|k-1} = \sum_{i=0}^{2n} W_i^{(m)} \boldsymbol{\chi}_{k,i}$$
   $$\mathbf{P}_{k|k-1} = \sum_{i=0}^{2n} W_i^{(c)} (\boldsymbol{\chi}_{k,i} - \hat{\mathbf{x}}_{k|k-1})(\boldsymbol{\chi}_{k,i} - \hat{\mathbf{x}}_{k|k-1})^\top + \mathbf{Q}_k$$
2. **Update Step**:
   - Generate new sigma points from the predicted state:
   $$\boldsymbol{\chi}_{k,0} = \hat{\mathbf{x}}_{k|k-1}$$
   $$\boldsymbol{\chi}_{k,i} = \hat{\mathbf{x}}_{k|k-1} \pm \left(\sqrt{(n+\lambda)\mathbf{P}_{k|k-1}}\right)_i, \quad i = 1, \ldots, n$$
   - Propagate sigma points through the measurement model:
   $$\boldsymbol{\mathcal{Z}}_{k,i} = \mathbf{h}_k(\boldsymbol{\chi}_{k,i}, \mathbf{0})$$
   - Compute the predicted measurement mean:
   $$\hat{\mathbf{z}}_{k|k-1} = \sum_{i=0}^{2n} W_i^{(m)} \boldsymbol{\mathcal{Z}}_{k,i}$$
   - Compute the measurement covariance and cross-covariance:
   $$\mathbf{S}_k = \sum_{i=0}^{2n} W_i^{(c)} (\boldsymbol{\mathcal{Z}}_{k,i} - \hat{\mathbf{z}}_{k|k-1})(\boldsymbol{\mathcal{Z}}_{k,i} - \hat{\mathbf{z}}_{k|k-1})^\top + \mathbf{R}_k$$
   $$\mathbf{P}_{\mathbf{x}_k\mathbf{z}_k} = \sum_{i=0}^{2n} W_i^{(c)} (\boldsymbol{\chi}_{k,i} - \hat{\mathbf{x}}_{k|k-1})(\boldsymbol{\mathcal{Z}}_{k,i} - \hat{\mathbf{z}}_{k|k-1})^\top$$
   - Compute the Kalman gain:
   $$\mathbf{K}_k = \mathbf{P}_{\mathbf{x}_k\mathbf{z}_k} \mathbf{S}_k^{-1}$$
   - Update the state and covariance:
   $$\hat{\mathbf{x}}_{k|k} = \hat{\mathbf{x}}_{k|k-1} + \mathbf{K}_k(\mathbf{z}_k - \hat{\mathbf{z}}_{k|k-1})$$
   $$\mathbf{P}_{k|k} = \mathbf{P}_{k|k-1} - \mathbf{K}_k \mathbf{S}_k \mathbf{K}_k^\top$$

**Initialization**: Start with initial state estimate $\hat{\mathbf{x}}_{0|0}$ and covariance $\mathbf{P}_{0|0}$.

### 8.4.2 Practical Implementation Parameters

The UKF implementation for our spacecraft formation uses the following parameters, chosen to balance accuracy, stability, and computational efficiency, as defined in the MATLAB code:

- **State and Measurement Dimensions**:

$$n = 12, \quad m = 8$$

  *Justification*: The state dimension $n = 12$ corresponds to the 6-dimensional chief's ECI coordinates and 6-dimensional deputy's ROE. The measurement dimension $m = 8$ reflects the chief's position and velocity (6 components), plus the deputy's relative range and bearing angles (azimuth and elevation, 2 components).

- **Scaling Parameters**:

$$\alpha = 0.6, \quad \kappa = 0, \quad \beta = 2, \quad \lambda = \alpha^2(n+\kappa) - n =$$

  *Justification*: $\alpha = 0.6$ controls the spread of sigma points, chosen small to keep points close to the mean, suitable for the nonlinear ROE transformations. $\kappa = 0$ ensures the sigma points are appropriately scaled for a high-dimensional state, though negative $\kappa$ is acceptable in UKF. $\beta = 2$ is optimal for Gaussian distributions, capturing higher-order moments. $\lambda$ is computed to scale the weights, balancing the contribution of the mean point.

- **Weights**:

$$W_0^{(m)} = \frac{\lambda}{n+\lambda}, \quad W_0^{(c)} = W_0^{(m)} + (1 - \alpha^2 + \beta)$$
$$W_i^{(m)} = W_i^{(c)} = \frac{1}{2(n+\lambda)} \approx \frac{1}{2 \times 0.03} \approx 16.67, \quad i = 1, \dots, 24$$

  *Justification*: The weights are derived from $\lambda$, ensuring the sigma points correctly represent the mean and covariance. The negative $W_0^{(m)}$ and $W_0^{(c)}$ are typical for negative $\lambda$, with higher weights on other points compensating to maintain accuracy.

- **Process Noise Covariance**:

$$\mathbf{Q}_k = \text{diag}([0.01^2, 0.01^2, 0.01^2, 0.001^2, 0.001^2, 0.001^2, 10^{-12}, 10^{-12}, 10^{-12}, 10^{-12}, 10^{-12}, 10^{-12}])$$

  where $\sigma_{\text{pos,ECI}} = 10^{-2}\,\text{m}$, $\sigma_{\text{vel,ECI}} = 10^{-3}\,\text{m/s}$, $\sigma_{\text{ROE,dimless}} = 10^{-6}$, $\sigma_{\text{ROE,angle}} = 10^{-6}\,\text{rad}$.
  *Justification*: The chief's position and velocity noise levels reflect expected modeling errors in ECI dynamics, accounting for unmodeled perturbations. The ROE noise levels are small, as the deputy's dynamics are modeled accurately via the STM, but non-zero to prevent covariance collapse and ensure filter stability.

- **Measurement Noise Covariance**:

$$\mathbf{R}_k = \text{diag}([10^{-6}, 10^{-6}, 10^{-6}, 10^{-6}, 10^{-6}, 10^{-6}, 10^{-6}, (4.848 \times 10^{-3})^2, (4.848 \times 10^{-3})^2])$$

  where $\sigma_{\text{pos}} = 10^{-3}\,\text{m}$, $\sigma_{\text{vel}} = 10^{-3}\,\text{m/s}$, $\sigma_{\text{range}} = 10^{-3}\,\text{m}$, $\sigma_{\text{angles}} = 4.848 \times 10^{-3}\,\text{rad}$.
  *Justification*: These values reflect sensor accuracies: GNSS provides 10 cm position and 1 mm/s velocity precision, laser ranging offers 5 mm range accuracy, and camera angles achieve 10 arcseconds ($\approx 4.848 \times 10^{-6}\,\text{rad}$). These are conservative estimates to account for potential biases and environmental noise, ensuring robust measurement updates.

- **Initial State and Covariance**:
$$\hat{\mathbf{x}}_{0|0} = \begin{bmatrix}\mathbf{x}_{c,0}^\top & \delta\overline{\boldsymbol{\alpha}}_{d,0}^\top\end{bmatrix}^\top$$
where $\mathbf{x}_{c,0}$ is the chief's initial ECI state derived from orbital elements ($a_c = 6771000\,\mathrm{m}$, $e_c = 0.0005$, etc.), and
$$\delta\overline{\boldsymbol{\alpha}}_{d,0} = \begin{bmatrix}0 & -0.0012 & 0.0007 & 0.0007 & 0.0087 & 0.0137\end{bmatrix}^\top$$
The initial covariance matrix is:
$$\mathbf{P}_{0|0} = \begin{bmatrix}0.1\,\mathbf{I}_6 & \mathbf{0}_{6\times6} \\ \mathbf{0}_{6\times6} & 0.01\,\mathbf{I}_6\end{bmatrix}$$
*Justification*: The initial state combines the chief's absolute ECI state and the deputy's relative orbital elements (ROEs). Although the chief's ECI state is computed from precise orbital elements, the larger magnitude of position and velocity values in ECI space leads to higher numerical uncertainty and reduced filter sensitivity. In contrast, the ROEs represent small relative deviations in formation geometry (e.g., $\delta\lambda_d = -3.3773°$), which are better scaled for estimation and yield more stable filter behavior. Hence, the covariance $\mathbf{P}_{0|0}$ assigns greater uncertainty to the chief (0.1 $\mathrm{m}^2$, 0.1 $\mathrm{m}^2/\mathrm{s}^2$) and lower uncertainty (0.01) to the deputy's ROEs, facilitating better convergence in the

These parameters ensure the UKF effectively estimates the spacecraft formation's state, leveraging the unscented transformation to handle nonlinearities while maintaining computational feasibility and robustness.

## 8.5 Results

We obtain the following results.

### 8.5.1 Chief

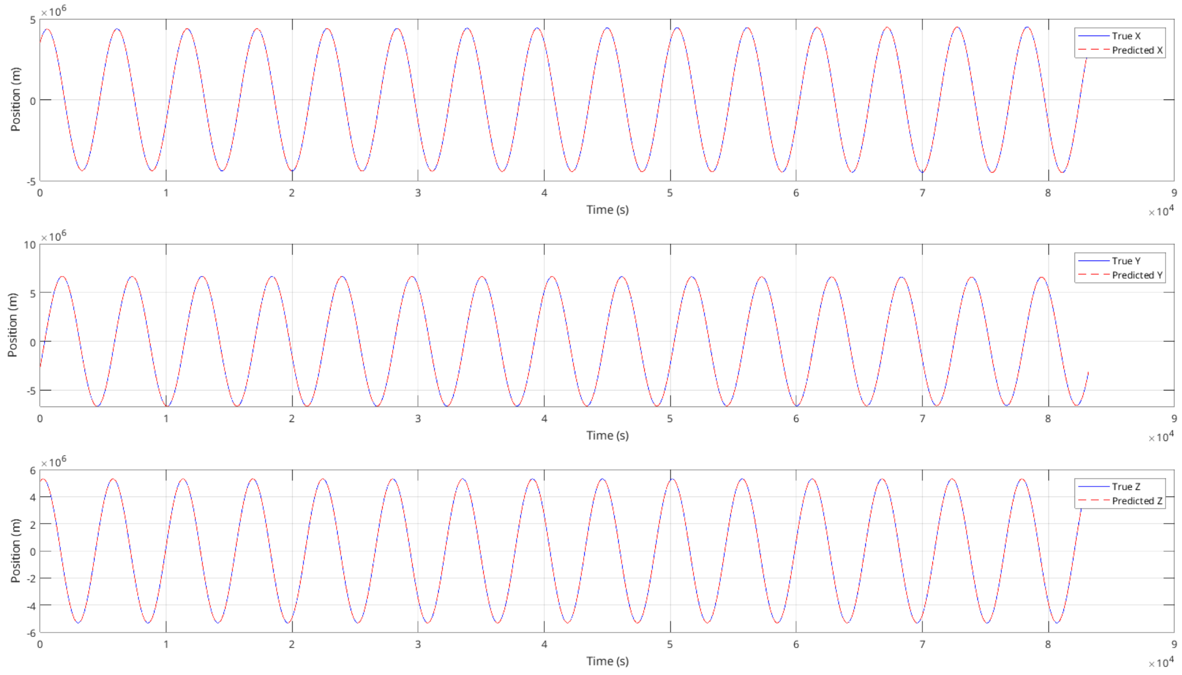


Figure 8.1: Chief's true and estimated position in ECI over time

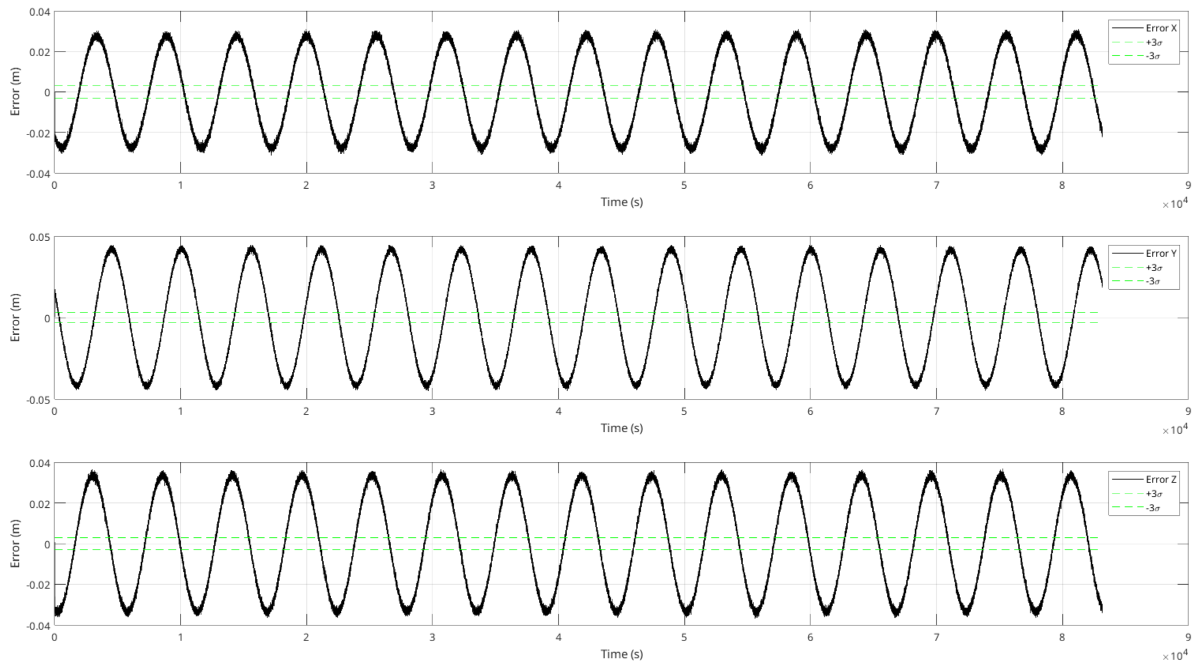


Figure 8.2: Chief's position in ECI error from UKF vs. ground truth with $\pm 3\sigma$ bounds

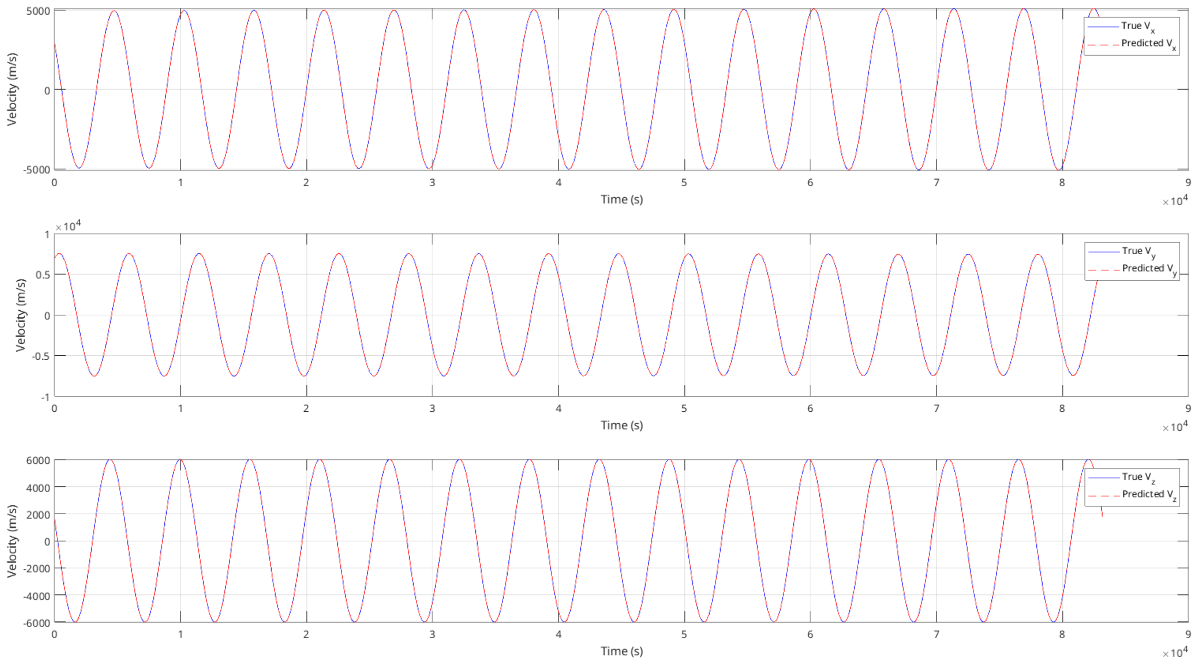


Figure 8.3: Chief's true and estimated velocity in ECI over time

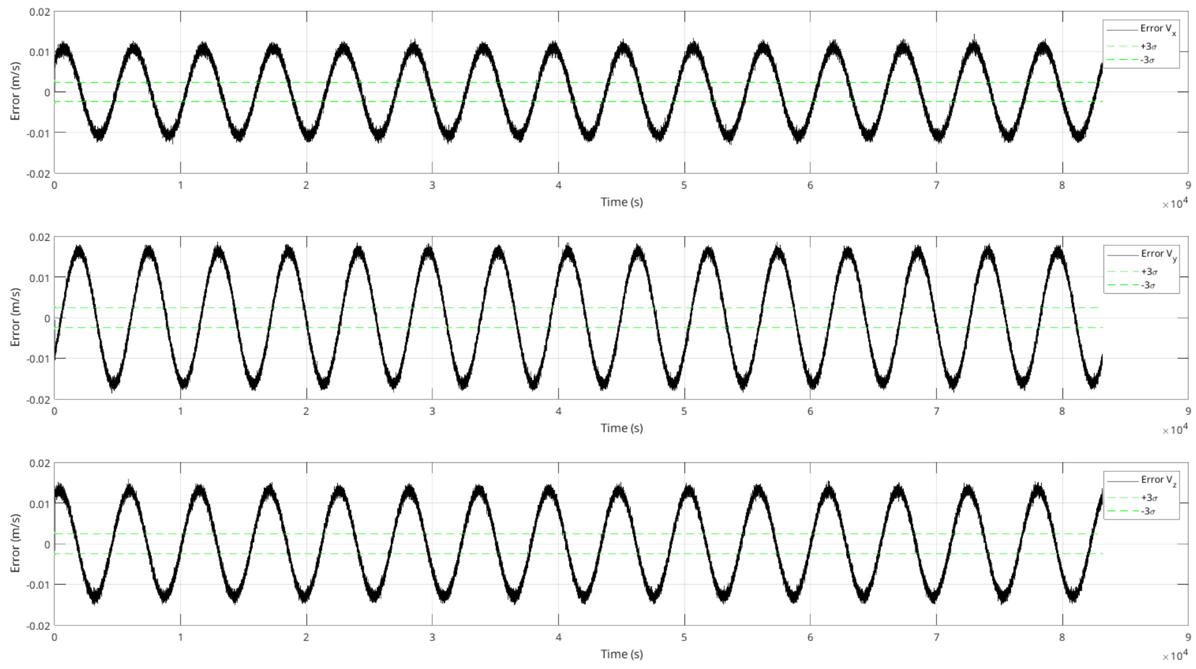


Figure 8.4: Chief's velocity in ECI error from UKF vs. ground truth with $\pm 3\sigma$ bounds

### 8.5.2 Deputy

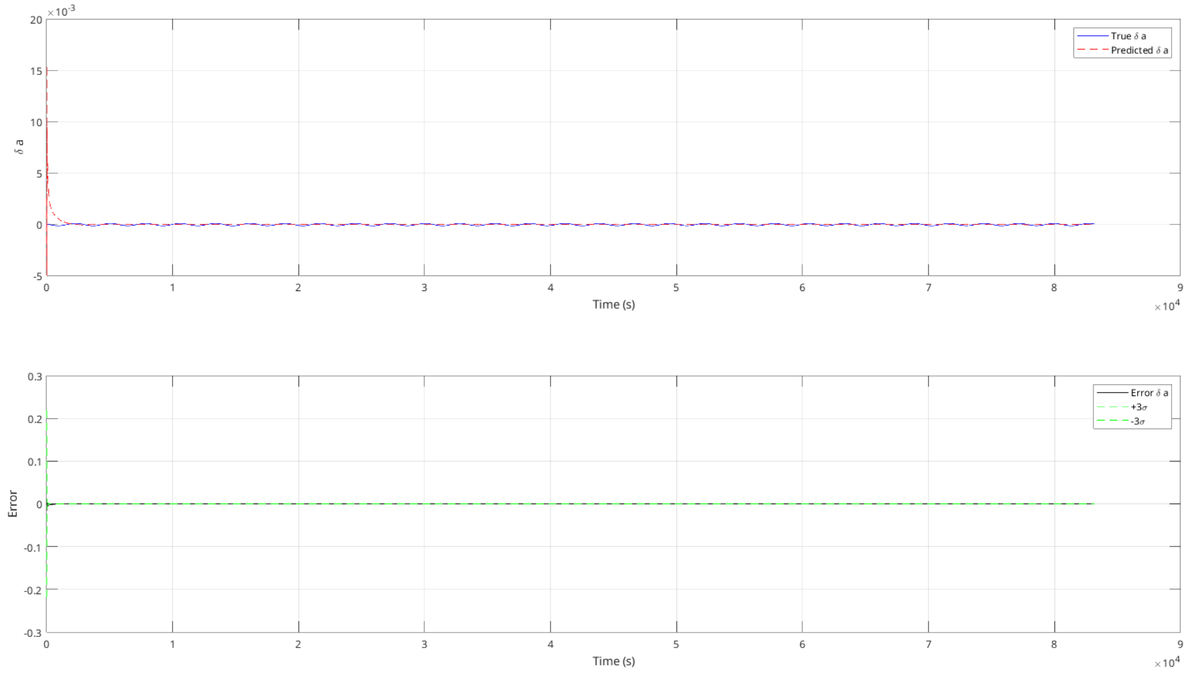


Figure 8.5: Deputy's true and estimated QNSROE $\delta a$ as a function of time. Also deputy's QNSROE $\delta a$ error from UKF vs. ground truth with $\pm 3\sigma$ bounds

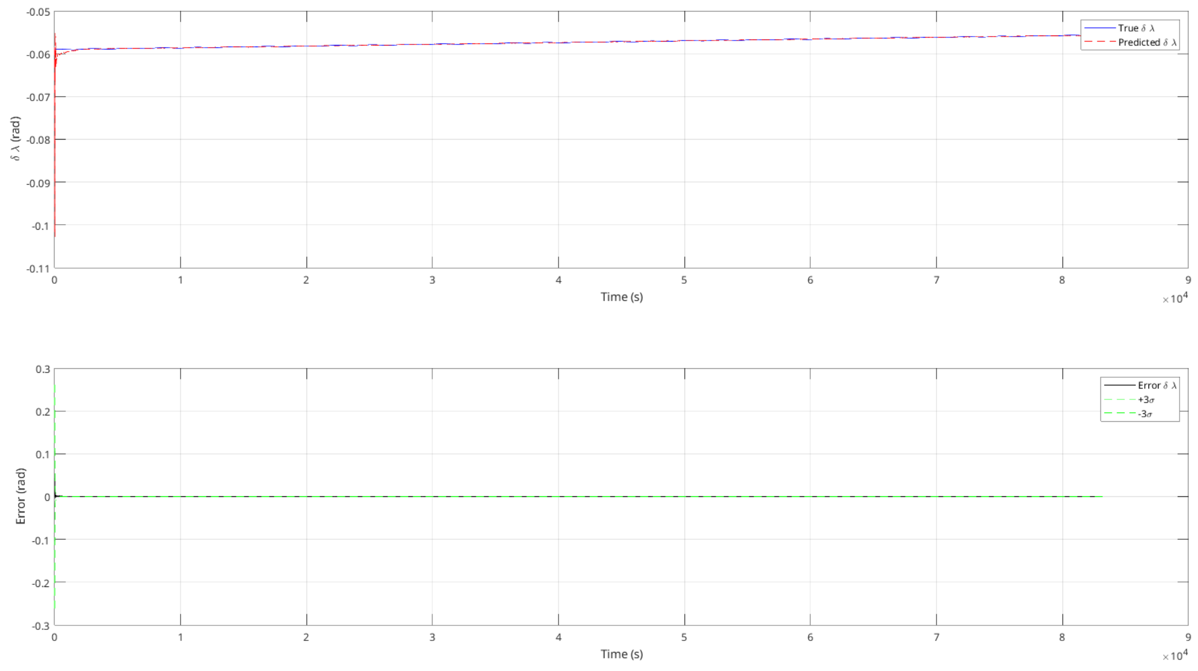


Figure 8.6: Deputy's true and estimated QNSROE $\delta\lambda$ as a function of time. Also deputy's QNSROE $\delta\lambda$ error from UKF vs. ground truth with $\pm 3\sigma$ bounds

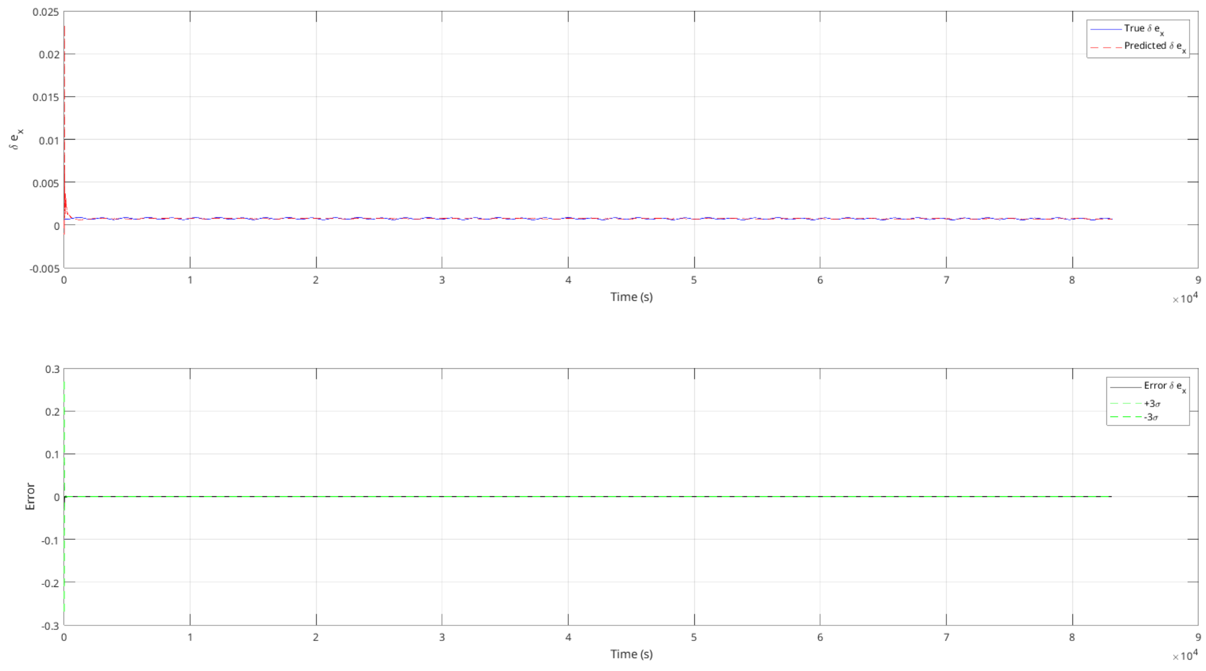
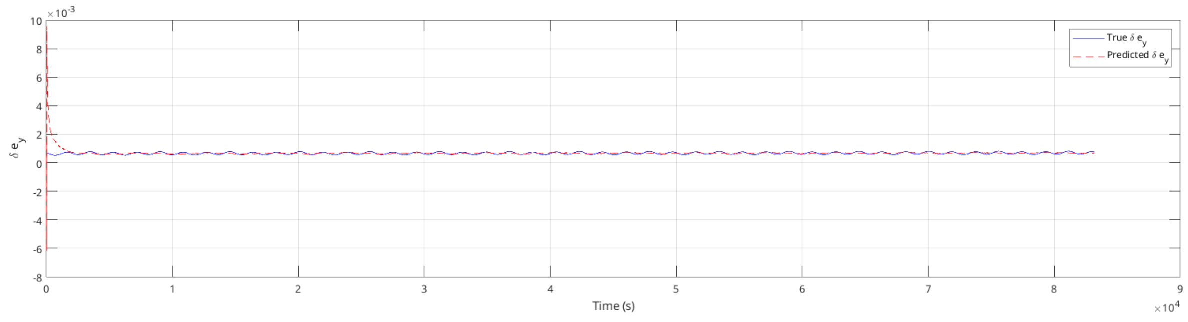


Figure 8.7: Deputy's true and estimated QNSROE $\delta e_x$ as a function of time. Also deputy's QNSROE $\delta e_x$ error from UKF vs. ground truth with $\pm 3\sigma$ bounds



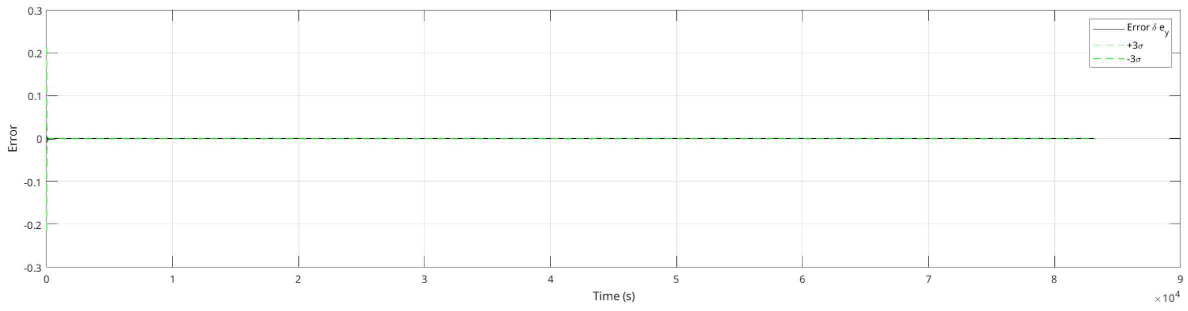


Figure 8.8: Deputy's true and estimated QNSROE $\delta e_y$ as a function of time. Also deputy's QNSROE $\delta e_y$ error from UKF vs. ground truth with $\pm 3\sigma$ bounds

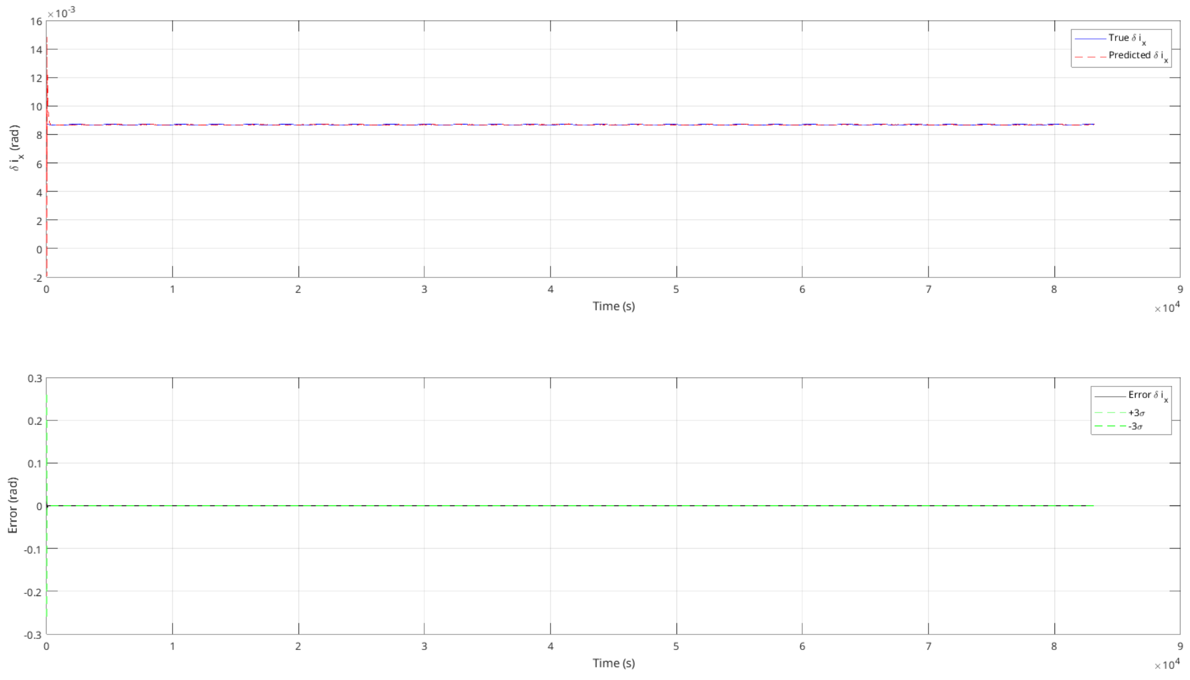


Figure 8.9: Deputy's true and estimated QNSROE $\delta i_x$ as a function of time. Also deputy's QNSROE $\delta i_x$ error from UKF vs. ground truth with $\pm 3\sigma$ bounds

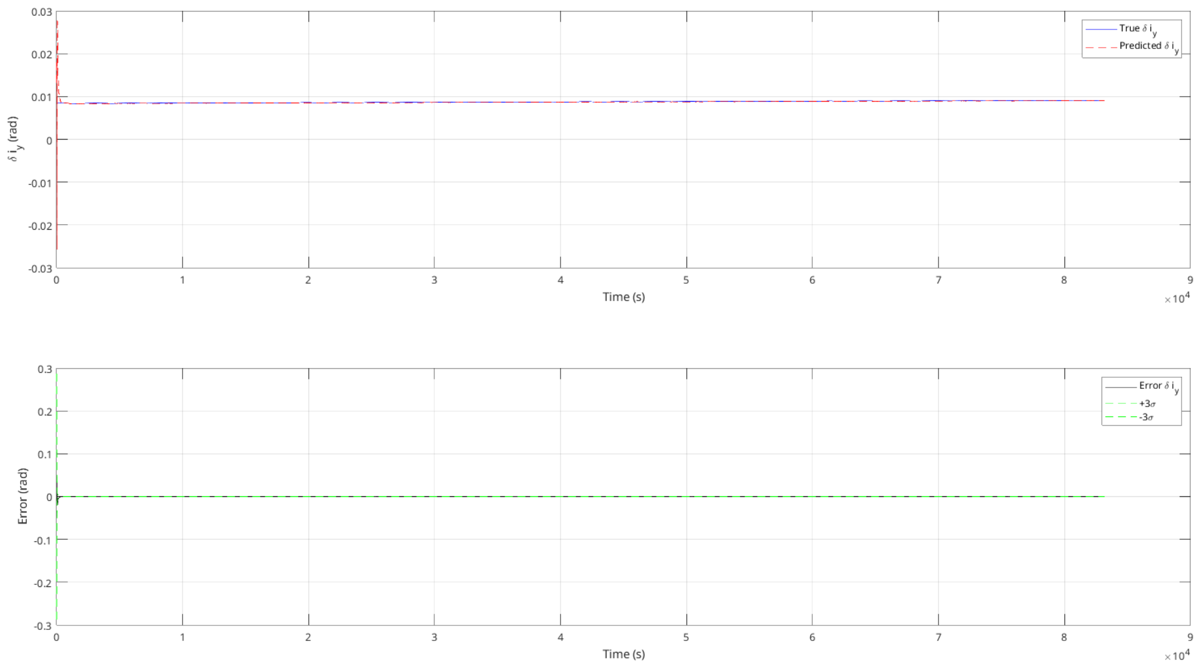


Figure 8.10: Deputy's true and estimated QNSROE $\delta i_y$ as a function of time. Also deputy's QNSROE $\delta i_y$ error from UKF vs. ground truth with $\pm 3\sigma$ bounds

### 8.5.3 Measurements Residuals

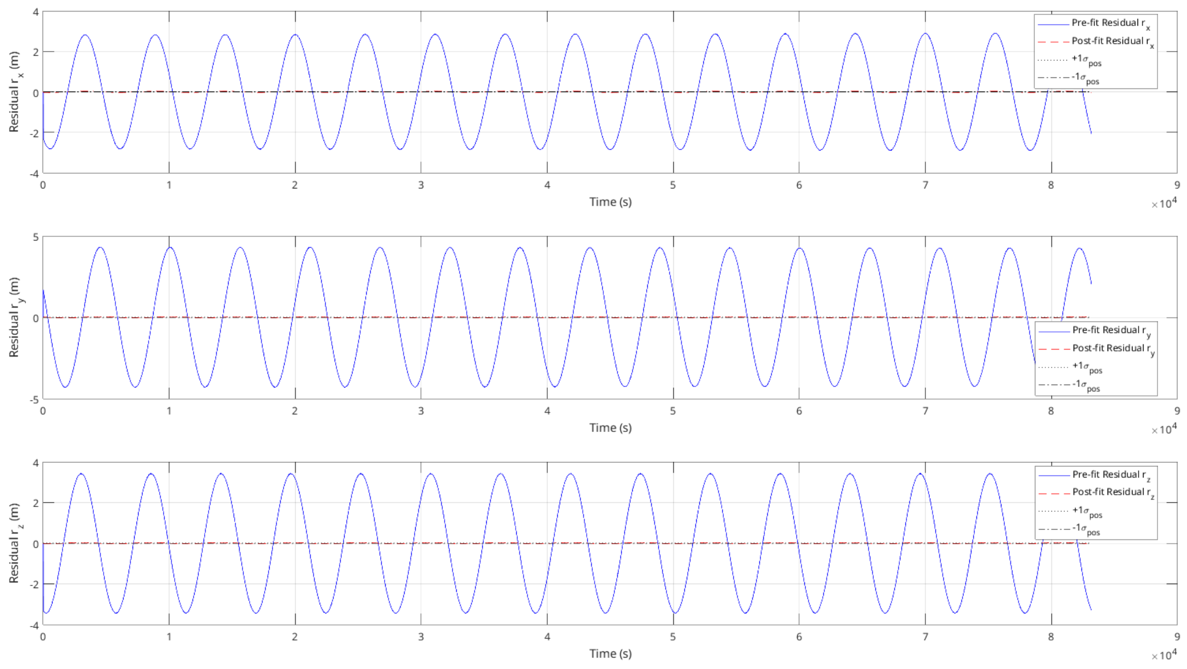


Figure 8.11: Pre-fit (blue) and post-fit (red) residuals for the relative position components $r_x$, $r_y$, and $r_z$, with $\pm 1\sigma_{\text{pos}}$ bounds shown in black.

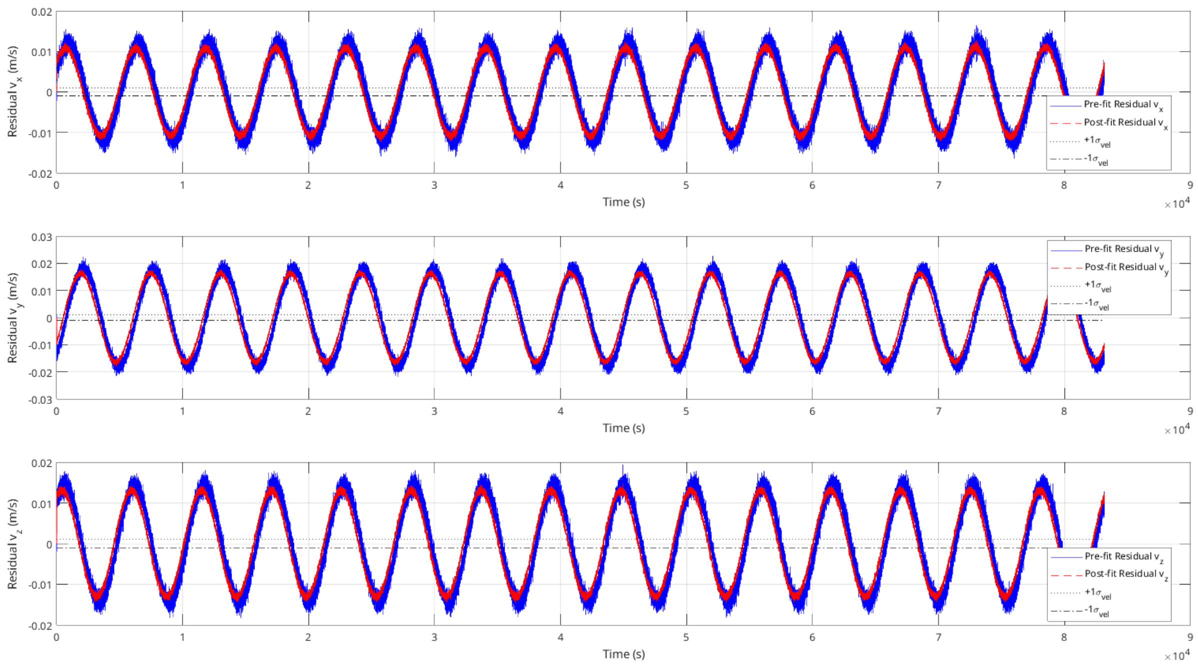


Figure 8.12: Pre-fit (blue) and post-fit (red) residuals for the relative velocity components $v_x$, $v_y$, and $v_z$, with $\pm 1\sigma_{\text{vel}}$ bounds shown in black.

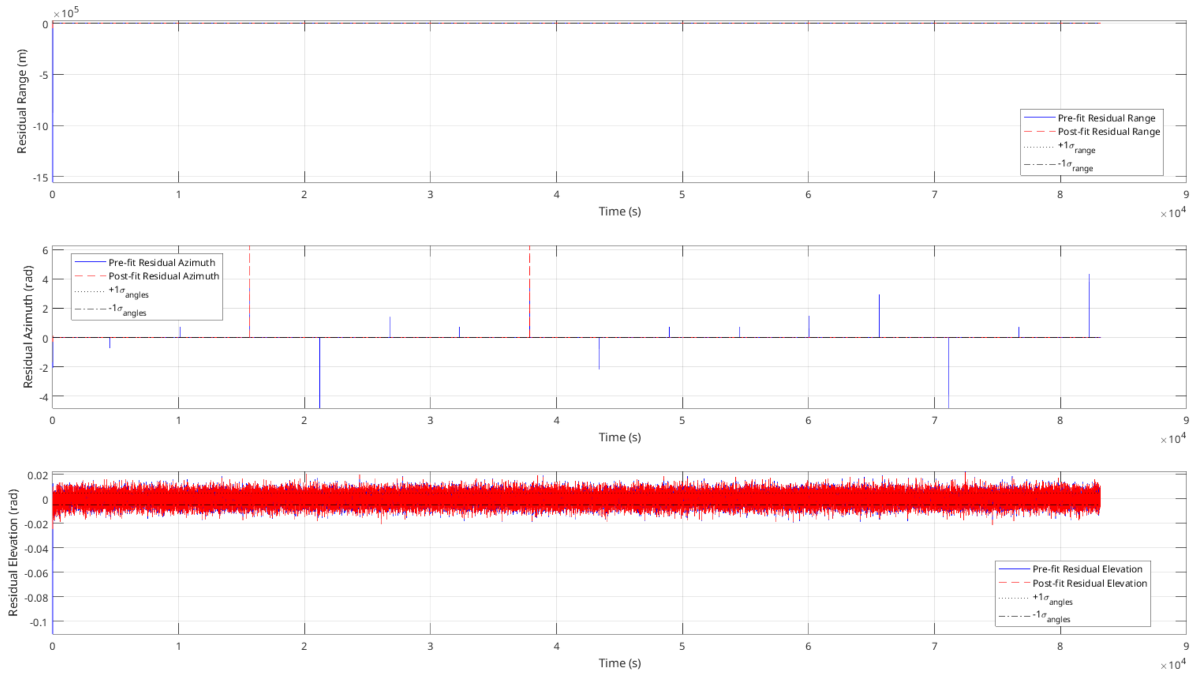
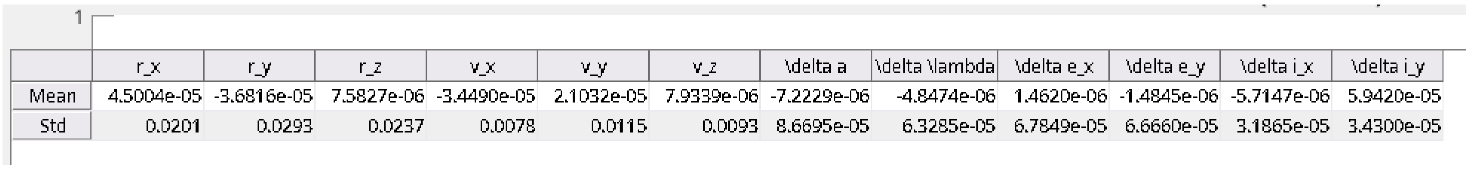


Figure 8.13: Pre-fit (blue) and post-fit (red) residuals for the deputy's range, azimuth, and elevation measurements, with $\pm 1\sigma_{\text{meas}}$ bounds shown in black.

### 8.5.4 Final Statistics



| | r_x | r_y | r_z | v_x | v_y | v_z | \delta a | \delta \lambda | \delta e_x | \delta e_y | \delta i_x | \delta i_y |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Mean | 4.5004e-05 | -3.6816e-05 | 7.5827e-06 | -3.4490e-05 | 2.1032e-05 | 7.9339e-06 | -7.2229e-06 | -4.8474e-06 | 1.4620e-06 | -1.4845e-06 | -5.7147e-06 | 5.9420e-05 |
| Std | 0.0201 | 0.0293 | 0.0237 | 0.0078 | 0.0115 | 0.0093 | 8.6695e-05 | 6.3285e-05 | 6.7849e-05 | 6.6660e-05 | 3.1865e-05 | 3.4300e-05 |

Figure 8.14: Mean and standard deviation of the UKF state-estimation errors for all components (position, velocity, and deputy's QNSROE parameters) over the final orbit.

### 8.5.5 Conclusion

The UKF demonstrates highly effective state estimation for the spacecraft formation, with results indicating robust convergence to the true state over time. As shown in the simulation results, the estimated chief's ECI position and speed as well as the deputy's quasi-nonsingular ROE closely track the ground truth, with errors in all state components converging to near zero.

This confirms UKF's ability to leverage noisy sensor measurements (GNSS, laser ranging, and camera angles) to achieve accurate and stable estimates, validating its suitability for precise formation control in orbital missions.

## 9 Problem Set 9

The code used for this assignment is available in our GitHub repository [1] and the Appendix. See ”Problem Set 9 Code” for code for each section. Additionally, see ”Propagators”, ”Control Helpers” and ”Utils” sections.

## 9.1 Catching Up

Below is a list of all modifications we made from the previous submission.

| Rev | Changes |
|---|---|
| PS1 | - Edited all sections of question 1 to reflect mission changes<br>- Completed question 2 (previously incomplete due to different mission)<br>- Added code to Appendix |
| PS2 | - Redid all questions and figures with corrected nonlinear propagator<br>- Previous results were invalid due to propagator malfunction<br>- Added code to Appendix |
| PS3 | - Updated STM in 2c) for YA matrix to be more accurate<br>- Consolidated code in Appendix |
| PS4 | - Q1: Updated orbital elements to reflect new maneuver<br>- Q3: Redid entirely with corrections<br>- Q4 + Q5: Updated plots with analysis<br>- Q6: Redid entirely<br>- Q7: Updated plots<br>- Q8: Added second plot and re-plotted with corrected STM<br>- Added code to Appendix |
| PS5 | - Q1: Rewrote "Orbit Modeling Dynamics Approach" section<br>- Q2: Completely redid analysis (previously submitted in PS7)<br>- Q2: Rewrote "Performance and Results" section |
| PS6 | - Q2: Updated orbital elements values and format (QNSROE) to reflect new maneuver<br>- Q2: Deleted simplified Lyapunov model and replaced with formal, nonlinear definition<br>- Q2: Updated graphs with corrected Lyapunov control law<br>- Q2: Updated results analysis |
| PS7 | - Q2: Updated EKF sensitivity matrix to account for long range QNSROE estimation |
| PS8 | - Q2: Corrected UKF navigation filter formulation<br>- Q2: Corrected variable values with latest<br>- Q2: Updated existing graphs with corrected UKF implementation<br>- Q2: Added error graphs with $\pm 3\Sigma$ bounds<br>- Q2: Added measurement residual graphs<br>- Q2: Added navigation statistic table<br>- Q2: Updated results analysis |
| PS9 | - Added PS9 material and code for all questions |
| Appendix | - Added 'Utils (Conversion Functions)', 'Propagation Functions', and 'Control Code Helper Functions' to Appendix for functions used persistently throughout Problem Sets<br>- Reorganized code by problem set and added all code for each problem set<br>- Created new GitHub link with organized files |
| References | - Added academic references used across PS11 to PS8<br>- Linked to respective PSETS sections where used |

## 9.2 Integration of Navigation and Control

The goal is to design a control law that drives the deputy satellite's QNSROE to a desired state, accounting for $J_2$ perturbations. Since we do not have access to the true state of either the deputy or the chief satellite, their states are estimated via an unscented Kalman filter (UKF).

### 9.2.1 Control Law Selection

As explained in the previous assignment, we will use the Lyapunov-based continuous control law, as developed in Problem Set 6. The UKF state estimation is outlined in Problem Set 8.

### 9.2.2 Objectives

Let's recap one last time the initial states of the chief and deputy - respectively in the ECI frame and the deputy's RTN frame - as well as the desired final deputy QNSROE.

#### Mathematical Foundations

We model the deputy satellite's motion relative to the chief using a 6-state QNSROE vector, which is well-suited for near-circular orbits.

The QNSROE state is defined as:

$$\delta\overline{\boldsymbol{\alpha}} = \begin{bmatrix} \delta a_d \\ \delta\lambda_d \\ \delta e_{x,d} \\ \delta e_{y,d} \\ \delta i_{x,d} \\ \delta i_{y,d} \end{bmatrix},$$

where each component is defined as follows:

- $\delta a_d = \dfrac{a_d - a_c}{a_c}$: Relative semi-major axis, normalized by the chief's semi-major axis $a_c$ (dimensionless).
- $\delta\lambda_d = M_d + \omega_d + \Omega_d \cos i_c - (M_c + \omega_c + \Omega_c \cos i_c)$: Relative mean longitude, combining mean anomaly $M$, argument of perigee $\omega$, and right ascension of the ascending node $\Omega$ (radians).
- $\delta e_{x,d} = e_d \cos\omega_d - e_c \cos\omega_c$: $x$-component of the relative eccentricity vector (dimensionless).
- $\delta e_{y,d} = e_d \sin\omega_d - e_c \sin\omega_c$: $y$-component of the relative eccentricity vector (dimensionless).
- $\delta i_{x,d} = i_d - i_c$: Difference in inclination between the deputy and the chief (radians).
- $\delta i_{y,d} = (\Omega_d - \Omega_c)\sin i_c$: $y$-component of the relative inclination vector (radians).

Here, subscripts $c$ and $d$ denote the chief and deputy satellites, respectively.

**Numerical Values**

The chief's OEs and deputy's QNSROEs are the same as PS5, defined below.

**Chief's Initial Orbital Elements:**

$$\boldsymbol{\alpha}_c = \begin{bmatrix} a_c \\ e_c \\ i_c \\ \Omega_c \\ \omega_c \\ \nu_c \end{bmatrix} = \begin{bmatrix} 6771 \times 10^3 \text{ m} \\ 5.0 \times 10^{-4} \\ 51.64^\circ \\ 257^\circ \\ 45^\circ \\ 30^\circ \end{bmatrix}$$

**Deputy's Initial Quasi-Non-Singular Relative Orbital Elements:**

$$\delta\boldsymbol{\alpha}_{\text{initial}} = \begin{bmatrix} \delta a_d \\ \delta\lambda_d \\ \delta e_{x,d} \\ \delta e_{y,d} \\ \delta i_{x,d} \\ \delta i_{y,d} \end{bmatrix} = \begin{bmatrix} 0 \\ -3.3773^\circ \\ 0.0007 \\ 0.0007 \\ 0.4989^\circ \\ 0.7850^\circ \end{bmatrix}$$

**Deputy's Final Quasi-Non-Singular Relative Orbital Elements:**

$$\delta\boldsymbol{\alpha}_{\text{desired}} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}$$

### 9.2.3 Noisy Ground Truth

**Setup**

To monitor the impact of noisy inputs on the continuous control law, we generate the chief's ground truth including $J_2$ perturbations and then add Gaussian-distributed noise using the same standard deviations as those used for the UKF process noise on the chief's ECI position and velocity.

Meaning:

- $\sigma_{\text{pos,ECI}} = 10^{-2}\,\text{m}$
- $\sigma_{\text{vel,ECI}} = 10^{-3}\,\text{m/s}$

These noise levels reflect expected modeling errors in the chief's ECI dynamics (unmodeled perturbations). We omit the full process-noise matrix here since only position and velocity noise are relevant for our analysis.

## Results

We obtain the following results:

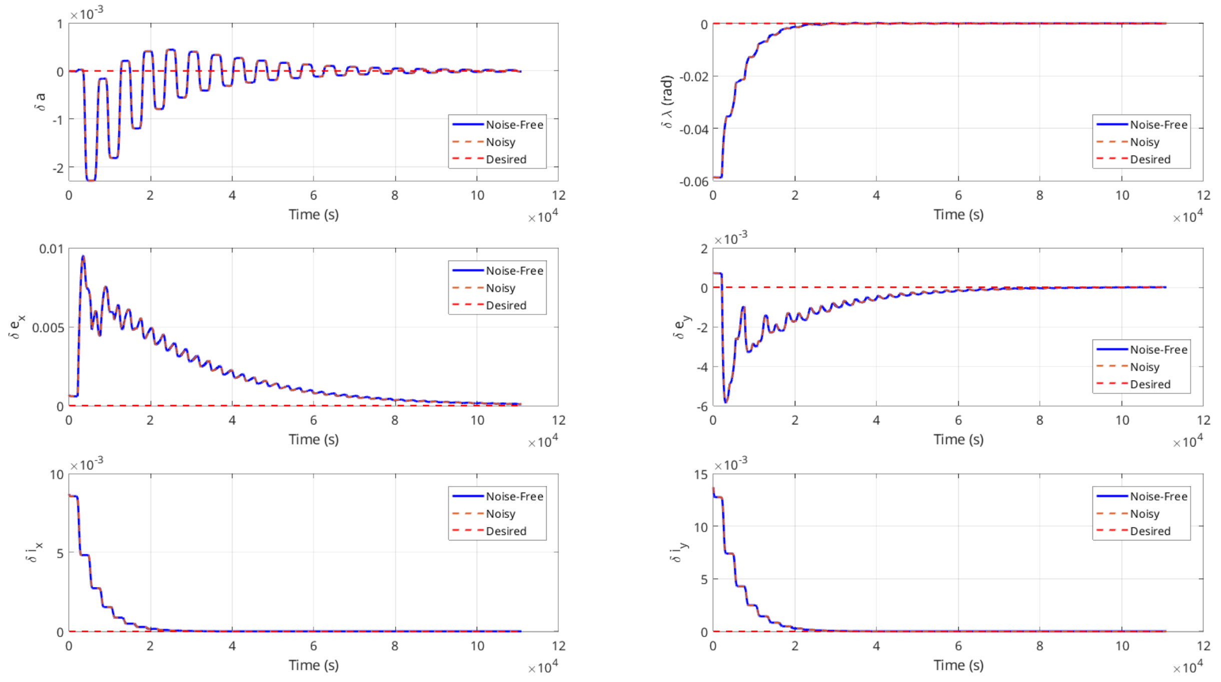


Figure 9.1: Evolution of deputy's relative quasi non singular orbital elements over time.

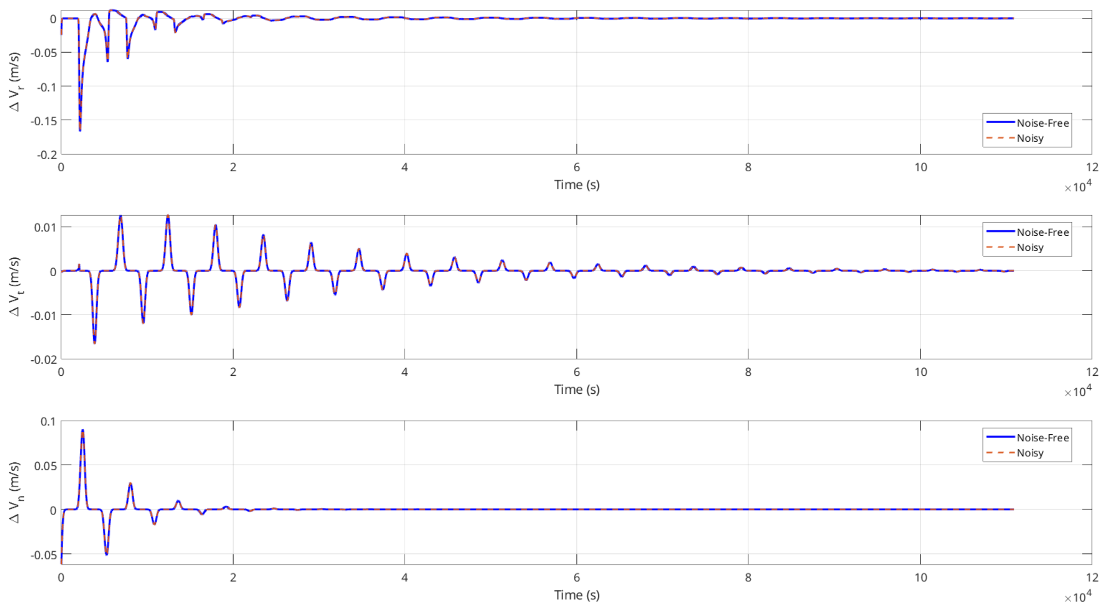


Figure 9.2: Evolution of the continuous impulse applied in RTN over time.

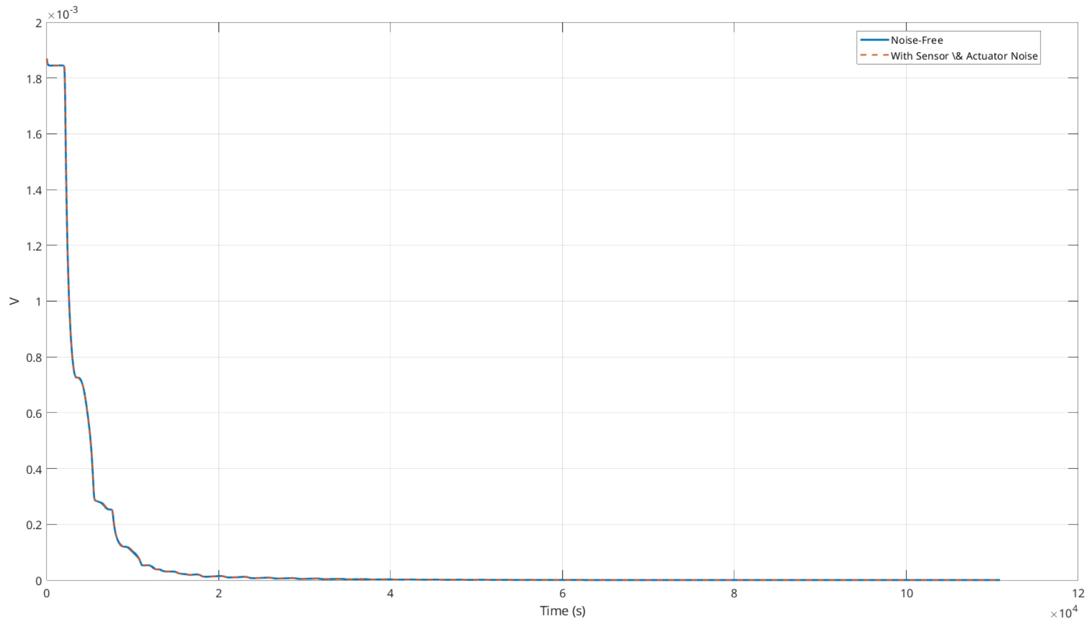


Figure 9.3: Evolution of the Lyapunov function over time.

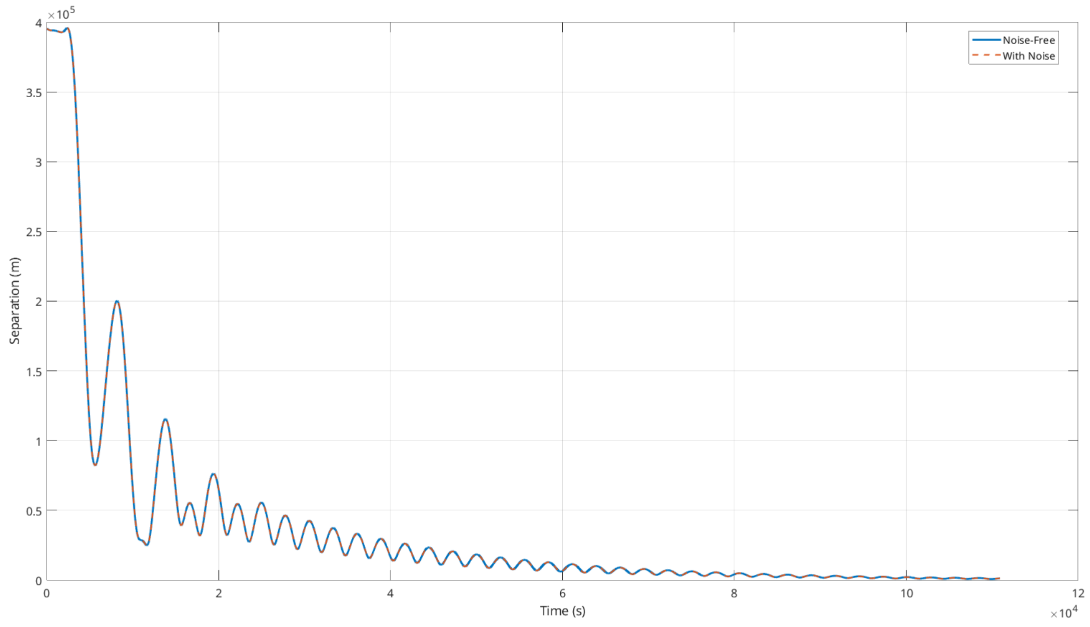


Figure 9.4: Evolution of the separation between the chief and the deputy spacecraft over time.

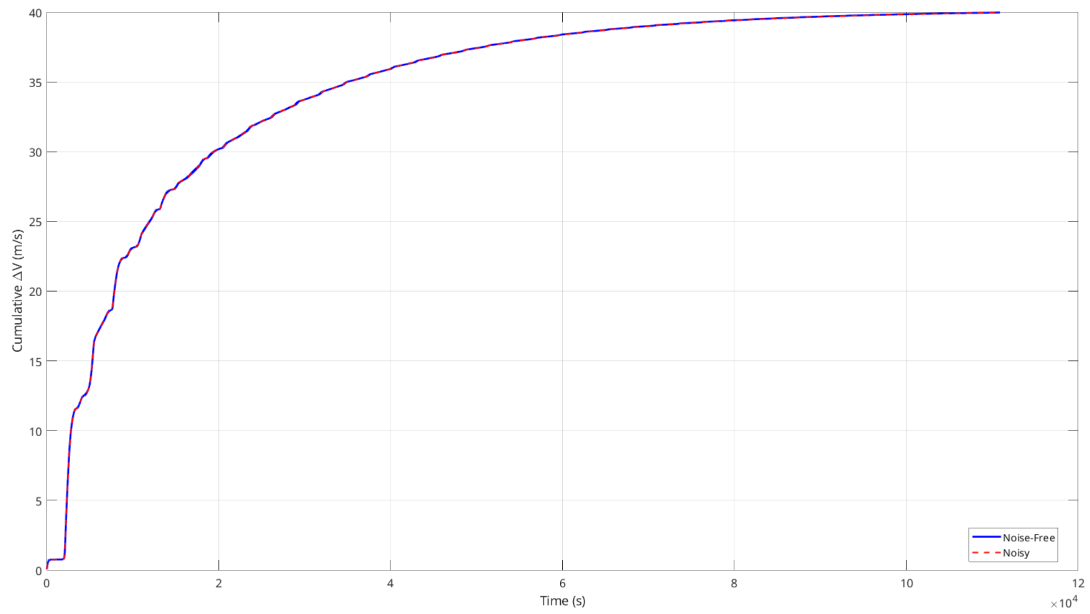


Figure 9.5: Evolution of the cumulative applied $\Delta V$ in the deputy's RTN frame over time.

The total final noisy cumulative applied $\Delta V_{tot_f}$ is 40.2 m/s.

**Conclusions**

Adding noise to the chief's ground truth-used to generate the control-law $A_c(t)$ and $B_c(t)$ matrices as explained in Pset 6-has only a minimal effect on the Lyapunov control law output. The deputy still converges in approximately the same amount of time (i.e., 15 orbits) to the desired final QNSROE. The total cumulative $\Delta V$ in the RTN frame is slightly higher than in the previous noiseless results, by about 1 m/s.

### 9.2.4 UKF Enabled Control

**Setup**

The integration of the UKF with the Lyapunov-based continuous control law enables precise state estimation and control for the deputy satellite, driving its QNSROE to the desired state under $J_2$ perturbations. The UKF estimates the chief and deputy states, feeding these into the Lyapunov control law to compute thrust commands in the deputy's RTN frame. To optimize fuel efficiency, control is delayed until the UKF estimates are reliable, using a convergence criterion based on state estimate stability.

**Methodology**

In the UKF's prediction step, sigma points - representing the state estimate and its uncertainty - are propagated through the nonlinear dynamics model. For each sigma point, the control input $\mathbf{u}_k$ is computed using the Lyapunov control law to align the deputy's QNSROE with the desired state. The process starts by extracting the chief's ECI state and deputy's QNSROE from the sigma point, then computing the chief's orbital elements to contextualize the relative motion.

The state transition matrix $\mathbf{A}_c$, derived from the $J_2$-perturbed dynamics, models the natural evolution of the QNSROE. The control input matrix $\mathbf{B}_c$, based on the chief's orbit, maps RTN thrusts to QNSROE changes. The error between the current and desired QNSROE drives the control, modulated by a fuel-optimizing matrix $\mathbf{P}$, which aligns thrusts with efficient orbital positions. The control input balances the QNSROE's natural drift (via $\mathbf{A}_c$) and corrective action (via $\mathbf{P}$), with $\mathbf{B}_c$ translating this into thrusts. These inputs are applied to each sigma point's deputy dynamics, ensuring controlled motion in the predicted state.

To enhance fuel efficiency, control is activated only when the UKF's state estimates converge, defined as the mean updated state changing by less than 10% from the previous step for 20 consecutive steps. This ensures reliable estimates, avoiding wasteful thrusts due to early uncertainties. The combination of UKF and Lyapunov control creates a special case: the filter's ability to refine estimates over time enables precise control, but only after sufficient convergence, balancing accuracy and efficiency.

### 9.2.5 Results and Key Metrics

We obtain the following results.

## Chief

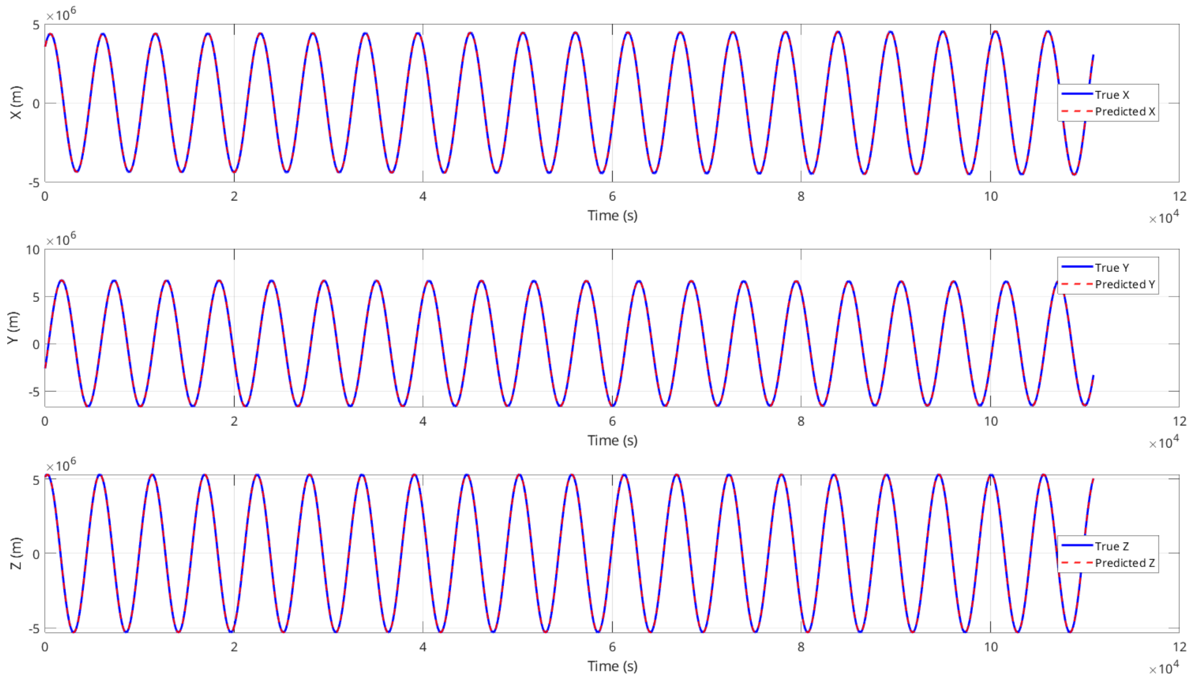


Figure 9.6: Chief's true and estimated position in ECI over time

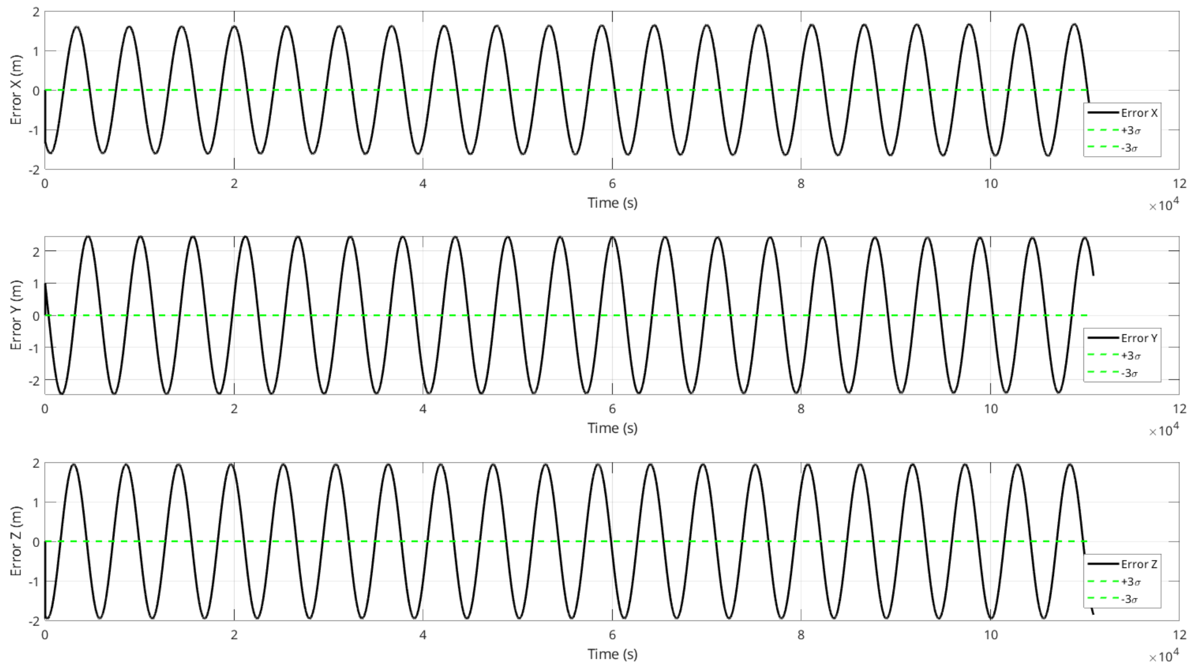


Figure 9.7: Chief's position in ECI error from UKF vs. ground truth with $\pm 3\sigma$ bounds

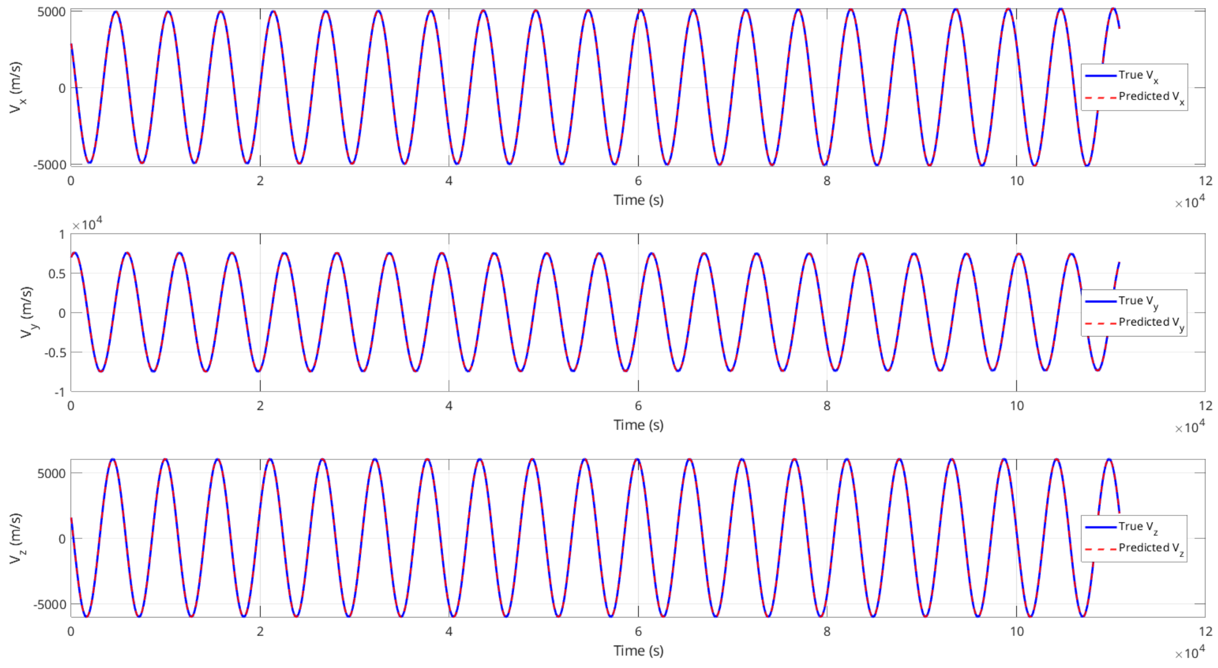


Figure 9.8: Chief's true and estimated velocity in ECI over time

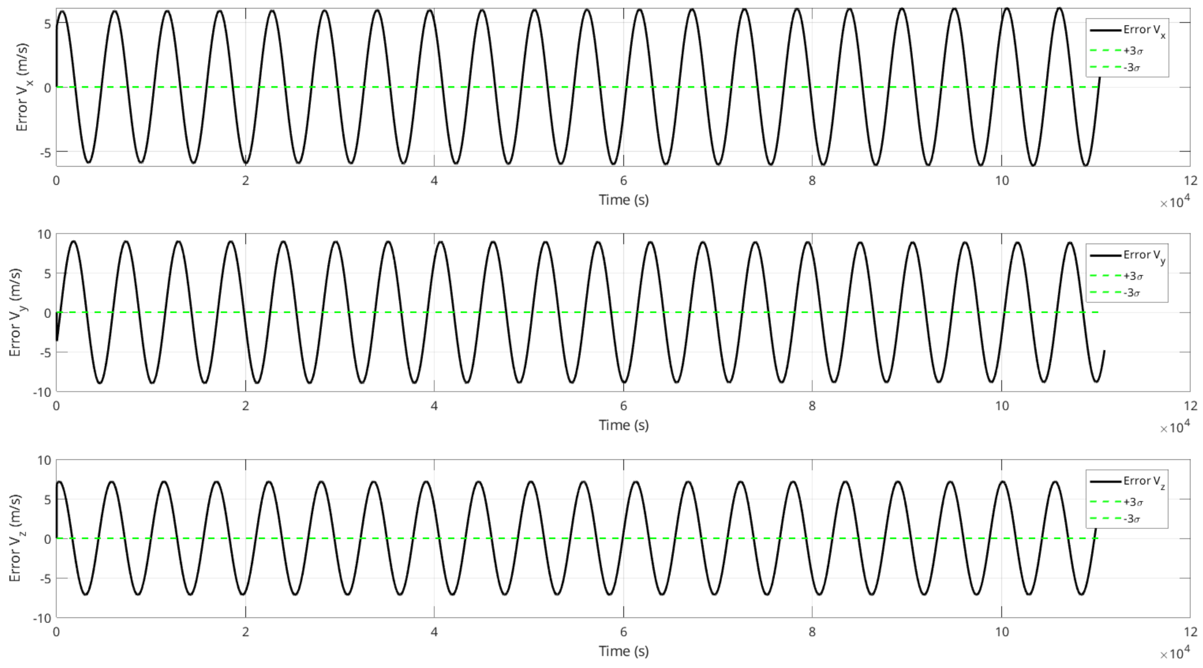


Figure 9.9: Chief's velocity in ECI error from UKF vs. ground truth with $\pm 3\sigma$ bounds

## Deputy

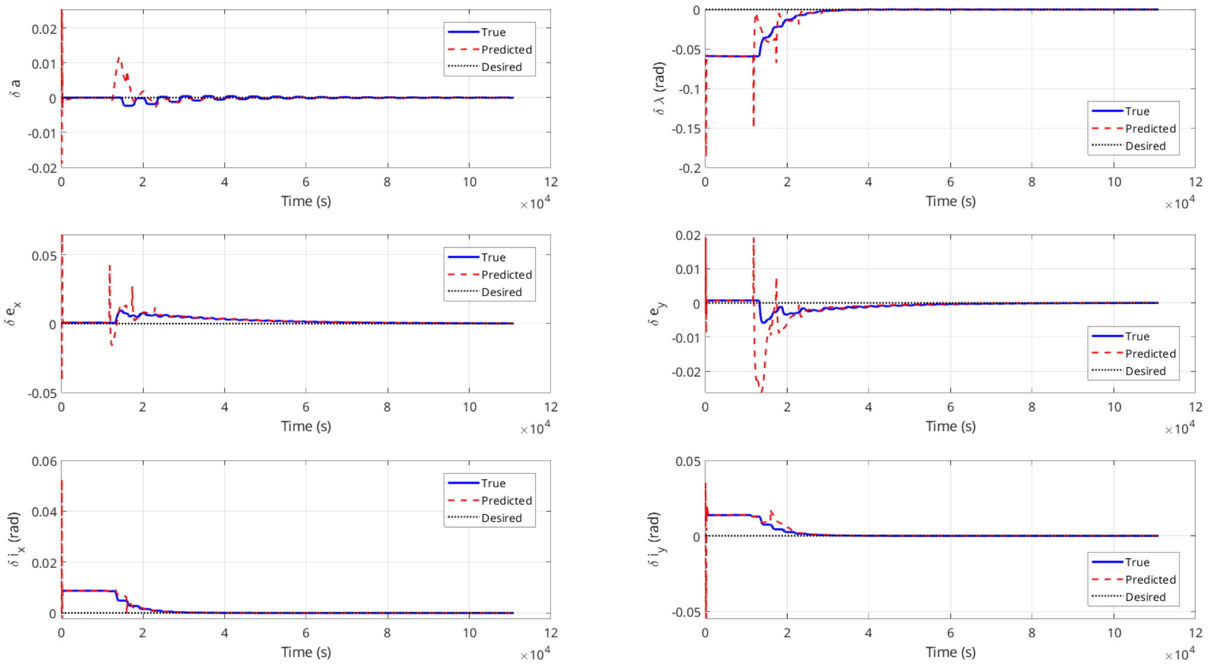


Figure 9.10: Deputy's true and estimated QNSROE as a function of time. Desired QNSROE is plotted as as reference.

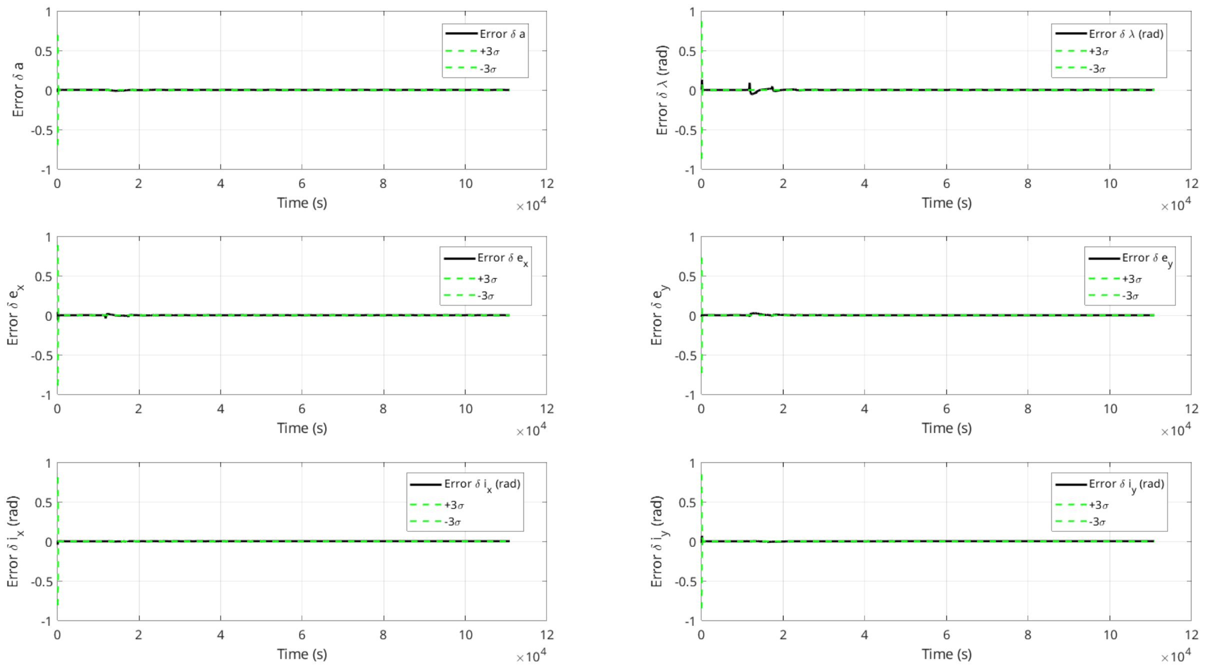


Figure 9.11: Deputy's QNSROE error from UKF vs. ground truth with $\pm 3\sigma$ bounds as a function of time.

## Measurements Residuals

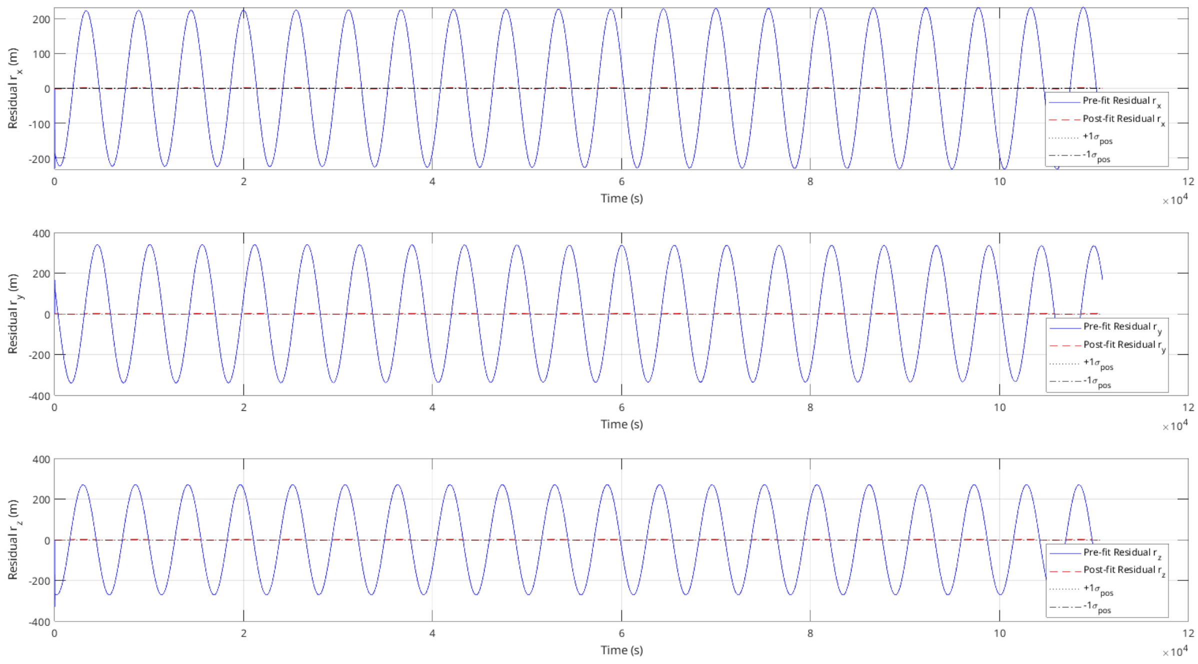


Figure 9.12: Pre-fit (blue) and post-fit (red) residuals for the relative position components $r_x$, $r_y$, and $r_z$, with $\pm 1\sigma_{\text{pos}}$ bounds shown in black.

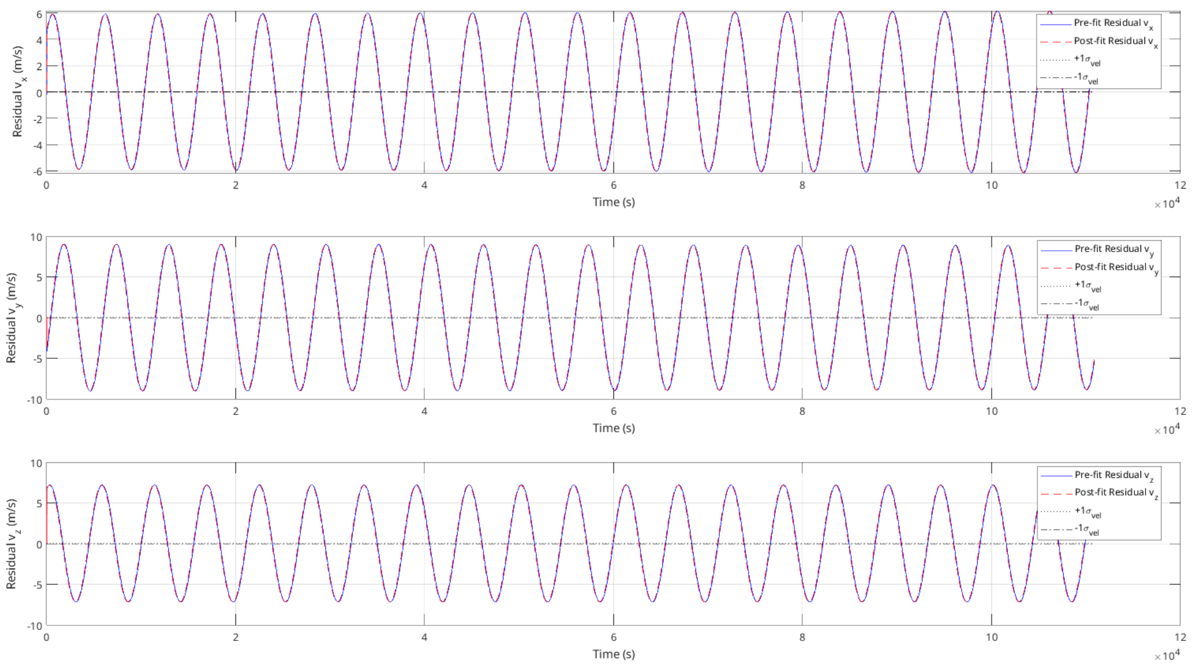


Figure 9.13: Pre-fit (blue) and post-fit (red) residuals for the relative velocity components $v_x$, $v_y$, and $v_z$, with $\pm 1\sigma_{\text{vel}}$ bounds shown in black.

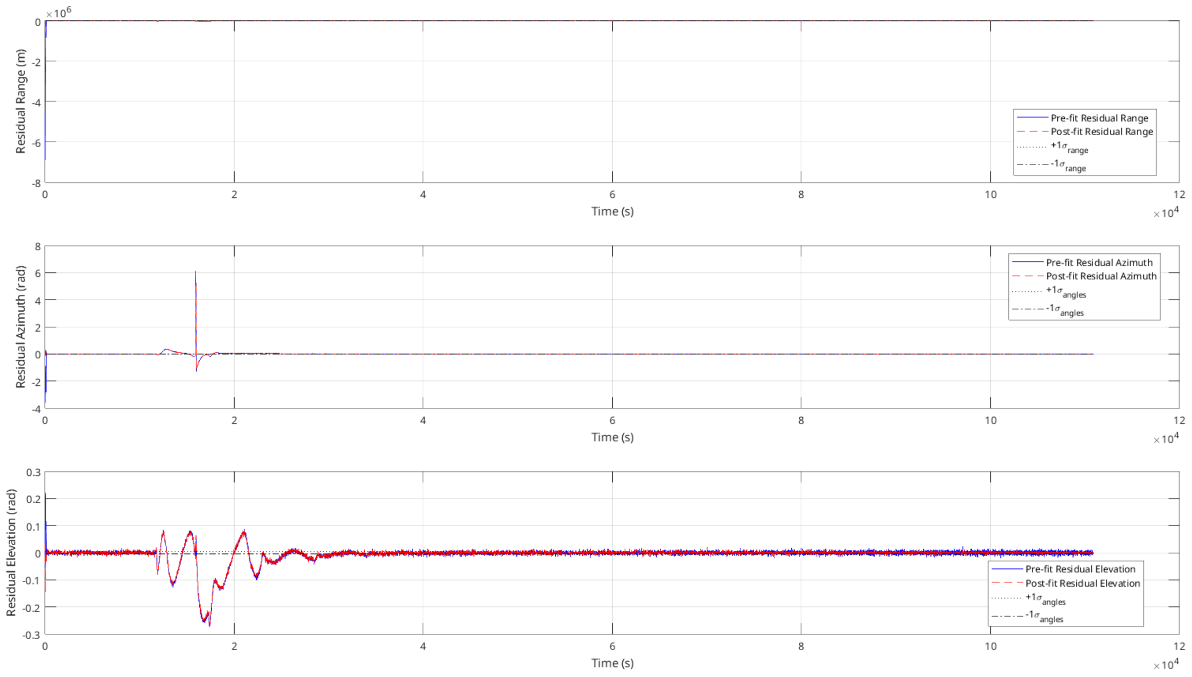
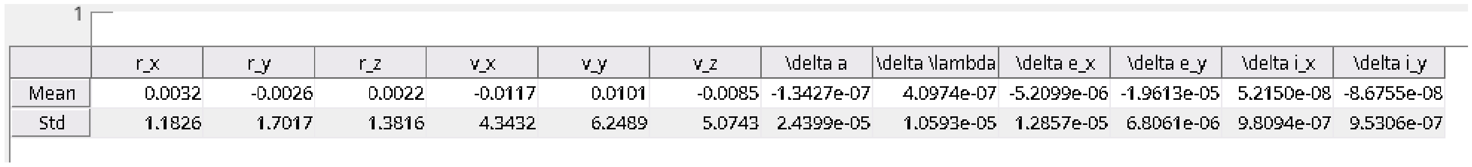


Figure 9.14: Pre-fit (blue) and post-fit (red) residuals for the deputy's range, azimuth, and elevation measurements, with $\pm 1\sigma_{\text{meas}}$ bounds shown in black.

## Navigation Statistics



1

| | r_x | r_y | r_z | v_x | v_y | v_z | \delta a | \delta \lambda | \delta e_x | \delta e_y | \delta i_x | \delta i_y |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Mean | 0.0032 | -0.0026 | 0.0022 | -0.0117 | 0.0101 | -0.0085 | -1.3427e-07 | 4.0974e-07 | -5.2099e-06 | -1.9613e-05 | 5.2150e-08 | -8.6755e-08 |
| Std | 1.1826 | 1.7017 | 1.3816 | 4.3432 | 6.2489 | 5.0743 | 2.4399e-05 | 1.0593e-05 | 1.2857e-05 | 6.8061e-06 | 9.8094e-07 | 9.5306e-07 |

Figure 9.15: Mean and standard deviation of the UKF state-estimation errors for all components (position, velocity, and deputy's QNSROE parameters) over the final orbit.

## Control Statistics

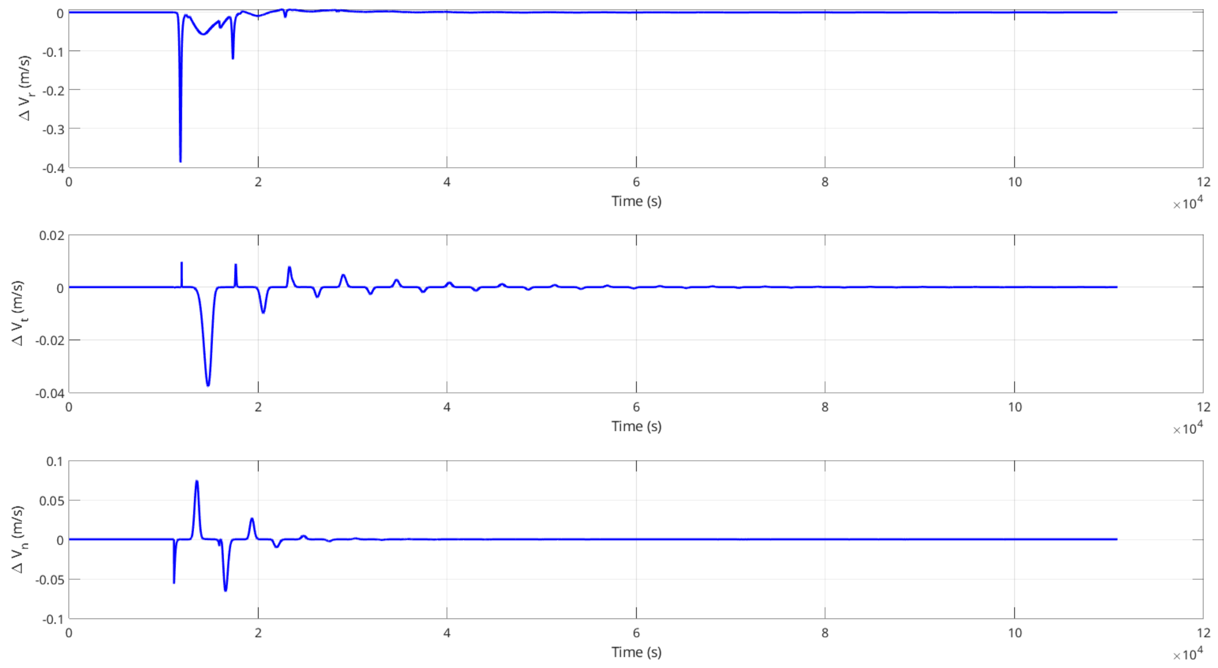


Figure 9.16: Evolution of the deputy's continuous impulse applied in its RTN frame over time.

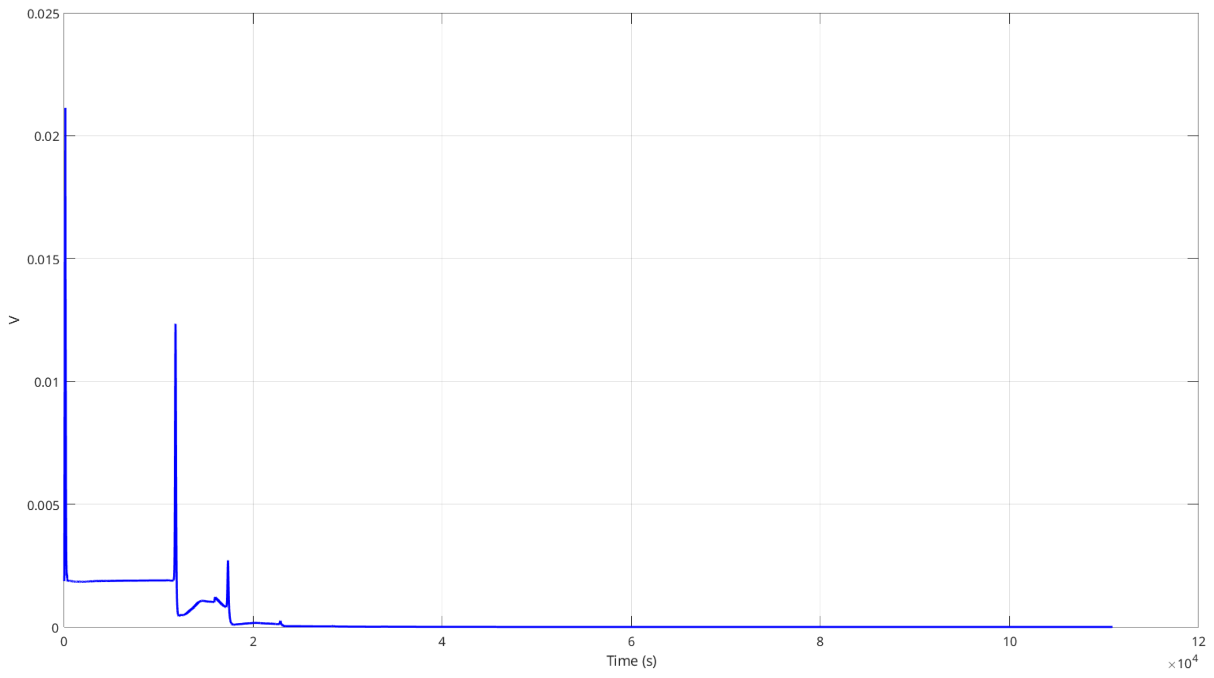


Figure 9.17: Evolution of the Lyapunov function over time.

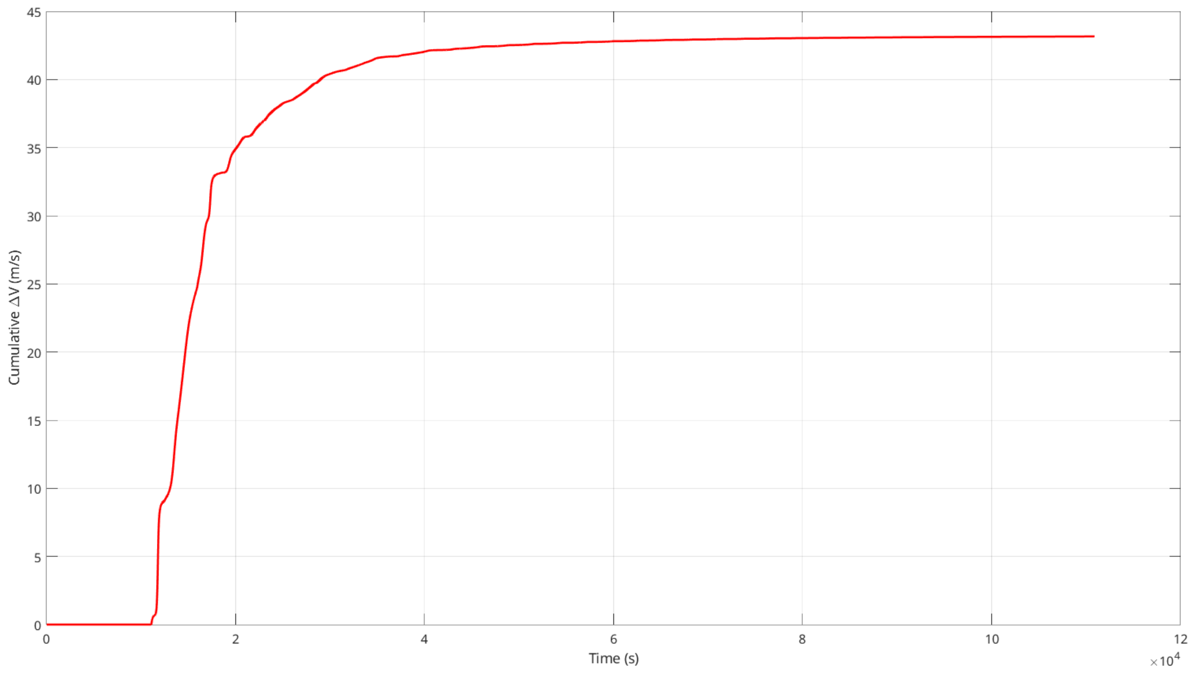


Figure 9.18: Evolution of the cumulative applied $\Delta V$ in the deputy's RTN frame.

The total final noisy cumulative applied $\Delta V_{tot_f}$ is 43.1564 m/s.

**Conclusion**

The integration of the UKF with a Lyapunov-based continuous control law demonstrates great performance in achieving precise state estimation and control for spacecraft formation flying under $J_2$ perturbations. The UKF effectively estimates the chief's ECI position and velocity, as well as the deputy's QNSROEs, leveraging noisy sensor measurements from GNSS, laser ranging, and camera angles. Simulation results show that the estimated states closely track the ground truth, with errors in all state components-chief's position, velocity, and deputy's QNSROE-converging to near zero, as evidenced by the tight $\pm 3\sigma$ bounds and low residual errors over the final orbit.

The Lyapunov control law, driven by the UKF's state estimates, successfully guides the deputy's QNSROE to the desired state (rendez-vous - $\delta \boldsymbol{\alpha}_{\text{desired}} \approx 0$) in approximately 15 orbits, achieving a fuel-efficient trajectory with a total cumulative $\Delta V$ of 43.1564 m/s. The control law's fuel-optimizing matrix $\mathbf{P}$ ensures thrusts are applied at orbitally advantageous positions, while the delayed control activation-based on UKF convergence-minimizes wasteful maneuvers during initial estimation uncertainties. The evolution of the Lyapunov function confirms stable convergence, and the cumulative $\Delta V$ remains only slightly higher than the noisy ground truth case (40.2 m/s), underscoring the robustness of the control strategy despite estimation errors.

This combined UKF and Lyapunov approach validates its suitability for precise formation control in orbital missions, offering a balance of accurate state estimation and efficient control. The minimal impact of noisy inputs, as seen in the small increase in $\Delta V$ compared to the ground truth, further highlights the system's resilience to modeling errors and sensor noise, making it a reliable solution for autonomous spacecraft formation maintenance.

## 9.3 Conclusions and Way Forward

### 9.3.1 Summary

Project DRAGON-DOCK developed a framework for spacecraft rendezvous and docking, integrating navigation and control for a chief-deputy pair in low Earth orbit. After evaluating both impulsive and continuous control strategies, as well as Extended and Unscented Kalman Filters for navigation, the final architecture employed a UKF with continuous thrust control. The project achieved precise relative motion and successful autonomous docking.

### 9.3.2 Results Critical Assessment

The UKF-enabled navigation and continuous control loop achieved high accuracy in simulations, ensuring stable rendezvous and docking.

However, the dynamics model oversimplifies by omitting atmospheric drag, solar pressure, higher-order J3/J4 terms, and third-body effects, limiting the STM's realism for control and navigation. A 3D satellite model would improve drag and torque calculations.

Moreover, simulated ground truth with uncalibrated noise lacks validation against space-grade sensors, and physical testbeds like SLAB TRON could enhance fidelity.

The basic rendezvous mission could include attitude constraints for sensors or communication, requiring IMU measurements. Control laws could restrict thruster inputs to specific RTN directions (e.g., radial and tangential) and incorporate collision avoidance.

Finally, the UKF assumes Gaussian noise, which may not capture real sensor behavior, and linearization errors in the STM affect long-term propagation accuracy.

#### 9.3.3 Lessons Learned

We learned several lessons.

First, accurate dynamics modeling is critical, but linearization introduces errors, particularly for eccentric orbits or long propagations.

Also, balancing computational complexity with simulation time steps is challenging; finer steps improve accuracy but increase runtime.

Finally, hardware-in-the-loop testing is essential for real-world validation. Simplifications in noise models and mission scope limit applicability, and iterative algorithm refinement is necessary for robustness.

#### 9.3.4 Project Continuations

In the future, we could incorporate more complex dynamics (drag, J3/J4, third-body effects) and 3D satellite models for realistic simulations.

We also want to validate results with hardware-in-the-loop testing using calibrated sensors and testbeds like SLAB TRON.

Finally, it would be interesting to explore new data-driven navigation techniques, such as computer vision, transformer and diffusion policy-based trajectory initialization to further spacecrafts autonomy. This could help ensure attitude control, collision avoidance, and constrained thruster inputs to enhance mission realism and safety.

## 10 References

[1] *Aa279d distributed space system*, `https://github.com/b9check/AA279D-Final-Code`, Accessed: 2025-06-11, 2025.

[2] T. Guffanti and S. D'Amico, "Linear models for spacecraft relative motion perturbed by solar radiation pressure and j2," *Journal of Guidance, Control, and Dynamics*, vol. 42, no. 9, pp. 1962–1981, 2019. DOI: `10.2514/1.G003695`. [Online]. Available: `https://slab.sites.stanford.edu/sites/g/files/sbiybj25201/files/media/file/jgcd2019_guffantidamico_online.pdf`.

[3] K. T. Alfriend, S. R. Vadali, P. Gurfil, J. P. How, and L. S. Breger, *Spacecraft Formation Flying: Dynamics, Control, and Navigation.* Elsevier Astrodynamics Series, 2010.

[4] M. Chernick and S. D'Amico, "New closed-form solutions for optimal impulsive control of spacecraft relative motion," *AIAA Journal of Guidance, Control, and Dynamics*, vol. 41, no. 2, pp. 419–434, 2018.

[5] M. Chernick and S. D'Amico, "Closedâform optimal impulsive control of spacecraft formations using reachable set theory," *Journal of Guidance, Control, and Dynamics*, vol. 44, no. 1, pp. 25–45, 2021.

[6] A. W. Koenig, T. Guffanti, and S. D'Amico, "New state transition matrices for spacecraft relative motion in perturbed orbits," *Journal of Guidance, Control, and Dynamics*, 2017. DOI: `10.2514/1.G002409`.

[7] H. Schaub and J. L. Junkins, *Analytical Mechanics of Aerospace Systems* (AIAA Education Series). Reston, VA: American Institute of Aeronautics and Astronautics, 2002, First edition (January 1, 2002), ISBN: 978-1-56347-563-4.

[8] R. Sullivan and S. D'Amico, "Nonlinear kalman filtering for improved angles-only navigation using relative orbital elements," *Journal of Guidance, Control, and Dynamics*, vol. 40, no. 7, pp. 1832–1847, 2017. DOI: `10.2514/1.G002233`.

[9] M. D'Errico, Ed., *Distributed Space Missions for Earth System Monitoring* (Space Technology Library). Springer, 2013, ISBN: 978-1-4614-4541-8. DOI: `10.1007/978-1-4614-4541-8`.

[10] J. H. Park and S. D'Amico, "Adaptive neural-network-based unscented kalman filter for robust pose tracking of noncooperative spacecraft," *Journal of Guidance, Control, and Dynamics*, vol. 46, no. 9, pp. 1671–1688, 2023. DOI: `10.2514/1.G007387`.

## A Appendix: Code

All code for this project can be found in the following Git repository: https://github.com/alex-crlhmmr/AA279D-Distributed-Space-System

### A.1 Utils (Conversion Functions)

```
function state_OE = ECI2OE(state_ECI, mu)
    r_ECI = state_ECI(1:3);
    v_ECI = state_ECI(4:6);

    h_vec = cross(r_ECI, v_ECI);
    h = norm(h_vec);

    e_vec = cross(v_ECI, h_vec) / mu - r_ECI / norm(r_ECI);
    e = norm(e_vec);

    % True anomaly
    argf = dot(e_vec, r_ECI) / (e * norm(r_ECI));
    argf = min(max(argf, -1), 1);
    f = acos(argf);
    if dot(r_ECI, v_ECI) < 0
        f = 2*pi - f;
    end

    % Inclination
    cosi = h_vec(3) / h;
    cosi = min(max(cosi, -1), 1);
    i = acos(cosi);

    % Node vector
    N_vec = cross([0 0 1], h_vec);
    N = norm(N_vec);

    % RAAN
    if N ~= 0
        argW = N_vec(1) / N;
        argW = min(max(argW, -1), 1);
        W = acos(argW);
        if N_vec(2) < 0
            W = 2*pi - W;
        end
    else
        W = 0;
    end

    % Argument of periapsis
    if N ~= 0 && e > 1e-10
        argw = dot(N_vec, e_vec) / (N * e);
        argw = min(max(argw, -1), 1);
        w = acos(argw);
        if e_vec(3) < 0
            w = 2*pi - w;
        end
    else
        w = 0;
    end

    % Semi-major axis
    a = 1 / (2 / norm(r_ECI) - norm(v_ECI)^2 / mu);
    state_OE = [a, e, i, W, w, f];
end


function rQNSOE = ECI2rQNSOE(chief_ECI, deputy_ECI, mu)
    N = size(chief_ECI, 1);
    rQNSOE = zeros(N, 6);
    for k = 1:N
        % Extract ECI states
        chief_state = chief_ECI(k, :)';
        deputy_state = deputy_ECI(k, :)';
        % Convert to Keplerian OE
        oe_chief = utils.ECI2OE(chief_state, mu);
        oe_deputy = utils.ECI2OE(deputy_state, mu);
        % Compute relative QNS
        rQNSOE(k, :) = utils.OE2rQNSOE(oe_chief, oe_deputy);
    end
end


function x_RTN = ECI2RTN(chief_ECI, deputy_ECI)
r0 = chief_ECI(1:3)';
v0 = chief_ECI(4:6)';
r1 = deputy_ECI(1:3)';
v1 = deputy_ECI(4:6)';

% ECI TO CHIEF RTN ROTATION MATRIX
R0_hat = r0 / norm(r0);
N0 = cross(r0, v0);
N0_hat = N0 / norm(N0);
T0 = cross(N0_hat, R0_hat);
T0_hat = T0 / norm(T0);
R_ECI2RTN = [R0_hat'; T0_hat'; N0_hat'];

% Relative position in RTN frame
r_rel = r1 - r0;
rRTN = R_ECI2RTN * r_rel;

% Relative velocity
omega_RTN = cross(r0, v0) / norm(r0)^2;
v_rel_inertial = v1 - v0;
v_rel_rotating = v_rel_inertial - cross(omega_RTN, r_rel);
vRTN = R_ECI2RTN * v_rel_rotating;

x_RTN = [rRTN; vRTN];

end


function f = MeanToTrueAnomaly(M, e)
    % Newton-Raphson to solve for E
    E = M;
    tol = 1e-12;
    max_iter = 1e5;
    for i = 1:max_iter
```

```
        f_E = E - e * sin(E) - M;
        f_prime_E = 1 - e * cos(E);
        dE = f_E / f_prime_E;
        E = E - dE;
        if abs(dE) < tol
            break;
        end
    end

    % Convert E to true anomaly
    f = 2 * atan2(sqrt(1 + e) * sin(E/2), sqrt(1 - e) * cos(E/2));
    f = mod(f, 2*pi);
end


function E = newton_raphson(M, e, epsilon)
    if nargin < 3
        epsilon = 1e-10;
    end

    E = M;
    max_iter = 1e5;

    for i = 1:max_iter
        f_E = E - e * sin(E) - M;
        f_prime_E = 1 - e * cos(E);
        increment = f_E / f_prime_E;
        E = E - increment;

        if abs(increment) <= epsilon
            break;
        end
    end
end


function state_ECI = OE2ECI(state_OE, mu)

    % Extract state variables
    i = state_OE(3); % inclination [rad]
    W = state_OE(4); % RAAN (Omega) [rad]
    w = state_OE(5); % argument of perigee [rad]

    % Get perifocal state
    state_perifocal = utils.OE2Perifocal(state_OE, mu);

    % Rotation matrices
    R3_W = [ cos(W), -sin(W), 0;
             sin(W),  cos(W), 0;
                  0,       0, 1];

    R1_i = [1,      0,       0;
            0, cos(i), -sin(i);
            0, sin(i),  cos(i)];

    R3_w = [ cos(w), -sin(w), 0;
             sin(w),  cos(w), 0;
                  0,       0, 1];

    % Total rotation matrix
    R_tot = R3_W * R1_i * R3_w;

    % Rotate position and velocity vectors
    r_ECI = R_tot * state_perifocal(1:3);
    v_ECI = R_tot * state_perifocal(4:6);

    % Output state vector
    state_ECI = [r_ECI; v_ECI];
end


function state_perifocal = OE2Perifocal(state_OE, mu)
    % Extract state variables
    a = state_OE(1); % semi-major axis [m]
    e = state_OE(2); % eccentricity [-]
    f = state_OE(6); % true anomaly [rad]

    % Compute the distance
    r = a * (1 - e^2) / (1 + e * cos(f));

    % Position in perifocal frame
    r_perifocal = [r * cos(f);
                   r * sin(f);
                   0];

    % Velocity in perifocal frame
    h = sqrt(mu * a * (1 - e^2)); % angular momentum
    v_perifocal = [-mu / h * sin(f);
                    mu / h * (e + cos(f));
                    0];

    % Output state vector
    state_perifocal = [r_perifocal; v_perifocal];
end


function QNS = OE2QNSOE(x)
% input: [a, e, i, Omega, omega, nu] (all in radians except a and e)

a     = x(1);
e     = x(2);
inc   = x(3);
RAAN  = x(4);
argp  = x(5);
nu    = x(6);

% All computations in radians
E     = 2 * atan( sqrt((1-e)/(1+e)) * tan(nu/2) );
M     = E - e * sin(E);
lam   = M + argp + RAAN * cos(inc);
ex    = e * cos(argp);
ey    = e * sin(argp);
ix    = inc;
iy    = RAAN * sin(inc);

QNS   = [a; lam; ex; ey; ix; iy];
end

```

```

function rQNSOE = OE2rQNSOE(oe_chief, oe_deputy)
    % Extract chief elements
    a_0 = oe_chief(1);
    e_0 = oe_chief(2);
    i_0 = oe_chief(3);
    W_0 = oe_chief(4);
    w_0 = oe_chief(5);
    f_0 = oe_chief(6);
    M_0 = utils.TrueToMeanAnomaly(f_0, e_0);
    % Extract deputy elements
    a_1 = oe_deputy(1);
    e_1 = oe_deputy(2);
    i_1 = oe_deputy(3);
    W_1 = oe_deputy(4);
    w_1 = oe_deputy(5);
    f_1 = oe_deputy(6);
    M_1 = utils.TrueToMeanAnomaly(f_1, e_1);

    % Compute QNS-ROE
    delta_a  = (a_1 - a_0) / a_0;
    delta_lambda = (M_1 + w_1) - (M_0 + w_0) + (W_1 - W_0) * cos(i_0);
    delta_ex = e_1 * cos(w_1) - e_0 * cos(w_0);
    delta_ey = e_1 * sin(w_1) - e_0 * sin(w_0);
    delta_ix = i_1 - i_0;
    delta_iy = (W_1 - W_0) * sin(i_0);
    rQNSOE = [delta_a, delta_lambda, delta_ex, delta_ey, delta_ix, delta_iy];
end


function QNSROE = ROE2QNSROE(chief_OE, deputy_SROE)
% ROE2QNSROE  Convert singular relative orbital elements to quasi-nonsingular ROEs
%   chief_OE      = [a0; e0; i0; RAAN0; argp0; nu0]   (all angles in radians)
%   deputy_SROE   = [da; de; di; dOmega; domega; dnu] (all angles in radians)

% Extract chief orbital elements
e0      = chief_OE(2);
i0      = chief_OE(3);
RAAN0   = chief_OE(4);
argp0   = chief_OE(5);
nu0     = chief_OE(6);

% Extract singular relative elements
da      = deputy_SROE(1);
de      = deputy_SROE(2);
di      = deputy_SROE(3);
dRAAN   = deputy_SROE(4);
domega  = deputy_SROE(5);
dnu     = deputy_SROE(6);

% Convert delta-nu to delta-mean anomaly (dM) (all in radians)
E0 = 2 * atan( sqrt((1 - e0) / (1 + e0)) * tan(nu0 / 2) );
Ed = 2 * atan( sqrt((1 - e0) / (1 + e0)) * tan((nu0 + dnu) / 2) );
M0 = E0 - e0 * sin(E0);
Md = Ed - e0 * sin(Ed);
dM = Md - M0;

% Compute QNSROE components (all trigs in radians)
dlam = dM + domega + dRAAN * cos(i0);
```

```
dex  = de * cos(argp0) - e0 * domega * sin(argp0);
dey  = de * sin(argp0) + e0 * domega * cos(argp0);
dix  = di;
diy  = dRAAN * sin(i0);

% Assemble output
QNSROE = [da; dlam; dex; dey; dix; diy];
end


function deputy_OE = rQNSOE2OE(chief_OE, deputy_rQNSOE)
    % Unpack chief OEs
    a_c = chief_OE(1);
    e_c = chief_OE(2);
    i_c = chief_OE(3);
    W_c = chief_OE(4);
    w_c = chief_OE(5);
    f_c = chief_OE(6);
    E_c = 2 * atan( sqrt((1 - e_c)/(1 + e_c)) * tan(f_c/2) );  % Eccentric anomaly
    M_c = E_c - e_c * sin(E_c);                                % Mean anomaly
    %lambda_c = W_c + w_c + M_c;
    % Unpack deputy rQNS OEs
    delta_a = deputy_rQNSOE(1);
    delta_lambda = deputy_rQNSOE(2);
    delta_e_x = deputy_rQNSOE(3);
    delta_e_y = deputy_rQNSOE(4);
    delta_i_x = deputy_rQNSOE(5);
    delta_i_y = deputy_rQNSOE(6);
    % Deputy orbital elements
    a_d = a_c*(delta_a + 1);
    W_d = delta_i_y*sin(i_c) + W_c;
    i_d = delta_i_x + i_c;
    alpha = delta_e_y + e_c*sin(w_c);
    beta = delta_e_x + e_c*cos(w_c);
    w_d = atan2(alpha, beta);
    e_d = sqrt(alpha^2 + beta^2);
    M_d = delta_lambda - (W_d - W_c)*cos(i_c) + (M_c + w_c) - w_d;
    E_d = newton_raphson(M_d,e_d);
    % Ensure e_d stays in valid [0,1) range
e_d = min(1 - 1e-12, max(0, e_d));

% Safeguard sqrt inputs from rounding errors
sqrt1pe = sqrt(max(0, 1 + e_d));
sqrt1me = sqrt(max(0, 1 - e_d));

f_d = 2 * atan2(sqrt1pe * sin(E_d/2), sqrt1me * cos(E_d / 2));
    deputy_OE = [a_d, e_d, i_d, W_d, w_d, f_d];
end


function E = newton_raphson(M, e, epsilon)
    if nargin < 3
        epsilon = 1e-10;
    end
    E = M;
    max_iter = 1e5;
    for i = 1:max_iter
        f_E = E - e * sin(E) - M;
        f_prime_E = 1 - e * cos(E);
        increment = f_E / f_prime_E;
        E = E - increment;
        if abs(increment) <= epsilon
            break;
        end
    end
end


function [rECI, vECI] = RTN2ECI(r_chief_ECI, v_chief_ECI, r_deputy_RTN, v_deputy_RTN)
r0 = r_chief_ECI;
v0 = v_chief_ECI;
r1 = r_deputy_RTN;
v1 = v_deputy_RTN;

% CHIEF RTN TO ECI ROTATION MATRIX
R0_hat = r0 / norm(r0);
N0 = cross(r0, v0);
N0_hat = N0 / norm(N0);
T0 = cross(N0_hat, R0_hat);
T0_hat = T0 / norm(T0);
R_RTN2ECI = [R0_hat'; T0_hat'; N0_hat']';

% CALCULATE DEPUTY RTN COORDS
rECI = r0 + R_RTN2ECI * r1;
vECI = v0 + R_RTN2ECI * v1;

end


function M = TrueToMeanAnomaly(f, e)
E = 2 * atan( sqrt((1 - e)/(1 + e)) * tan(f/2) );
E = mod(E, 2*pi);
M = mod(E - e*sin(E), 2*pi);
end
```

## A.2 Propagation Functions

### A.2.1 FODE Propagation Functions

```
function statedot = getStatedot(t, state, mu, R, J2, perturbated)
    % Extract state variables
    x = state(1);
    y = state(2);
    z = state(3);
    vx = state(4);
    vy = state(5);
    vz = state(6);

    % Precompute common terms
    r_vec = [x; y; z];
    v_vec = [vx; vy; vz];
    r = norm(r_vec);

    % Allocate output
    statedot = zeros(6, 1);

    % Two-body (Keplerian) acceleration
    accel_kepler = -mu / r^3 * r_vec;

    % Perturbations
    if nargin < 5 || perturbated
        perturbations = propagators.FodePropagator.getPerturbations(t, state, mu, R, J2);
    else
        perturbations = zeros(3, 1);
    end

    % State derivative
    statedot(1:3) = v_vec;
    statedot(4:6) = accel_kepler + perturbations;
end


function perturbations = getPerturbations(t, state, mu, R, J2)
    % Initialize perturbation in RTN frame
    perturbations = zeros(3, 1);

    % Add J2 perturbations
    J2Perturbations = propagators.FodePropagator.getJ2Perturbations(t, state, mu, R, J2);

    perturbations = perturbations + J2Perturbations;
end


function J2perturbations = getJ2Perturbations(t, state, mu, R, J2)
    % Extract state variables
    x = state(1);
    y = state(2);
    z = state(3);


    % Precompute common terms
    r_vec = [x; y; z];
    r = norm(r_vec);
    r2 = r^2;
    z2 = z^2;

    % J2 factor
    J2_factor = 1.5 * J2 * mu * R^2 / r^5;


    % J2 perturbations in ECI frame
    ax_J2 = J2_factor * x * (5 * z2 / r2 - 1);
    ay_J2 = J2_factor * y * (5 * z2 / r2 - 1);
    az_J2 = J2_factor * z * (5 * z2 / r2 - 3);

    % Output
    J2perturbations = [ax_J2; ay_J2; az_J2];
end
```

### A.2.2 Keplerian Propagation Functions

```
function statedot = getStatedot(t, state, mu, R, J2, perturbated)
    % Extract state variables
    a = state(1); % semi-major axis [m]
    e = state(2); % eccentricity [-]
    i = state(3); % inclination [rad]
    W = state(4); % RAAN (Omega) [rad]
    w = state(5); % argument of perigee [rad]
    f = state(6); % true anomaly [rad]

    % Precompute common terms
    p = a * (1 - e^2);
    r = p / (1 + e * cos(f));
    u = w + f;
    n = sqrt(mu / a^3);
    eta = sqrt(1-e^2);

    % Get perturbations
    if perturbated
        perturbations = propagators.KeplerianPropagator.getPerturbations(t, state, mu, R,
    J2);
    else
        perturbations = zeros(3, 1);
    end
    f_R = perturbations(1);
    f_T = perturbations(2);
    f_N = perturbations(3);

    % Allocate output
    statedot = zeros(6, 1);

    % Gauss' variational equations

    % da/dt
    statedot(1) = ((2*e*sin(f))/(n*eta))*f_R + ((2*a*eta)/(n*r))*f_T;

    % de/dt
    statedot(2) = ((eta*sin(f))/(n*a))*f_R + ((eta)/(n*a^2*e))*(((a^2*eta^2)/(r))-r)*f_T;

    % di/dt
    statedot(3) = ((r*cos(u))/(n*a^2*eta))*f_N;

    % dOmega/dt
    statedot(4) = ((r*sin(u))/(n*a^2*eta*sin(i)))*f_N;

    % domega/dt
    statedot(5) = ((-eta*cos(f))/(n*a*e))*f_R + ((eta)/(n*a*e))*((2+e*cos(f))/(1+e*cos(f))
    )*(sin(f))*f_T + ((-r*(1/tan(i))*sin(u))/(n*a^2*eta))*f_N;

    % df/dt
    statedot(6) = (sqrt(mu * p) / r^2) + (eta / (e * a * n)) * (cos(f) * f_R - (1 + r / p)
     * sin(f) * f_T);
end


function mean_dot = getMeanStatedot(t, state, mu, R, J2, J2_perturbed)
    % state = [a; e; i; Omega; omega; nu]
    a     = state(1);
    e     = state(2);
    i     = state(3);
    Omega = state(4);
    omega = state(5);
    nu    = state(6);

    % Convert true anomaly Î½ â  eccentric anomaly E
    E = 2 * atan( sqrt((1 - e) / (1 + e)) * tan(nu / 2) );

    % Convert eccentric anomaly E â  mean anomaly M
    M = E - e * sin(E);

    % Common parameters
    p     = a * (1 - e^2);
    n     = sqrt(mu / a^3);
    cos_i = cos(i);

    if J2_perturbed
        % J2 secular perturbations (mean rates)
        dOmega_dt = -1.5 * n * J2 * (R^2 / p^2) * cos_i;
        domega_dt =  0.75 * n * J2 * (R^2 / p^2) * (5 * cos_i^2 - 1);
        dM_dt     =  n + 0.75 * n * J2 * (R^2 / p^2) * sqrt(1 - e^2) * (3 * cos_i^2 - 1);
    else
        % Two-body motion only
        dOmega_dt = 0;
        domega_dt = 0;
        dM_dt     = n;
    end

    % Time derivative of Î½ (based on chain rule through M)
    dE_dM    = 1 / (1 - e * cos(E));
    dnu_dE   = sqrt(1 - e^2) / (1 - e * cos(E));
    dnu_dt   = dnu_dE * dE_dM * dM_dt;

    % Mean state time derivative
    mean_dot = [
        0;            % da/dt (constant semi-major axis under J2)
        0;            % de/dt (no secular change under J2)
        0;            % di/dt (no secular change under J2)
        dOmega_dt;
        domega_dt;
        dnu_dt
    ];
end


function perturbations = getPerturbations(t, state, mu, R, J2)
    % Initialize perturbation in RTN frame
    perturbations = zeros(3, 1);

    % Add J2 perturbations
    J2Perturbations = propagators.KeplerianPropagator.getJ2Perturbations(t, state, mu, R,
    J2);

    perturbations = perturbations + J2Perturbations;
end


function J2perturbations = getJ2Perturbations(t, state, mu, R, J2)
    % Extract state variables
    a = state(1); % semi-major axis [m]
    e = state(2); % eccentricity [-]
    i = state(3); % inclination [rad]
    W = state(4); % RAAN (Omega) [rad]
    w = state(5); % argument of perigee [rad]
    f = state(6); % true anomaly [rad]

    % Precompute common terms
    r = a * (1 - e^2) / (1 + e * cos(f));
    sin_i = sin(i);
    cos_i = cos(i);
    sin_u = sin(w + f);
    cos_u = cos(w + f);

    % J2 factor
    J2_factor = -1.5 * mu * J2 * R^2 / r^4;

    % J2 perturbations in RTN frame
    f_R = J2_factor * (1 - 3 * sin_i^2 * sin_u^2);
    f_T = J2_factor * sin_i^2 * sin_u * cos_u;
    f_N = J2_factor * sin_i * cos_i * sin_u;

    % Output
    J2perturbations = [f_R; f_T; f_N];
end
```

### A.2.3 Nonlinear Propagation Function

```
function [statedot] = nonlinear_state_dot(t, state, mu, Re, J2)
statedot = zeros(12, 1);

% CHIEF ECI INTEGRATOR ---------------------------------
% Unpack state
x0 = state(1);
y0 = state(2);
z0 = state(3);
vx0 = state(4);
vy0 = state(5);
```

```
vz0 = state(6);
r0_vec = [x0; y0; z0];
v0_vec = [vx0; vy0; vz0];
r0 = norm(r0_vec);

% Get perturbations
z2 = z0^2;
r2 = r0^2;
factor = -1.5 * J2 * mu * Re^2 / r0^7;
perturbations = factor * [ ...
    x0 * (1 - 5*z2/r2);
    y0 * (1 - 5*z2/r2);
    z0 * (3 - 5*z2/r2) ];

% Calculate stateerivative
statedot(1:3) = state(4:6);
accel_kepler = -mu / r0^3 * r0_vec;
statedot(4:6) = accel_kepler + perturbations;

% DEPUTY RTN INTEGRATOR
% Unpack state
x1 = state(7);
y1 = state(8);
z1 = state(9);
x1dot = state(10);
y1dot = state(11);
z1dot = state(12);

% Positional derivative = velocity
statedot(7:9) = state(10:12);

% Compute chief theta derivatives and r derivatives
h = cross(r0_vec, v0_vec);
theta0dot = norm(h) / r0^2;
r0dot = dot(r0_vec, v0_vec) / r0;
theta0dot2 = (-2*r0dot*theta0dot) / r0;

% Calculate theta/derivates and r/derivatives from chief state
statedot(10) = 2*theta0dot*y1dot + theta0dot2*y1 + theta0dot^2*x1 - (mu*(r0+x1)) / ((r0+x1
    )^2 + y1^2 + z1^2)^1.5 + mu/r0^2;
statedot(11) = -2*theta0dot*x1dot - theta0dot2*x1 + theta0dot^2*y1 - (mu*y1) / ((r0+x1)^2
    + y1^2 + z1^2)^1.5;
statedot(12) = -(mu*z1) / ((r0+x1)^2 + y1^2 + z1^2)^1.5;

end
```

## A.3 Control Code Helper Functions

```matlab
function Phi_J2_qns = getQNS_J2_STM(OE, tau, mu, J2, R)
    % Extract orbital elements
    a = OE(1); e = OE(2); i = OE(3);
    W = OE(4); w = OE(5);

    % Derived quantities
    eta = sqrt(1 - e^2);
    kappa = (3/4) * J2 * R^2 * sqrt(mu) / (a^(7/2) * eta^4);

    % Auxiliary terms
    E = 1 + eta;
    F = 4 + 3 * eta;
    G = 1 / eta^2;
    P = 3 * cos(i)^2 - 1;
    Q = 5 * cos(i)^2 - 1;
    R = cos(i);
    S = sin(2*i);
    T = sin(i)^2;

    % Perifocal eccentricity vector components
    exi = e * cos(w);
    eyi = e * sin(w);

    % Secular rates
    w_dot = kappa * Q;

    % Final perigee argument
    omega_f = w + w_dot * tau;
    exf = e * cos(omega_f);
    eyf = e * sin(omega_f);

    % Trig terms
    cos_wt = cos(w_dot * tau);
    sin_wt = sin(w_dot * tau);

    % Construct STM
    Phi_J2_qns = zeros(6,6);
    Phi_J2_qns(1,1) = 1;
    Phi_J2_qns(2,1) = -((3/2)*sqrt(mu/a^3) + (7/2)*kappa*E*P) * tau;
    Phi_J2_qns(2,2) = 1;
    Phi_J2_qns(2,3) = kappa * exi * F * G * P * tau;
    Phi_J2_qns(2,4) = kappa * eyi * F * G * P * tau;
    Phi_J2_qns(2,5) = -kappa * F * S * tau;

    Phi_J2_qns(3,1) = (7/2) * kappa * eyf * Q * tau;
    Phi_J2_qns(3,3) = cos_wt - 4 * kappa * exi * eyf * G * Q * tau;
    Phi_J2_qns(3,4) = -sin_wt - 4 * kappa * eyi * eyf * G * Q * tau;
    Phi_J2_qns(3,5) = 5 * kappa * eyf * S * tau;

    Phi_J2_qns(4,1) = -(7/2) * kappa * exf * Q * tau;
    Phi_J2_qns(4,3) = sin_wt + 4 * kappa * exi * exf * G * Q * tau;
    Phi_J2_qns(4,4) = cos_wt + 4 * kappa * eyi * exf * G * Q * tau;
    Phi_J2_qns(4,5) = -5 * kappa * exf * S * tau;

    Phi_J2_qns(5,5) = 1;
    Phi_J2_qns(6,1) = (7/2) * kappa * S * tau;
    Phi_J2_qns(6,3) = -4 * kappa * exi * G * S * tau;
    Phi_J2_qns(6,4) = -4 * kappa * eyi * G * S * tau;
    Phi_J2_qns(6,5) = 2 * kappa * T * tau;
    Phi_J2_qns(6,6) = 1;
end


function Bc = getBcMatrix(chief_OE, mu, J2, R)
     % Extract orbital elements
    a = chief_OE(1); e = chief_OE(2); i = chief_OE(3);
    W = chief_OE(4); w = chief_OE(5); f = chief_OE(6);

    % Derived quantities
    eta = sqrt(1 - e^2);
    nc = sqrt(mu / a^3);
    gamma = (3/4) * J2 * R^2 * sqrt(mu);
    kappa = gamma / (a^(7/2) * eta^4);

    % Trig values
    sin_wf = sin(w + f);
    cos_wf = cos(w + f);
    tan_i = tan(i);
    cos_f = cos(f);
    sin_f = sin(f);

    % Eccentricity vector components
    ex = e * cos(w);
    ey = e * sin(w);

    % Common factors
    denom = 1 + e * cos_f;
    ec2pf = (2 + e * cos_f); % shorthand

    % Compute B matrix
    Bc = zeros(6,3);
    Bc(1,1) = 2 * e * sin_f / eta;
    Bc(1,2) = 2 * (1 + e * cos_f) / eta;
    Bc(2,1) = -2 * eta^2 / denom;

    Bc(3,1) = eta * sin_wf;
    Bc(3,2) = eta * (ec2pf * cos_wf + ex) / denom;
    Bc(3,3) = eta * ey * sin_wf / (tan_i * denom);

    Bc(4,1) = -eta * cos_wf;
    Bc(4,2) = eta * (ec2pf * sin_wf + ey) / denom;
    Bc(4,3) = -eta * ex * sin_wf / (tan_i * denom);

    Bc(5,3) = eta * cos_wf / denom;

    Bc(6,3) = eta * sin_wf / denom;

    % Normalize
    Bc = Bc / (a * nc);
end


function P_bar = getPMatrix(chief_OE, current_QNSROE, desired_QNSROE, k, N)
    % Extract chief orbital elements
    a = chief_OE(1); % Semi-major axis (m)
    e = chief_OE(2); % Eccentricity (-)
    i = chief_OE(3); % Inclination (rad)
    W = chief_OE(4); % RAAN (rad)
    w = chief_OE(5); % Argument of perigee (rad)
    f = chief_OE(6); % True anomaly (rad)

    % Compute mean anomaly (M) from true anomaly (f)
    E = 2 * atan2(sqrt(1 - e) * sin(f/2), sqrt(1 + e) * cos(f/2)); % Eccentric anomaly
    M = E - e * sin(E); % Mean anomaly

    % Mean argument of latitude
    phi = w + M;

    % Compute tracking errors
    delta_rQNSOE = current_QNSROE - desired_QNSROE;
    delta_rQNSOE_ex = delta_rQNSOE(3);
    delta_rQNSOE_ey = delta_rQNSOE(4);
    delta_rQNSOE_ix = delta_rQNSOE(5);
    delta_rQNSOE_iy = delta_rQNSOE(6);

    % Compute fuel-optimal angles
    phi_ip = atan2(delta_rQNSOE_ey, delta_rQNSOE_ex); % In-plane optimal angle
    phi_oop = atan2(delta_rQNSOE_iy, delta_rQNSOE_ix); % Out-of-plane optimal angle
    phi_lambda = w; % Radial thrust at perigee (approximate)

    % Angular offsets
    J = phi - phi_ip;
    H = phi - phi_oop;
    K = phi - phi_lambda;

    % Ensure N is even and positive
    if mod(N, 2) ~= 0 || N < 2
        error('N must be an even positive integer >= 2');
    end

    % Construct P_bar
    P_bar = (1/k) * diag([cos(J)^N, cos(K)^N, cos(J)^N, cos(J)^N, cos(H)^N, cos(H)^N]);
end
```

### A.4 Problem Set 1 Code

FIGURES / PLOTTING / DISPLAY CODE REMOVED FOR BREVITY

```matlab
%% Problem Set 1 - Full Non-Plotting Code

%% 2b - Constants and Initialization
mu = 3.986004418e14;   % Earth's gravitational parameter [m^3/s^2]
R = 6378137;            % Earth's radius [m]
J2 = 1.08262668e-3;     % J2 perturbation term

% Define initial orbital elements
chief_OE = [6771e3, 0.0005, deg2rad(51.64), deg2rad(257), deg2rad(0), deg2rad(30)];
deputy_OE = [6771e3, 0.0015, deg2rad(52.14), deg2rad(258), deg2rad(0.5), deg2rad(25)];

% Convert orbital elements to ECI states
chief_ECI = utils.OE2ECI(chief_OE, mu);
deputy_ECI = utils.OE2ECI(deputy_OE, mu);

%% 2c - Cartesian Propagation (Perturbed and Unperturbed)
T = 2*pi*sqrt(chief_OE(1)^3 / mu);
tspan = 0:1:10*T;  % 10 orbital periods
state0 = chief_ECI;

% Unperturbed
perturbated = false;
odefun = @(t, state) propagators.FodePropagator.getStatedot(t, state, mu, R, J2,
    perturbated);
opts = odeset('RelTol',1e-12,'AbsTol',1e-14);
[t_out1, x_out1] = ode113(odefun, tspan, state0, opts);
r_mag1 = vecnorm(x_out1(:,1:3), 2, 2);
v_mag1 = vecnorm(x_out1(:,4:6), 2, 2);

% Perturbed (J2)
perturbated = true;
odefun = @(t, state) propagators.FodePropagator.getStatedot(t, state, mu, R, J2,
    perturbated);
[t_out2, x_out2] = ode113(odefun, tspan, state0, opts);
r_mag2 = vecnorm(x_out2(:,1:3), 2, 2);
v_mag2 = vecnorm(x_out2(:,4:6), 2, 2);

%% 2d - Keplerian Propagation (Osculating)
state0 = chief_OE;
perturbated = false;
odefun = @(t, state) propagators.KeplerianPropagator.getStatedot(t, state, mu, R, J2,
    perturbated);
[t_out3, x_out3] = ode113(odefun, tspan, state0, opts);

x_RTNs = zeros(length(tspan), 6);
for i = 1:length(tspan)
    ECI_1 = x_out1(i, :);
    OE_2 = x_out3(i, :);
    ECI_2 = utils.OE2ECI(OE_2, mu)';
    x_RTN = utils.ECI2RTN(ECI_1, ECI_2)';
    x_RTNs(i, :) = x_RTN;
end

%% 2e - Keplerian Propagation (Perturbed)
state0 = chief_OE;
```

```
perturbated = true;
odefun = @(t, state) propagators.KeplerianPropagator.getStatedot(t, state, mu, R, J2,
    perturbated);
[t_out4, x_out4] = ode113(odefun, tspan, state0, opts);

n = size(x_out3,1);
e_vecs1 = zeros(n,3);
h_vecs1 = zeros(n,3);
energy_vec1 = zeros(n,1);
e_vecs2 = zeros(n,3);
h_vecs2 = zeros(n,3);
energy_vec2 = zeros(n,1);

for k = 1:n
    % Unperturbed
    oe1 = x_out3(k,:);
    state1 = utils.OE2ECI(oe1, mu)';
    r1 = state1(1:3);
    v1 = state1(4:6);
    e_vecs1(k,:) = (1/mu)*((norm(v1)^2 - mu/norm(r1))*r1 - dot(r1,v1)*v1);
    h_vecs1(k,:) = cross(r1, v1);
    energy_vec1(k) = norm(v1)^2/2 - mu/norm(r1);

    % Perturbed
    oe2 = x_out4(k,:);
    state2 = utils.OE2ECI(oe2, mu)';
    r2 = state2(1:3);
    v2 = state2(4:6);
    e_vecs2(k,:) = (1/mu)*((norm(v2)^2 - mu/norm(r2))*r2 - dot(r2,v2)*v2);
    h_vecs2(k,:) = cross(r2, v2);
    energy_vec2(k) = norm(v2)^2/2 - mu/norm(r2);
end

e_mag1 = vecnorm(e_vecs1, 2, 2);
h_mag1 = vecnorm(h_vecs1, 2, 2);
e_mag2 = vecnorm(e_vecs2, 2, 2);
h_mag2 = vecnorm(h_vecs2, 2, 2);

%% 2f - Mean Elements Propagation
state0 = chief_OE;
odefun = @(t, state) propagators.KeplerianPropagator.getMeanStatedot(t, state, mu, R, J2);
[t_out5, x_out5] = ode113(odefun, tspan, state0, opts);
```

## A.5 Problem Set 2 Code

FIGURES / PLOTTING / DISPLAY CODE REMOVED FOR BREVITY

```
%% 1b
% Constants
mu = 3.986004418e14;
Re = 6378137;
J2 = 1.08262668e-3;

% Chief orbital elements
chief_OE = [6771e3; 0.0005; deg2rad(51.64); deg2rad(257); deg2rad(0); deg2rad(30)];
deputy_OE = [6771e3; 0.0015; deg2rad(52.14); deg2rad(257.5); deg2rad(0.5); deg2rad(25)];

% Convert OE to ECI and RTN state vectors
chief_ECI  = utils.OE2ECI(chief_OE, mu);
deputy_ECI = utils.OE2ECI(deputy_OE, mu);
deputy_RTN = utils.ECI2RTN(chief_ECI', deputy_ECI');

% Initialize combined state: [chief_ECI; deputy_RTN]
state0 = zeros(12, 1);
state0(1:6, :) = chief_ECI;
state0(7:12, :) = deputy_RTN;

% Propagate nonlinear dynamics
T = 2*pi*sqrt(chief_OE(1)^3 / mu); % Orbital period
tspan = 0:1:T*10;
options = odeset('RelTol', 1e-12, 'AbsTol', 1e-12);
odefun = @(t, state) propagators.NonlinearPropagator.nonlinear_state_dot(t, state, mu, Re, J2);
[t_out1, x_out1] = ode113(odefun, tspan, state0, options);

%% 1c
% Propagate chief
T = 2*pi*sqrt(chief_OE(1)^3 / mu); % Period
tspan = 0:1:10*T;
state0 = chief_ECI;

% Call ode113 to propagate chief without J2 perturbations
perturbated = false;
odefun = @(t, state) propagators.FodePropagator.getStatedot(t, state, mu, Re, J2, perturbated);
opts = odeset('RelTol',1e-12,'AbsTol',1e-14);
[t_out2, x_out2] = ode113(odefun, tspan, state0, opts);

% Propagate deputy
state0 = utils.OE2ECI(deputy_OE, mu);
[t_out3, x_out3] = ode113(odefun, tspan, state0, opts);
deputy_ECI = utils.OE2ECI(deputy_OE, mu);

RTNs = zeros(length(tspan), 6);
for i = 1:length(tspan)
    RTN = utils.ECI2RTN(x_out2(i,:), x_out3(i,:));
    RTNs(i, :) = RTN;
end

%% 1d â   New set of ICs with â a  â   0 (drift-inducing)
deputy_OE_drift = [6781e3; 0.0015; deg2rad(52.14); deg2rad(257.5); deg2rad(0.5); deg2rad(25)];
```

```matlab

deputy_ECI_drift = utils.OE2ECI(deputy_OE_drift, mu);
deputy_RTN_drift = utils.ECI2RTN(chief_ECI', deputy_ECI_drift');

state0_drift = zeros(12,1);
state0_drift(1:6) = chief_ECI;
state0_drift(7:12) = deputy_RTN_drift;

[t_drift_rel, x_drift_rel] = ode113(@(t, state) propagators.NonlinearPropagator.
    nonlinear_state_dot(t, state, mu, Re, J2*0), tspan, state0_drift, options);

state0 = utils.OE2ECI(deputy_OE_drift, mu);
[t_drift_abs, x_drift_abs] = ode113(@(t, state) propagators.FodePropagator.getStatedot(t,
    state, mu, Re, J2*0, false), tspan, state0, opts);

RTNs_drift = zeros(length(tspan),6);
for i = 1:length(tspan)
    RTNs_drift(i,:) = utils.ECI2RTN(x_out2(i,:), x_drift_abs(i,:));
end

%% 1e: Impulse calculation
a = chief_OE(1);
v_end = norm(x_out2(end, 4:6));
da = a - deputy_OE_drift(1);
dvT = (mu*da) / (2*a^2*v_end);

%% 1f: Simulate
chief_initial = x_drift_rel(end, 1:6);
deputy_initial = x_drift_rel(end, 7:12) + [0, 0, 0, 0, dvT, 0];
state0 = [chief_initial'; deputy_initial']';

T = 2*pi*sqrt(chief_OE(1)^3 / mu); % Orbital period
tspan = 0:1:T*10;
options = odeset('RelTol', 1e-12, 'AbsTol', 1e-12);
odefun = @(t, state) propagators.NonlinearPropagator.nonlinear_state_dot(t, state, mu, Re,
    J2);
[t_out3, x_out3] = ode113(odefun, tspan, state0, options);

RTN_full = [x_drift_rel(:, 7:12); x_out3(2:end, 7:12)];
t_full = [t_drift_rel; t_out3(2:end) + t_drift_rel(end)];
t_orb_full = t_full / T;
```

## A.6 Problem Set 3 Code

FIGURES / PLOTTING / DISPLAY CODE REMOVED FOR BREVITY

```
%% 2B
alpha0 = [6771; 0.1005; deg2rad(51.64); deg2rad(257); deg2rad(0); deg2rad(30)]; % Chief
alpha1 = [6771; 0.1006; deg2rad(51.69); deg2rad(257.05); deg2rad(0.05); deg2rad(29.95)]; %
    Deputy

% Calculate initial state of chief
mu = 398600.4418;
chief_ECI0 = utils.OE2ECI(alpha0, mu);
r0 = chief_ECI0(1:3);
v0 = chief_ECI0(4:6);

% Calculate initial state [RTN position and velocity] of deputy
deputy_ECI0 = utils.OE2ECI(alpha1, mu);
r1 = deputy_ECI0(1:3);
v1 = deputy_ECI0(4:6);
RTN_0 = utils.ECI2RTN(chief_ECI0, deputy_ECI0);
rRTN = RTN_0(1:3);
vRTN = RTN_0(4:6);

% Solve for constants
a = alpha0(1);
e = alpha0(2);
f = alpha0(6);
mu = 398600.4418;
n = sqrt(mu/a^3);
h = norm(cross(r0,v0));

% Calculate values needed to get constants
fdot = h/norm(r0)^2;
eta = sqrt(1-e^2);
k = 1 + e*cos(f);
c = k*cos(f);
s = k*sin(f);

% Initial state
xbar0 = [ ...
   rRTN(1)/a;
   vRTN(1)/(fdot*a);
   rRTN(2)/a;
   vRTN(2)/(fdot*a);
   rRTN(3)/a;
   vRTN(3)/(fdot*a);
];

% Get constants
phi_inv = (1/eta^2) * [
    -3*s*(k+e^2)/k^2,          c - 2*e,     0,  -s*(k+1)/k,      0,              0;
    -3*(e + c/k),              -s,          0,  -(c*(k+1)/k + e), 0,             0;
     3*k - eta^2,              e*s,         0,   k^2,            0,               0;
    -3*e*s*(k+1)/k^2,         -2 + e*c,     eta^2, -e*s*(k+1)/k,  0,              0;
     0,                        0,           0,   0,              eta^2*cos(f),  -eta^2*
    sin(f);
     0,                        0,           0,   0,              eta^2*sin(f),   eta^2*
    cos(f)
];
```

```

Constants = phi_inv * xbar0;

%% 2C
% Propagate
step_size = 1*(pi/180);
f_change = 15*2*pi;
[f_vals, states] = get_YA_states(alpha0, Constants, step_size, f_change, n, mu, h);

x  = states(:, 1);
vx = states(:, 2);
y  = states(:, 3);
vy = states(:, 4);
z  = states(:, 5);
vz = states(:, 6);

% Remove normalization
x  = x  * a;
vx = vx * a * fdot;
y  = y  * a;
vy = vy * a * fdot;
z  = z  * a;
vz = vz * a * fdot;

% Plotting and display commands have been removed

function [f_vals, states] = get_YA_states(alpha0, constants, step_size, f_change, n, mu, h
    )
    %GET_YA_STATES  Propagate YA state transition matrix instead of manual phi
    %   alpha0: 6Ã1 [Ï0; e; â¦; f0]
    %   constants: 6Ã1 initial YA state
    %   step_size: step size in true anomaly (rad)
    %   f_change: total trueâ anomaly change (rad)
    %   n: mean motion (rad/s)
    %   mu: gravitational parameter (m^3/s^2)
    %   h: specific angular momentum (m^2/s)

    % extract elements
    e         = alpha0(2);
    f_initial = alpha0(6);
    f_final   = f_initial + f_change;
    f_vals    = f_initial:step_size:f_final;
    states    = zeros(length(f_vals), 6);

    % compute meanâ anomaly time history
    E = 2*atan( sqrt((1-e)./(1+e)) .* tan(f_vals/2) );
    E = unwrap(E);
    M = E - e.*sin(E);
    M0 = M(1);
    t  = (M - M0) ./ n;        % time since f0

    % recover semi-major axis from n: n = sqrt(mu / a^3)
    a = (mu / n^2)^(1/3);

    % build STM function handle using STMYA
    Options.f0 = f_initial;
    Options.e  = e;
    Options.a  = a;
    Options.mu = mu;
    Phi_fun     = STMYA([], Options);

    t0 = 0;
    for i = 1:length(f_vals)
        tf = t(i);
        Phi = Phi_fun(tf, t0);
        state_i = Phi * constants;
        states(i, :) = state_i';
    end
end

function A = EOMYA(System, Options)
    % Scaled Version:
    % A = @(f) [0 0 0 1 0 0;0 0 0 0 1 0;0 0 0 0 0 1;3/(1+Options.e*cos(f)) 0 0 0 -2 0;0 0
    0 2 0 0;0 0 -1 0 0 0];

    % Unscaled Version (independent variable is true anomaly):
    % A = @(f) [zeros(3,3) eye(3,3);
    %            (3+e*cos(f))/(1+e*cos(f)) -2*e*sin(f)/(1+e*cos(f)) 0 2*e*sin(f)/(1+e*cos(f
    )) 2 0;
    %            2*e*sin(f)/(1+e*cos(f)) e*cos(f)/(1+e*cos(f)) 0 -2 2*e*sin(f)/(1+e*cos(f))
     0;
    %            0 0 -1/(1+e*cos(f)) 0 0 2*e*sin(f)/(1+e*cos(f))];

    % Unscaled Version (independent variable is time):
    A = @(t) EOM(t, Options.f0, Options.e, Options.a, Options.mu);
end

function Phi = STMYA(System, Options)
    function Phi = STM(tf, t0, f0, e, a, mu)
        M0 = utils.TrueToMeanAnomaly(f0, e);
        n  = sqrt(mu/a^3);
        ff = utils.MeanToTrueAnomaly((tf-t0)*n + M0, e);
        rhof = 1 + e*cos(ff);
        rho0 = 1 + e*cos(f0);
        sf = rhof*sin(ff);
        s0 = rho0*sin(f0);
        cf = rhof*cos(ff);
        c0 = rho0*cos(f0);
        spf = cos(ff) + e*cos(2*ff);
        sp0 = cos(f0) + e*cos(2*f0);
        cpf = -sin(ff) - e*sin(2*ff);
        cp0 = -sin(f0) - e*sin(2*f0);
        k2  = sqrt(mu)/sqrt(a*(1-e^2))^3;
        J   = k2*(tf-t0);

        Phixy = [sf 0 (2-3*e*sf*J) -cf;
                 cf*(1+1/rhof) 1 -3*rhof^2*J sf*(1+1/rhof);
                 spf -0 -3*e*(spf*J+sf/rhof^2) -cpf;
                 -2*sf 0 -3*(1-2*e*sf*J) 2*cf-e] * ...
                [-3*s0*(1/rho0+e^2/rho0^2) 0 c0-2*e -s0*(1+1/rho0);
                 -3*e*s0*(1/rho0+1/rho0^2) 1-e^2 e*c0-2 -e*s0*(1+1/rho0);
                  3*rho0+e^2-1 0 e*s0 rho0^2;
                  3*(c0/rho0+e) 0 s0 c0*(1+1/rho0)+e] * ...
                1/(1-e^2);

        rhof0 = 1 + e*cos(ff-f0);
        cf0   = rhof0*cos(ff-f0);
        sf0   = rhof0*sin(ff-f0);
        Phiz  = 1/rhof0 * [cf0 sf0; -sf0 cf0];

        Phi = [Phixy(1:2,1:2) zeros(2,1) Phixy(1:2,3:4) zeros(2,1);
               zeros(1,2) Phiz(1,1) zeros(1,2) Phiz(1,2);
               Phixy(3:4,1:2) zeros(2,1) Phixy(3:4,3:4) zeros(2,1);
               zeros(1,2) Phiz(2,1) zeros(1,2) Phiz(2,2)];

        Phi = [1/rhof*eye(3,3) zeros(3,3);
               k2*e*sin(ff)*eye(3,3) k2*rhof*eye(3,3)] * ...
              Phi * ...
              [rho0*eye(3,3) zeros(3,3);
               -e*sin(f0)*eye(3,3) 1/(k2*rho0)*eye(3,3)];
    end

    Phi = @(tf, t0) STM(tf, t0, Options.f0, Options.e, Options.a, Options.mu);
end

%% 2e
[a_0,e_0,i_0,W_0,w_0,f_0] = deal( ...
    6771, ...                       % a1
    0.1005, ...                     % e1
    deg2rad(51.64), ...             % i1
    deg2rad(257), ...               % O1
    deg2rad(0), ...                 % w1
    deg2rad(30) ...                 % f1
);
[a_1,e_1,i_1,W_1,w_1,f_1] = deal( ...
    6771, ...
    0.1006, ...
    deg2rad(51.69), ...
    deg2rad(257.05), ...
    deg2rad(0.05), ...
    deg2rad(29.95) ...
);

E_0 = 2*atan2(sqrt((1 - e_0)/(1 + e_0))*tan(f_0/2), 1);
M_0 = E_0 - e_0*sin(E_0);

E_1 = 2*atan2(sqrt((1 - e_1)/(1 + e_1))*tan(f_1/2), 1);
M_1 = E_1 - e_1*sin(E_1);

delta_a      = (a_1 - a_0)/a_0;
delta_lambda = (M_1 + w_1) - (M_0 + w_0) + (W_1 - W_0)*cos(i_0);
delta_ex     = e_1*cos(w_1) - e_0*cos(w_0);
delta_ey     = e_1*sin(w_1) - e_0*sin(w_0);
delta_ix     = i_1 - i_0;
delta_iy     = (W_1 - W_0)*sin(i_0);
qns_init     = [delta_a, delta_lambda, delta_ex, delta_ey, delta_ix, delta_iy];

%% 2f
quasi_elements = [delta_a, delta_lambda, delta_ex, delta_ey, delta_ix, delta_iy]';
alpha0         = [6771; 0.1005; deg2rad(51.64); deg2rad(257); deg2rad(0); deg2rad(30)]; %
    Chief

[f_vals2, states] = linear_ecc_mapping(quasi_elements, alpha0, 15*pi*2, 1*(pi/180));

x2  = states(:, 1);
y2  = states(:, 2);
z2  = states(:, 3);
```

```
vx2 = states(:, 4);
vy2 = states(:, 5);
vz2 = states(:, 6);

function [f_vals, states] = linear_ecc_mapping(quasi_elements, alpha0, f_change, step_size
    )
    % Solve for constants
    a = alpha0(1);
    e = alpha0(2);
    w = alpha0(5);
    i = alpha0(3);
    mu = 398600.4418;
    n  = sqrt(mu/a^3);

    % Calculate values needed to get constants
    eta = sqrt(1-e^2);
    ex  = e*cos(w);
    ey  = e*sin(w);

    % F vals
    f_initial = alpha0(6);
    f_final   = f_initial + f_change;
    f_vals    = f_initial:step_size:f_final;

    % Calculate t matrix
    E  = 2*atan2(sqrt((1 - e)./(1 + e)).*tan(f_vals./2), 1);
    E  = unwrap(E);
    M  = E - e.*sin(E);
    M0 = M(1);
    ts = (M - M0) ./ n;

    % Initialize states
    states = zeros(length(f_vals), 6);

    for j = 1:length(f_vals)
        f  = f_vals(j);
        k  = 1 + e*cos(f);
        kp = -e*sin(f);
        u  = f + w;
        t  = ts(j);

        % Compute B matrix entries (x component shown as example)
        bx1 = 1 / k + (3 / 2) * kp * n / eta^3 * t;
        bx2 = -kp / eta^3;
        bx3 = (1 / eta^3) * (ex * (k - 1) / (1 + eta) - cos(u));
        bx4 = (1 / eta^3) * (ey * (k - 1) / (1 + eta) - sin(u));
        bx6 = kp / eta^3 * cot(i);
        by1 = -(3 / 2) * k * n / eta^3 * t;
        by2 = k / eta^3;
        by3 = (1 / eta^2) * ((1 + 1 / k) * sin(u) + ey / k + k / eta * (ey / (1 + eta)));
        by4 = -(1 / eta^2) * ((1 + 1 / k) * cos(u) + ex / k + k / eta * (ex / (1 + eta)));
        by6 = (1 / k - k / eta^3) * cot(i);
        bz5 = (1 / k) * sin(u);
        bz6 = -(1 / k) * cos(u);
        % Velocities
        bxd1 = kp / 2 + (3 / 2) * k^2 * (1 - k) * n / eta^
        % ... [rest of the B and M assembly as in original code] ...
        % state update
        Phi   = M * B;
        state = Phi * quasi_elements;
        states(j,:) = state;
    end
end
```

## A.7 Problem Set 4 Code

FIGURES / PLOTTING / DISPLAY CODE REMOVED FOR BREVITY

```
% (all plotting/display calls removed)

clear;

%% 1. Setup
mu  = 3.986004418e14;
Re  = 6378137;
J2  = 1.08262668e-3;

% ICs (a)
chief_OE   = [6771e3; 0.0005; deg2rad(51.64); deg2rad(257); 0; deg2rad(30)];
deputy_OE  = [6771e3; 0.0015; deg2rad(52.14); deg2rad(257.5); deg2rad(0.5); deg2rad(25)];

% ICs (b)
rQNS_b      = [0;100;50;100;30;200] ./ chief_OE(1);
deputy2_OE = utils.rQNSOE2OE(chief_OE, rQNS_b);

% part g
qnsroes0   = utils.OE2rQNSOE(chief_OE, deputy_OE);
rQNS_b      = qnsroes0 - [0,0,0,0,qnsroes0(5),0];
deputy2_OE = utils.rQNSOE2OE(chief_OE, rQNS_b);

T     = 2*pi*sqrt(chief_OE(1)^3/mu);
tspan = (0:1:15*T)';

ode_noJ2 = @(t,s) propagators.KeplerianPropagator.getStatedot(t,s,mu,Re,J2,false);
ode_J2   = @(t,s) propagators.KeplerianPropagator.getStatedot(t,s,mu,Re,J2,true);
opts     = odeset('RelTol',1e-12,'AbsTol',1e-14);

%% 2. Propagate all four osculating cases
[tc1, xc1] = ode113(ode_noJ2, tspan, chief_OE,   opts);
[td1, xd1] = ode113(ode_noJ2, tspan, deputy_OE,  opts);
[tc2, xc2] = ode113(ode_J2,   tspan, chief_OE,   opts);
[td2, xd2] = ode113(ode_J2,   tspan, deputy_OE,  opts);
[tc3, xc3] = ode113(ode_noJ2, tspan, chief_OE,   opts);
[td3, xd3] = ode113(ode_noJ2, tspan, deputy2_OE, opts);
[tc4, xc4] = ode113(ode_J2,   tspan, chief_OE,   opts);
[td4, xd4] = ode113(ode_J2,   tspan, deputy2_OE, opts);

cases = { {xc1,xd1}, {xc2,xd2}, {xc3,xd3}, {xc4,xd4} };

%% 3. Build QNSOE & rQNSOE for each case
N     = numel(tspan);
QNS   = zeros(N,6,4);
rQNS  = zeros(N,6,4);

for c = 1:4
    xc = cases{c}{1};
    xd = cases{c}{2};
    for k = 1:N
        oe_c          = utils.ECI2OE(utils.OE2ECI(xc(k,:)',mu), mu);
        oe_d          = utils.ECI2OE(utils.OE2ECI(xd(k,:)',mu), mu);
        QNS(k,:,c)    = utils.OE2QNSOE(oe_d);
        rQNS(k,:,c)   = utils.OE2rQNSOE(oe_c, oe_d);
    end
    QNS(:,2,c)  = unwrap(QNS(:,2,c));
    rQNS(:,2,c) = unwrap(rQNS(:,2,c));
end

%% 4. Compute regression slopes & intercepts
t_orb          = tspan / T;
slopes_QNS     = zeros(4,6);
intercepts_QNS= zeros(4,6);
slopes_rQNS    = zeros(4,6);
intercepts_rQNS= zeros(4,6);

for c = 1:4
    for j = 1:6
        % QNS regression
        y        = QNS(:,j,c);
        p        = polyfit(t_orb, y, 1);
        slopes_QNS(c,j)      = p(1);
        intercepts_QNS(c,j) = p(2);

        % rQNS regression (converted to meters)
        y_m      = chief_OE(1) * rQNS(:,j,c);
        p_m      = polyfit(t_orb, y_m, 1);
        slopes_rQNS(c,j)      = p_m(1);
        intercepts_rQNS(c,j) = p_m(2);
    end
end

%% 5. Compute RTN trajectories for case 2 (IC a, with & without J2)
RTN_unp = zeros(N,3);
RTN_J2  = zeros(N,3);

for k = 1:N
    c1 = utils.ECI2RTN(utils.OE2ECI(xc1(k,:)',mu), utils.OE2ECI(xd1(k,:)',mu));
    RTN_unp(k,:) = c1(1:3)';
    c2 = utils.ECI2RTN(utils.OE2ECI(xc2(k,:)',mu), utils.OE2ECI(xd2(k,:)',mu));
    RTN_J2(k,:)  = c2(1:3)';
end

%% 6. Compute new deputy OE from part g
qnsroes0       = utils.OE2rQNSOE(chief_OE, deputy_OE);
qnsroes1       = qnsroes0;
qnsroes1(5)    = 0;
deputy_OE_new = utils.rQNSOE2OE(chief_OE, qnsroes1);

disp('Updated deputy OE (elements 2:end):');
disp(deputy_OE_new(2:end));

%% 7. STM-only propagation for two initial conditions
a0        = chief_OE(1);
e0        = chief_OE(2);
i0        = chief_OE(3);
omega0    = chief_OE(5);
n         = sqrt(mu/a0^3);
eta       = sqrt(1 - e0^2);
p_sma     = a0*(1 - e0^2);
kappa     = 1.5 * J2 * n * (Re/p_sma)^2;
P         = 0.5*(1 - 5*cos(i0)^2);
Qmat      = 0.25*(5*cos(i0)^2 - 1);
S         = 0.25*sin(i0)*cos(i0);
```

```
Tmat      = Qmat;
F         = 1/eta;
G         = 1/eta^3;
dot_omega= kappa * Qmat;

qns0_row = utils.OE2rQNSOE(chief_OE, deputy_OE);
qns1_row = qns0_row;
qns1_row(5) = 0;
qns0 = qns0_row(:);
qns1 = qns1_row(:);

anal0 = zeros(N,6);
anal1 = zeros(N,6);

for k = 1:N
    tau = tspan(k);
    wf  = omega0 + dot_omega * tau;
    exi = e0 * cos(omega0);
    eyi = e0 * sin(omega0);
    exf = e0 * cos(wf);
    eyf = e0 * sin(wf);
    Phi = eye(6);
    Phi(2,1) = -(1.5*n + 3.5*kappa*eta*P)*tau;
    Phi(2,3) =  kappa*exi*F*G*P*tau;
    Phi(2,4) =  kappa*eyi*F*G*P*tau;
    Phi(2,5) = -kappa*F*S*tau;
    Phi(3,1) =  3.5*kappa*eyf*Qmat*tau;
    Phi(3,3) =  cos(dot_omega*tau) - 4*kappa*exi*eyf*G*Qmat*tau;
    Phi(3,4) = -sin(dot_omega*tau) - 4*kappa*eyi*eyf*G*Qmat*tau;
    Phi(3,5) =  5*kappa*eyf*S*tau;
    Phi(4,1) = -3.5*kappa*exf*Qmat*tau;
    Phi(4,3) =  sin(dot_omega*tau) + 4*kappa*exi*exf*G*Qmat*tau;
    Phi(4,4) =  cos(dot_omega*tau) + 4*kappa*eyi*exf*G*Qmat*tau;
    Phi(4,5) = -5*kappa*exf*S*tau;
    Phi(6,1) =  3.5*kappa*S*tau;
    Phi(6,3) = -4*kappa*exi*G*S*tau;
    Phi(6,4) = -4*kappa*eyi*G*S*tau;
    Phi(6,5) =  2*kappa*Tmat*tau;

    anal0(k,:) = (a0*(Phi*qns0))';
    anal1(k,:) = (a0*(Phi*qns1))';
end

% Final results:
%   - slopes_QNS, intercepts_QNS
%   - slopes_rQNS, intercepts_rQNS
%   - RTN_unp, RTN_J2
%   - deputy_OE_new
%   - anal0, anal1
```

## A.8 Problem Set 5 Impulsive Control Code

FIGURES / PLOTTING / DISPLAY CODE REMOVED FOR BREVITY

```
%% — FIGURES / PLOTTING / DISPLAY CODE REMOVED FOR BREVITY

%% Define Rendezvous
alpha_c = [6771e3; 5.0e−4; deg2rad(51.64); deg2rad(257); deg2rad(45); deg2rad(30)];
delta_alpha_initial = [0; 0.001; deg2rad(0.5); deg2rad(1); deg2rad(1); deg2rad(−5)];
delta_alpha_desired = [0; 0; 0; 0; 0; deg2rad(−0.05)];
alpha_d_0 = alpha_c + delta_alpha_initial;
alpha_d_f = alpha_c + delta_alpha_desired;

QNS_chief    = utils.OE2QNSOE(alpha_c);
QNSROE_0     = utils.ROE2QNSROE(alpha_c, delta_alpha_initial);
QNSROE_F     = utils.ROE2QNSROE(alpha_c, delta_alpha_desired);

%% Compute Reachable Sets for Max Delta V
mu       = 3.986004418e14;
N_nu     = 200;   % orbit samples
N_phi    = 200;   % direction samples
N_th     = 200;   % polar samples for inclination plane
dv_max   = 7.36;

[kpolys, allPts] = computeReachableHulls(alpha_c, mu, N_nu, N_phi, N_th, dv_max);
QNSROE_des = alpha_c(1) * (QNSROE_F − QNSROE_0);

%% METHOD 1: Calculate optimal nu / min delta v using planes
deltaQNS  = QNSROE_F − QNSROE_0;
planes    = {[1 2], [3 4], [5 6]};
mu        = 3.986004418e14;
N         = 200;  % number of Î½ samples in dv_min_calc
for p = 1:3
    rows         = planes{p};
    dpl          = deltaQNS(rows);
    [dv_p, nu_p] = dv_min_calc_planes(rows, dpl, alpha_c, N, mu);
    dv_min(p)    = dv_p;
    nu_opt(p)    = nu_p;
end
[dv_tot, idx] = max(dv_min);
nu_star       = nu_opt(idx);     % anomaly for worst plane
fprintf('Method 1 (Planes): Minimum â v = %.3f m/s at Î½ = %.1fÂ°\n', dv_tot, rad2deg(
    nu_star));

%% METHOD 2: Calculate optimal nu / min delta v using all 6 at once
[v_best, nu_best] = dv_min_calc_global(alpha_c, N, deltaQNS, mu);
fprintf('Method 2 (True optimal): Minimum â v = %.3f m/s at Î½ = %.1fÂ°\n', norm(v_best),
     rad2deg(nu_best));
disp('RTN vector:'), disp(v_best)

%% Impulse control
J2      = 0.001082;
Re      = 6378e3;
tau     = 12*3600;
t_burn = (0.025:0.025:0.975) * tau;
t0      = 0;
t_f     = tau;
M       = numel(t_burn);
```

```
nu0    = alpha_c(6);
a      = alpha_c(1);
e      = alpha_c(2);

% Build A
A = zeros(6,3*M);
for j = 1:M
    dt_b      = t_burn(j) - t0;
    dt_to_f   = t_f - t_burn(j);

    Phi0_burn = PHI_J2(alpha_c, mu, J2, Re, dt_b);
    Phiburn_f = PHI_J2(alpha_c, mu, J2, Re, dt_to_f);
    nu_j      = propagate_nu(nu0, e, a, dt_b, mu, J2, Re, alpha_c(3));
    Gamma_j   = computeGamma(alpha_c, mu, nu_j);

    A(:,3*(j-1)+1:3*j) = Phiburn_f * Gamma_j;
end

% Pseudostate and solve
Phi0_f   = PHI_J2(alpha_c, mu, J2, Re, t_f - t0);
b        = QNSROE_F - Phi0_f * QNSROE_0;
dv_stack = pinv(A) * b;            % minimal â dvâ â  solution
t_star   = max(abs(dv_stack));     % largest singleâ burn magnitude

disp('Burns:')
for j = 1:M
    dvj = dv_stack(3*(j-1)+1:3*j);
    fprintf(' burn %d  â   [%.3f  %.3f  %.3f]\n', j, dvj);
end

dvR       = dv_stack(1:3:end);    % radial
dvT       = dv_stack(2:3:end);    % tangential
dvN       = dv_stack(3:3:end);    % normal
burn_mags = sqrt(dvR.^2 + dvT.^2 + dvN.^2);
total_dv  = sum(burn_mags);
fprintf('Maximum impulse = %.3f m/s\n', t_star);
fprintf('Total â v = %.3f m/s\n', total_dv);

%% Groundâ Truth Simulation, with and without control error
err_sigma = 0.05;

for caseIdx = 1:2
    if caseIdx == 1
        dv_use = dv_stack;
    else
        dv_use = dv_stack .* (1 + err_sigma * randn(size(dv_stack)));
    end

    dep_ECI = utils.OE2ECI(alpha_d_0, mu);
    chi_ECI = utils.OE2ECI(alpha_c,   mu);
    dep_RTN = utils.ECI2RTN(chi_ECI', dep_ECI');
    state   = [chi_ECI; dep_RTN];
    t_prev  = 0;

    T_all = [];
    X_all = [];

    for j = 1:M
        tspan = [t_prev, t_burn(j)];
        [T, X] = ode113(@(t,x) propagators.NonlinearPropagator.nonlinear_state_dot(t,x, mu, Re, J2), tspan, state);
        T_all = [T_all; T(1:end-1)];
        X_all = [X_all; X(1:end-1,:)];

        state          = X(end,:)';
        dvj            = dv_use(3*(j-1)+1 : 3*j);
        state(10:12)   = state(10:12) + dvj;
        t_prev         = t_burn(j);
    end

    [T, X] = ode113(@(t,x) propagators.NonlinearPropagator.nonlinear_state_dot(t,x, mu, Re, J2), [t_prev, tau], state);
    T_all   = [T_all; T];
    X_all   = [X_all; X];

    N      = size(T_all,1);
    Qhist = zeros(N,6);
    for k = 1:N
        r_c    = X_all(k,1:3)'; v_c = X_all(k,4:6)';
        r_rel = X_all(k,7:9)'; v_rel = X_all(k,10:12)';
        [r_d, v_d] = utils.RTN2ECI(r_c, v_c, r_rel, v_rel);

        chief_OE = utils.ECI2OE([r_c; v_c], mu);
        deputy_OE= utils.ECI2OE([r_d; v_d], mu);
        dSROE     = deputy_OE - chief_OE;
        Qhist(k,:) = utils.ROE2QNSROE(chief_OE, dSROE)';
    end

    if caseIdx == 1
        QNSROE_nominal    = Qhist;
    else
        QNSROE_perturbed = Qhist;
    end
end
```

## A.9 Problem Set 6 Code

FIGURES / PLOTTING / DISPLAY CODE REMOVED FOR BREVITY

```
%% - Initialization

% Constants
mu = 3.986004418e14;
R = 6378137;
J2 = 1.08262668e-3;

% Chief initial orbital elements
a_0 = 6771000 ;          % semi-major axis [m]
e_0 = 0.0005;            % eccentricity [-]
i_0 = deg2rad(51.64);    % inclination [rad]
W_0 = deg2rad(257);      % RAAN [rad]
w_0 = deg2rad(45);       % argument of perigee [rad]
f_0 = deg2rad(30);       % true anomaly [rad]

% Chief initial state in OE & ECI
initial_state_0_OE = [a_0, e_0, i_0, W_0, w_0, f_0];
initial_state_0_ECI = utils.OE2ECI(initial_state_0_OE, mu);
r0_init = initial_state_0_ECI(1:3);
v0_init = initial_state_0_ECI(4:6);

% Deputy initial relative quasi-non-singular OE
delta_a = 0;
delta_lambda = deg2rad(-3.3773);
delta_e_x = 0.0007;
delta_e_y = 0.0007;
delta_i_x = deg2rad(0.4989);
delta_i_y = deg2rad(0.7850);

initial_state_1_rQNS_OE = [delta_a, delta_lambda, delta_e_x, delta_e_y, delta_i_x,
    delta_i_y];

% Deputy initial state in OE & ECI
initial_state_1_OE = utils.rQNSOE2OE(initial_state_0_OE, initial_state_1_rQNS_OE);
initial_state_1_ECI = utils.OE2ECI(initial_state_1_OE, mu);
r1_init = initial_state_1_ECI(1:3);
v1_init = initial_state_1_ECI(4:6);

% Compute the distance between the two spacecraft
distance = norm(r1_init - r0_init);

% Display the distance
fprintf('Distance between the two spacecraft: %.2f meters\n', distance);

% Desired rQNSOE
epsilon = 1e-7;
desired_QNSROE = [epsilon; epsilon; epsilon; epsilon; epsilon; epsilon];

%% - Groundtruth Chief & Control Matrices

% Orbital period
T = 2 * pi * sqrt(a_0^3 / mu);

% Simulation parameters
tstart = 0;
```

```
tint = 10;
tend = 20 * T;
options = odeset('RelTol', 1e-12, 'AbsTol', 1e-12);

% Run the perturbed FODE propagator for chief in ECI
perturbated = true;
odefun = @(t, state) propagators.FodePropagator.getStatedot(t, state, mu, R, J2,
    perturbated);
[t, state_ECI_0_p] = ode113(odefun, (tstart:tint:tend)', initial_state_0_ECI, options);

% Preallocate storage for A and B matrices at each time step
N = length(t);
A_all = zeros(6, 6, N);
B_all = zeros(6, 3, N);

% Small time step for A_c numerical differentiation
tau_diff = 1e-3; % seconds

for idx = 1:N
    % Get current chief state in ECI and convert to OE
    state_ECI = state_ECI_0_p(idx, :)';
    OE_curr = utils.ECI2OE(state_ECI, mu);

    % Compute STM over small tau_diff and approximate A_c
    Phi_J2_qns = control_helpers.getQNS_J2_STM(OE_curr, tau_diff, mu, J2, R);
    A_all(:, :, idx) = (Phi_J2_qns - eye(6)) / tau_diff;

    % Compute B_c
    B_all(:, :, idx) = control_helpers.getBcMatrix(OE_curr, mu, J2, R);
end

%% - Noiseless Lyapunov

% Convert ECI trajectory to orbital elements
chief_OE_traj = zeros(N, 6);
for idx = 1:N
    state_ECI = state_ECI_0_p(idx, :)';
    chief_OE_traj(idx, :) = utils.ECI2OE(state_ECI, mu);
end

% Control parameters
k = 1e3;            % Scaling factor for P
N_param = 14;       % Thrust concentration exponent
dt = tint;          % Time step for Euler integration

% Initialize deputy ROE and storage
delta_alpha = initial_state_1_rQNS_OE';
delta_alpha_traj = zeros(N, 6);
u_traj = zeros(3, N);
V_traj = zeros(N, 1);
delta_alpha_traj(1, :) = delta_alpha';

for idx = 1:N-1
    A_c = A_all(:, :, idx);
    B_c = B_all(:, :, idx);

    % Compute P
    P = control_helpers.getPMatrix(chief_OE_traj(idx, :), delta_alpha, desired_QNSROE, k,
    N_param);

    % Compute Lyapunov function
    err = delta_alpha - desired_QNSROE;
    V_traj(idx) = 0.5 * (err' * err);

    % Compute control input
    u = -pinv(B_c) * (A_c * delta_alpha + P * err);
    u_traj(:, idx) = u;

    % Update ROE using Euler integration
    delta_alpha_dot = A_c * delta_alpha + B_c * u;
    delta_alpha = delta_alpha + delta_alpha_dot * dt;
    delta_alpha_traj(idx+1, :) = delta_alpha';
end

% Compute final V and u
V_traj(N) = 0.5 * norm(delta_alpha - desired_QNSROE)^2;
u_traj(:, N) = -pinv(B_all(:, :, N)) * (A_all(:, :, N) * delta_alpha + ...
    control_helpers.getPMatrix(chief_OE_traj(N, :), delta_alpha, desired_QNSROE, k,
   N_param) * (delta_alpha - desired_QNSROE));

% Compute relative distance trajectory
r_rel_traj = zeros(N,1);
for idx = 1:N
    chief_ECI = state_ECI_0_p(idx, :)';
    OE_c = chief_OE_traj(idx, :);
    delta_rQNS = delta_alpha_traj(idx, :);
    deputy_OE = utils.rQNSOE2OE(OE_c, delta_rQNS);
    deputy_ECI = utils.OE2ECI(deputy_OE, mu);
    r_rel_traj(idx) = norm(deputy_ECI(1:3) - chief_ECI(1:3));
end

% Save results
save('control_results.mat', 't', 'delta_alpha_traj', 'V_traj', ...
     'u_traj', 'A_all', 'B_all', 'chief_OE_traj', 'r_rel_traj');

%% - Noisy Sensors and Actuators

% Noise parameters
sigma_sensor = [1e-5; 1e-5; 1e-6; 1e-6; 1e-7; 1e-7];
sigma_act      = [1e-5; 1e-5; 1e-5];

% Preallocate storage
delta_alpha_noisy_traj = zeros(N,6);
u_noisy_traj           = zeros(3,N);
V_noisy_traj           = zeros(N,1);

% Initialize
delta_alpha_true = initial_state_1_rQNS_OE';
delta_alpha_noisy_traj(1, :) = delta_alpha_true';
V_noisy_traj(1) = 0.5 * norm(delta_alpha_true - desired_QNSROE)^2;

% Reseed for reproducibility
rng(0);

for idx = 1:N-1
    A_c = A_all(:,:,idx);
    B_c = B_all(:,:,idx);

    % Sensor measurement with noise
    meas_delta = delta_alpha_true + sigma_sensor .* randn(6,1);

    % Compute control based on noisy measurement
    err_meas = meas_delta - desired_QNSROE;
    P = control_helpers.getPMatrix(chief_OE_traj(idx,:), meas_delta, desired_QNSROE, k,
    N_param);
    u_nom = -pinv(B_c) * (A_c * meas_delta + P * err_meas);

    % Apply actuator noise
    u_applied = u_nom + sigma_act .* randn(3,1);
    u_noisy_traj(:, idx) = u_applied;

    % Propagate true ROE with noisy actuation
    delta_dot = A_c * delta_alpha_true + B_c * u_applied;
    delta_alpha_true = delta_alpha_true + delta_dot * dt;
    delta_alpha_noisy_traj(idx+1, :) = delta_alpha_true';

    % Lyapunov evaluation
    V_noisy_traj(idx+1) = 0.5 * norm(delta_alpha_true - desired_QNSROE)^2;
end

% Final actuator at N
u_noisy_traj(:,N) = -pinv(B_all(:,:,N)) * ...
    (A_all(:,:,N) * delta_alpha_true + ...
     control_helpers.getPMatrix(chief_OE_traj(N,:), delta_alpha_true, desired_QNSROE, k,
    N_param) * (delta_alpha_true - desired_QNSROE));

% Compute cumulative delta-V
deltaV_mag_free  = sqrt(sum(u_traj.^2, 1));
cumulative_deltaV_free = cumsum(deltaV_mag_free);

deltaV_mag_noisy = sqrt(sum(u_noisy_traj.^2, 1));
cumulative_deltaV_noisy = cumsum(deltaV_mag_noisy);

fprintf('Final cumulative deltaV (noise-free): %.4f m/s\n', cumulative_deltaV_free(end));
fprintf('Final cumulative deltaV (noisy):      %.4f m/s\n', cumulative_deltaV_noisy(end));
```

## A.10 Problem Set 8 Code

FIGURES / PLOTTING / DISPLAY CODE REMOVED FOR BREVITY

```matlab
%% Initialization
% Constants
mu = 3.986004418e14;
R = 6378137;
J2 = 1.08262668e-3;

% Chief initial orbital elements
a_0 = 6771000 ;         % semi-major axis [m]
e_0 = 0.0005;           % eccentricity [-]
i_0 = deg2rad(51.64);   % inclination [rad]
W_0 = deg2rad(257);     % RAAN [rad]
w_0 = deg2rad(45);      % argument of perigee [rad]
f_0 = deg2rad(30);      % true anomaly [rad]

% Chief initial state in OE & ECI
initial_state_0_OE = [a_0, e_0, i_0, W_0, w_0, f_0];
initial_state_0_ECI = utils.OE2ECI(initial_state_0_OE, mu);
r0_init = initial_state_0_ECI(1:3);
v0_init = initial_state_0_ECI(4:6);

% Deputy initial relative quasi-non-singular OE
delta_a = 0;
delta_lambda = deg2rad(-3.3773);
delta_e_x = 0.0007;
delta_e_y = 0.0007;
delta_i_x = deg2rad(0.4989);
delta_i_y = deg2rad(0.7850);
initial_state_1_rQNS_OE = [delta_a; delta_lambda; delta_e_x; delta_e_y; delta_i_x;
    delta_i_y];

% Deputy initial state in OE & ECI
initial_state_1_OE = utils.rQNSOE2OE(initial_state_0_OE,initial_state_1_rQNS_OE);
initial_state_1_ECI = utils.OE2ECI(initial_state_1_OE, mu);
r1_init = initial_state_1_ECI(1:3);
v1_init = initial_state_1_ECI(4:6);

% Compute the distance between the two spacecraft
distance = norm(r1_init - r0_init);

% Display the distance
fprintf('Distance between the two spacecraft: %.2f meters\n', distance);

%% - Generate Ground Truth State

% Orbital period
T = 2 * pi * sqrt(a_0^3 / mu); % Orbital period [s]

% Simulation parameters
tstart = 0;
tint = 1;
tend = 15*T;
options = odeset('RelTol', 1e-12, 'AbsTol', 1e-12);

% Run the perturbed FODE propagator for chief in ECI
perturbated = true;
```

```
odefun = @(t, state) propagators.FodePropagator.getStatedot(t, state, mu, R, J2,
    perturbated);
[t, state_ECI_0_p] = ode113(odefun, (tstart:tint:tend)', initial_state_0_ECI, options);

% Run the perturbed FODE for deputy in ECI
perturbated = true;
odefun = @(t, state) propagators.FodePropagator.getStatedot(t, state, mu, R, J2,
    perturbated);
[t, state_ECI_1_p] = ode113(odefun, (tstart:tint:tend)', initial_state_1_ECI, options);

% Consolidated state
x_c_gt = state_ECI_0_p;
x_d_gt = utils.ECI2rQNSOE(state_ECI_0_p, state_ECI_1_p, mu);
x_d_gt(:,2) = unwrap(x_d_gt(:,2));
x_gt = [x_c_gt, x_d_gt];

%% - Generate Measurements

% Number of time steps
N = length(t);

% Initialize measurement array
z_k = zeros(N, 9);

% Measurement noise covariance (diagonal, SI units)
sigma_pos = 0.001; % m, GNSS position (1 mm)
sigma_vel = 0.001; % m/s, GNSS velocity (1 mm/s)
sigma_range = 0.001; % m, laser range (1 mm)
sigma_angles = 4.848e-3; % rad, camera angles (~1000 arcsecond)

R_k = diag([sigma_pos^2, sigma_pos^2, sigma_pos^2, ...
            sigma_vel^2, sigma_vel^2, sigma_vel^2, ...
            sigma_range^2, sigma_angles^2, sigma_angles^2]);

% Square root of covariance
sqrt_R_k = sqrtm(R_k);

for k = 1:N
    % Ground truth state
    x_k_gt = x_gt(k,:)'; % 12 x 1

    % Compute noiseless measurement
    z_k_true = h_k(x_k_gt, zeros(9,1), mu);

    % Generate noise
    v_k = sqrt_R_k * randn(9,1);

    % Add noise
    z_k(k,:) = z_k_true + v_k;
end

%% - UKF

% UKF Parameters
n = 12; % State dimension
m = 9;  % Measurement dimension
alpha = 6*1e-1;
kappa = 0;
beta = 2;
```

```matlab
lambda = alpha^2 * (n + kappa) - n;

% Weights
Wm = zeros(2*n+1,1);
Wc = zeros(2*n+1,1);
Wm(1) = lambda / (n + lambda);
Wc(1) = lambda / (n + lambda) + (1 - alpha^2 + beta);
for i = 2:(2*n+1)
    Wm(i) = 1 / (2 * (n + lambda));
    Wc(i) = 1 / (2 * (n + lambda));
end

% Process noise covariance
sigma_pos_ECI = 1e-2; % m, position noise for chief ECI
sigma_vel_ECI = 1e-3; % m/s, velocity noise for chief ECI
sigma_rQNS_dimless = 1e-6; % dimensionless, for delta_a, delta_ex, delta_ey
sigma_rQNS_angle = 1e-6; % rad, for delta_lambda, delta_ix, delta_iy
Q_k = diag([sigma_pos_ECI^2 * ones(3,1); ... % ECI position
            sigma_vel_ECI^2 * ones(3,1); ... % ECI velocity
            sigma_rQNS_dimless^2; ... % delta_a
            sigma_rQNS_angle^2; ... % delta_lambda
            sigma_rQNS_dimless^2 * ones(2,1); ... % delta_ex, delta_ey
            sigma_rQNS_angle^2 * ones(2,1)]); % delta_ix, delta_iy

% Measurement noise covariance (diagonal, SI units)
sigma_pos = 0.001; % m, GNSS position (1 mm)
sigma_vel = 0.001; % m/s, GNSS velocity (1 mm/s)
sigma_range = 0.001; % m, laser range (1 mm)
sigma_angles = 4.848e-3; % rad, camera angles (~1000 arcsecond)

R_k = diag([sigma_pos^2, sigma_pos^2, sigma_pos^2, ...
            sigma_vel^2, sigma_vel^2, sigma_vel^2, ...
            sigma_range^2, sigma_angles^2, sigma_angles^2]);

% Initial state and covariance
x_hat = [initial_state_0_ECI; initial_state_1_rQNS_OE];
P_c = 1e-1 * eye(6);
P_d = 1e-2 * eye(6);
P = [P_c, zeros(6);
     zeros(6), P_d];

% Storage
x_hat_store = zeros(N, n);
P_store = zeros(n, n, N);

z_hat_store = zeros(N, m);
z_post_store = zeros(N, m);

for k = 1:N
    % Time step
    delta_t = tint;
    if k == 1
        delta_t = 0;
    end

    % Generate sigma points
    sqrt_P = chol((n + lambda) * P, 'lower');
    chi = zeros(n, 2*n+1);
    chi(:,1) = x_hat;
    for i = 1:n
        chi(:,i+1) = x_hat + sqrt_P(:,i);
        chi(:,i+n+1) = x_hat - sqrt_P(:,i);
    end

    % Prediction step
    chi_pred = zeros(n, 2*n+1);
    for i = 1:(2*n+1)
        chi_pred(:,i) = f_k(chi(:,i), zeros(6,1), zeros(n,1), delta_t, mu, J2, R);
    end
    x_hat_pred = zeros(n,1);
    for i = 1:(2*n+1)
        x_hat_pred = x_hat_pred + Wm(i) * chi_pred(:,i);
    end
    P_pred = Q_k;
    for i = 1:(2*n+1)
        diff = chi_pred(:,i) - x_hat_pred;
        P_pred = P_pred + Wc(i) * (diff * diff');
    end

    % Update step
    sqrt_P_pred = chol((n + lambda) * P_pred, 'lower');
    chi_pred_new = zeros(n, 2*n+1);
    chi_pred_new(:,1) = x_hat_pred;
    for i = 1:n
        chi_pred_new(:,i+1) = x_hat_pred + sqrt_P_pred(:,i);
        chi_pred_new(:,i+n+1) = x_hat_pred - sqrt_P_pred(:,i);
    end

    % Measurement sigma points
    Z_pred = zeros(m, 2*n+1);
    for i = 1:(2*n+1)
        Z_pred(:,i) = h_k(chi_pred_new(:,i), zeros(m,1), mu);
    end
    z_hat = zeros(m,1);
    for i = 1:(2*n+1)
        z_hat = z_hat + Wm(i) * Z_pred(:,i);
    end

    % Measurement covariance and cross-covariance
    S_k = R_k;
    P_xz = zeros(n,m);
    for i = 1:(2*n+1)
        z_diff = Z_pred(:,i) - z_hat;
        S_k = S_k + Wc(i) * (z_diff * z_diff');
        x_diff = chi_pred_new(:,i) - x_hat_pred;
        P_xz = P_xz + Wc(i) * (x_diff * z_diff');
    end

    % Kalman gain
    K_k = P_xz / S_k;

    % Update state and covariance
    z_k_current = z_k(k,:)';
    x_hat = x_hat_pred + K_k * (z_k_current - z_hat);
    P = P_pred - K_k * S_k * K_k';

    % Store results
    x_hat_store(k,:) = x_hat';
    P_store(:,:,k) = P;
    z_hat_store(k,:) = z_hat';
    z_post_store(k,:) = h_k(x_hat, zeros(m,1), mu)';
end

%% - Functions

function x_k = f_k(x_k1, u_k, w_k, delta_t, mu, J2, R)
    % Chief dynamics (ECI)
    x_c_k1 = x_k1(1:6);
    u_k_c = u_k(1:3);
    w_k_c = w_k(1:6);
    r_c_k1_n = norm(x_k1(1:3));
    x_c_k1_x = x_k1(1);
    x_c_k1_y = x_k1(2);
    x_c_k1_z = x_k1(3);
    help_vec = [x_c_k1_x*(1-5*x_c_k1_z^2/r_c_k1_n^2);
                x_c_k1_y*(1-5*x_c_k1_z^2/r_c_k1_n^2);
                x_c_k1_z*(3-5*x_c_k1_z^2/r_c_k1_n^2)];
    a_J2 = -(3*mu*J2*R^2/(2*r_c_k1_n^5))*help_vec;
    if delta_t == 0
        a_control = zeros(3,1);
    else
        a_control = u_k_c / delta_t;
    end
    two_body_acc = -(mu/r_c_k1_n^3)*x_k1(1:3);
    total_acc = two_body_acc + a_J2 + a_control;

    x_c_k1_dot = zeros(6,1);
    x_c_k1_dot(1:3) = x_k1(4:6);
    x_c_k1_dot(4:6) = total_acc;

    f_k_c = x_c_k1 + x_c_k1_dot*delta_t + w_k_c;

    % Deputy dynamics (rQNSOE)
    x_d_k1 = x_k1(7:12);
    u_k_d = u_k(4:6);
    w_k_d = w_k(7:12);
    chief_OE_k1 = utils.ECI2OE(x_k1(1:6), mu);
    Phi_J2_qns_k1 = control_helpers.getQNS_J2_STM(chief_OE_k1, delta_t, mu, J2, R);
    B_k1 = control_helpers.getBcMatrix(chief_OE_k1, mu, J2, R);
    f_k_d = Phi_J2_qns_k1*x_d_k1 + delta_t*B_k1*u_k_d + w_k_d;

    x_k = [f_k_c; f_k_d];
end

function z_k = h_k(x_k, v_k, mu)
    x_c_k = x_k(1:6);
    r_c_k = x_k(1:3);
    v_c_k = x_k(4:6);
    rQNSOE_d_k = x_k(7:12);
    OE_c_k = utils.ECI2OE(x_c_k,mu);
    OE_d_k = utils.rQNSOE2OE(OE_c_k,rQNSOE_d_k);
    x_d_k = utils.OE2ECI(OE_d_k,mu);
    r_d_k = x_d_k(1:3);
    delta_r = r_d_k - r_c_k;
    delta_r_n = norm(delta_r);
    range = sqrt(delta_r' * delta_r);
    delta_r_x = delta_r(1);
    delta_r_y = delta_r(2);
    delta_r_z = delta_r(3);
    azimuth = atan2(delta_r_y, delta_r_x);
    elevation = asin(delta_r_z/delta_r_n);
    z_k = [r_c_k; v_c_k; range; azimuth; elevation] + v_k;
end
```

## A.11 Problem Set 9 Code

FIGURES / PLOTTING / DISPLAY CODE REMOVED FOR BREVITY

```
%% - Reset

clc; clear; close all;

%% Initialization

% Constants
mu = 3.986004418e14;
R = 6378137;
J2 = 1.08262668e-3;

% Chief initial orbital elements
a_0 = 6771000 ;          % semi-major axis [m]
e_0 = 0.0005;            % eccentricity [-]
i_0 = deg2rad(51.64);    % inclination [rad]
W_0 = deg2rad(257);      % RAAN [rad]
w_0 = deg2rad(45);       % argument of perigee [rad]
f_0 = deg2rad(30);       % true anomaly [rad]

% Chief initial state in OE & ECI
initial_state_0_OE   = [a_0, e_0, i_0, W_0, w_0, f_0];
initial_state_0_ECI  = utils.OE2ECI(initial_state_0_OE, mu);
r0_init              = initial_state_0_ECI(1:3);
v0_init              = initial_state_0_ECI(4:6);

% Deputy initial relative quasi-non-singular OE
delta_a        = 0;
delta_lambda   = deg2rad(-3.3773);
delta_e_x      = 0.0007;
delta_e_y      = 0.0007;
delta_i_x      = deg2rad(0.4989);
delta_i_y      = deg2rad(0.7850);
initial_state_1_rQNS_OE = [delta_a; delta_lambda; delta_e_x; delta_e_y; delta_i_x;
    delta_i_y];

% Deputy initial state in OE & ECI
initial_state_1_OE  = utils.rQNSOE2OE(initial_state_0_OE, initial_state_1_rQNS_OE);
initial_state_1_ECI = utils.OE2ECI(initial_state_1_OE, mu);
r1_init             = initial_state_1_ECI(1:3);
v1_init             = initial_state_1_ECI(4:6);

% Compute the distance between the two spacecraft
distance = norm(r1_init - r0_init);
fprintf('Distance between the two spacecraft: %.2f meters\n', distance);

% Desired rQNSOE
epsilon     = 1e-7;
desired_QNSROE = [epsilon; epsilon; epsilon; epsilon; epsilon; epsilon];
tau_diff    = 1e-3;   % Small time step for A_c numerical differentiation (s)
k           = 1e3;    % Scaling factor for P
N_param     = 14;     % Thrust concentration exponent

%% - Generate Ground Truth State

% Orbital period
```

```
T = 2 * pi * sqrt(a_0^3 / mu);

% Simulation parameters
tstart = 0;
tint   = 10;          % Match UKF time step
tend   = 20 * T;
options = odeset('RelTol',1e-12,'AbsTol',1e-12);

% Propagate chief in ECI under J2
perturbated = true;
odefun = @(t, state) propagators.FodePropagator.getStatedot(t, state, mu, R, J2,
    perturbated);
[t, state_ECI_0_p] = ode113(odefun, (tstart:tint:tend)', initial_state_0_ECI, options);

% Precompute A_c and B_c along the chief trajectory
N = length(t);
chief_OE_traj = zeros(N,6);
A_all = zeros(6,6,N);
B_all = zeros(6,3,N);
for idx = 1:N
    state_ECI = state_ECI_0_p(idx,:)';
    chief_OE_traj(idx,:) = utils.ECI2OE(state_ECI, mu);
    Phi_J2_qns = control_helpers.getQNS_J2_STM(chief_OE_traj(idx,:), tau_diff, mu, J2, R);
    A_all(:,:,idx) = (Phi_J2_qns - eye(6)) / tau_diff;
    B_all(:,:,idx) = control_helpers.getBcMatrix(chief_OE_traj(idx,:), mu, J2, R);
end

% Propagate deputy QNS ROE with Lyapunov control
nb_orbit = 2;
k_control_start = find(t >= nb_orbit*T, 1, 'first');
delta_alpha = initial_state_1_rQNS_OE;
delta_alpha_traj = zeros(N,6);
u_traj = zeros(3,N);
V_traj = zeros(N,1);
delta_alpha_traj(1,:) = delta_alpha';
for idx = 1:N-1
    A_c    = A_all(:,:,idx);
    B_c    = B_all(:,:,idx);
    err    = delta_alpha - desired_QNSROE;
    V_traj(idx) = 0.5 * (err' * err);
    if idx >= k_control_start
        P     = control_helpers.getPMatrix(chief_OE_traj(idx,:), delta_alpha,
    desired_QNSROE, k, N_param);
        u_d   = -pinv(B_c) * (A_c*delta_alpha + P*err);
    else
        u_d = zeros(3,1);
    end
    u_traj(:,idx) = u_d;
    delta_alpha = delta_alpha + (A_c*delta_alpha + B_c*u_d)*tint;
    delta_alpha_traj(idx+1,:) = delta_alpha';
end
% Final control and Lyapunov
V_traj(N) = 0.5 * (delta_alpha - desired_QNSROE)' * (delta_alpha - desired_QNSROE);
if N >= k_control_start
    P         = control_helpers.getPMatrix(chief_OE_traj(N,:), delta_alpha, desired_QNSROE
    , k, N_param);
    u_traj(:,N) = -pinv(B_all(:,:,N)) * (A_all(:,:,N)*delta_alpha + P*(delta_alpha -
    desired_QNSROE));
else
```

```matlab
    u_traj(:,N) = zeros(3,1);
end

% Consolidated state
x_c_gt = state_ECI_0_p;
x_d_gt = delta_alpha_traj;
x_d_gt(:,2) = unwrap(x_d_gt(:,2));
x_gt = [x_c_gt, x_d_gt];

% Ground truth cumulative delta-V
deltaV_mag       = sqrt(sum(u_traj.^2,1));
cumulative_deltaV = cumsum(deltaV_mag);
fprintf('Ground Truth Final cumulative deltaV: %.4f m/s\n', cumulative_deltaV(end));

%% - Generate Measurements

z_k = zeros(N,9);
sigma_pos    = 0.001;    % m
sigma_vel    = 0.001;    % m/s
sigma_range  = 0.001;    % m
sigma_angles = 4.848e-3; % rad
R_k = diag([sigma_pos^2*ones(1,3), sigma_vel^2*ones(1,3), sigma_range^2, sigma_angles^2*
    ones(1,2)]);
sqrt_R_k = sqrtm(R_k);

for k = 1:N
    x_k_gt    = x_gt(k,:)';
    z_k_true  = h_k(x_k_gt, zeros(9,1), mu);
    v_k       = sqrt_R_k * randn(9,1);
    z_k(k,:)  = z_k_true + v_k;
end

%% - UKF

n     = 12;  % State dimension
m     = 9;   % Measurement dimension
alpha = 5e-1;
kappa = 0;
beta  = 2;
lambda = alpha^2*(n+kappa) - n;

% Weights
Wm = zeros(2*n+1,1);
Wc = zeros(2*n+1,1);
Wm(1) = lambda/(n+lambda);
Wc(1) = lambda/(n+lambda) + (1 - alpha^2 + beta);
for i = 2:2*n+1
    Wm(i) = 1/(2*(n+lambda));
    Wc(i) = Wm(i);
end

% Process noise covariance
sigma_pos_ECI    = 1e-2;
sigma_vel_ECI    = 1e-3;
sigma_rQNS_dim   = 1e-6;
sigma_rQNS_ang   = 1e-6;
Q_k = diag([sigma_pos_ECI^2*ones(3,1); sigma_vel_ECI^2*ones(3,1); ...
            sigma_rQNS_dim^2; sigma_rQNS_ang; sigma_rQNS_dim^2*ones(2,1); sigma_rQNS_ang*
    ones(2,1)]);
```

```matlab

% Measurement noise covariance
sigma_pos     = 0.001;
sigma_vel     = 0.001;
sigma_range   = 0.001;
sigma_angles = 4.848e-3;
R_k = diag([sigma_pos^2*ones(3,1); sigma_vel^2*ones(3,1); sigma_range^2; sigma_angles^2*
    ones(2,1)]);

% Initial state & covariance
x_hat = [ initial_state_0_ECI + [sigma_pos*randn(3,1); sigma_vel*randn(3,1)];
          initial_state_1_rQNS_OE + [sigma_rQNS_dim*randn; sigma_rQNS_ang; sigma_rQNS_dim*
    randn(2,1); sigma_rQNS_ang*randn(2,1)] ];
P_c = eye(6);
P_d = 1e-1*eye(6);
P   = [P_c, zeros(6); zeros(6), P_d];

% Storage
x_hat_store       = zeros(N,n);
P_store           = zeros(n,n,N);
z_hat_store       = zeros(N,m);
z_post_store      = zeros(N,m);
u_ukf_traj        = zeros(3,N);
V_ukf_traj        = zeros(N,1);
cumulative_deltaV_ukf = zeros(N,1);

for k = 1:N
    delta_t = (k==1) * 0 + (k>1)*tint;

    % Sigma points
    sqrt_P = chol((n+lambda)*P,'lower');
    chi    = [x_hat, x_hat + sqrt_P, x_hat - sqrt_P];

    % Prediction
    chi_pred = zeros(n,2*n+1);
    for i = 1:2*n+1
        if k >= k_control_start
            u_k1 = get_control_Lyapunov(chi(:,i), desired_QNSROE, k, N_param, tau_diff, mu
    , J2, R);
        else
            u_k1 = zeros(6,1);
        end
        chi_pred(:,i) = f_k(chi(:,i), u_k1, zeros(n,1), delta_t, mu, J2, R);
    end
    x_hat_pred = chi_pred * Wm;
    P_pred     = Q_k;
    for i = 1:2*n+1
        diff = chi_pred(:,i) - x_hat_pred;
        P_pred = P_pred + Wc(i)*(diff*diff');
    end

    % Control & Lyapunov
    V_k = 0.5*(x_hat_pred(7:12)-desired_QNSROE)'*(x_hat_pred(7:12)-desired_QNSROE);
    if k >= k_control_start
        u_k = get_control_Lyapunov(x_hat_pred, desired_QNSROE, k, N_param, tau_diff, mu,
    J2, R);
    else
        u_k = zeros(6,1);
    end
    u_ukf_traj(:,k) = u_k(4:6);
    V_ukf_traj(k)   = V_k;
    deltaV_k        = norm(u_k(4:6));
    cumulative_deltaV_ukf(k) = cumulative_deltaV_ukf(max(k-1,1)) + deltaV_k;

    % Update
    sqrt_P_pred = chol((n+lambda)*P_pred,'lower');
    chi_pred_new = [x_hat_pred, x_hat_pred + sqrt_P_pred, x_hat_pred - sqrt_P_pred];
    Z_pred = zeros(m,2*n+1);
    for i = 1:2*n+1
        Z_pred(:,i) = h_k(chi_pred_new(:,i), zeros(m,1), mu);
    end
    z_hat = Z_pred * Wm;
    S_k   = R_k; P_xz = zeros(n,m);
    for i = 1:2*n+1
        z_diff = Z_pred(:,i) - z_hat;
        x_diff = chi_pred_new(:,i) - x_hat_pred;
        S_k      = S_k + Wc(i)*(z_diff*z_diff');
        P_xz     = P_xz + Wc(i)*(x_diff*z_diff');
    end
    K_k      = P_xz / S_k;
    z_k_curr = z_k(k,:)';
    x_hat    = x_hat_pred + K_k*(z_k_curr - z_hat);
    P        = P_pred - K_k*S_k*K_k';

    % Store
    x_hat_store(k,:)  = x_hat';
    P_store(:,:,k)    = P;
    z_post_store(k,:) = h_k(x_hat, zeros(m,1), mu)';
end

fprintf('UKF Final cumulative deltaV: %.4f m/s\n', cumulative_deltaV_ukf(end));

%% - Control Statistics

control_error        = u_traj - u_ukf_traj;
cumulative_deltaV_diff = cumulative_deltaV - cumulative_deltaV_ukf;
lyapunov_diff        = V_traj - V_ukf_traj;

fprintf('Final cumulative Delta V difference (Ground Truth - UKF): %.4f m/s\n',
    cumulative_deltaV_diff(end));

max_control_error = max(abs(control_error), [], 2);
fprintf('Maximum control input errors (RTN):\n');
fprintf('  \\Delta V_r: %.4e m/s\n', max_control_error(1));
fprintf('  \\Delta V_t: %.4e m/s\n', max_control_error(2));
fprintf('  \\Delta V_n: %.4e m/s\n', max_control_error(3));

%% - Functions

function x_k = f_k(x_k1, u_k, w_k, delta_t, mu, J2, R)
    % Chief dynamics
    r = norm(x_k1(1:3));
    help_vec = [x_k1(1)*(1-5*x_k1(3)^2/r^2);
                x_k1(2)*(1-5*x_k1(3)^2/r^2);
                x_k1(3)*(3-5*x_k1(3)^2/r^2)];
    a_J2 = -(3*mu*J2*R^2/(2*r^5))*help_vec;
    a_control = (delta_t>0) * (u_k(1:3)/delta_t);
    total_acc = -(mu/r^3)*x_k1(1:3) + a_J2 + a_control;
```

```matlab

    x_c_dot = [x_k1(4:6); total_acc];
    f_k_c   = x_k1(1:6) + x_c_dot*delta_t + w_k(1:6);

    % Deputy dynamics
    x_d    = x_k1(7:12);
    OE_c   = utils.ECI2OE(x_k1(1:6), mu);
    Phi    = control_helpers.getQNS_J2_STM(OE_c, delta_t, mu, J2, R);
    B_mat  = control_helpers.getBcMatrix(OE_c, mu, J2, R);
    f_k_d  = Phi*x_d + delta_t*B_mat*u_k(4:6) + w_k(7:12);

    x_k = [f_k_c; f_k_d];
end

function z_k = h_k(x_k, v_k, mu)
    r_c = x_k(1:3);
    v_c = x_k(4:6);
    delta = utils.OE2ECI(utils.rQNSOE2OE(utils.ECI2OE(x_k(1:6),mu), x_k(7:12)),mu);
    dr = delta(1:3) - r_c;
    rho = norm(dr);
    az  = atan2(dr(2), dr(1));
    el  = asin(dr(3)/rho);
    z_k = [r_c; v_c; rho; az; el] + v_k;
end

function u_k = get_control_Lyapunov(x_k, desired, k, N_param, tau, mu, J2, R)
    delta = x_k(7:12);
    OE    = utils.ECI2OE(x_k(1:6), mu);
    Phi   = control_helpers.getQNS_J2_STM(OE, tau, mu, J2, R);
    A_c   = (Phi - eye(6))/tau;
    B_c   = control_helpers.getBcMatrix(OE, mu, J2, R);
    P     = control_helpers.getPMatrix(OE, delta, desired, k, N_param);
    err   = delta - desired;
    u_d   = -pinv(B_c)*(A_c*delta + P*err);
    u_k   = [zeros(3,1); u_d];
end
```