Optimizing Satellite Orbit Transfer Efficiency Through Matrix Diagonalization

Athian Nugraha Muarajuang and 13523106[1,2]
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jl. Ganesa 10 Bandung 40132, Indonesia
[1]13523106@std.stei.itb.ac.id, [2]athianbintang@gmail.com

***Abstract*—The purpose of this paper is to present optimization techniques in transfer maneuvers by satellites using matrix diagonalization to enhance maneuver planning so that it is both efficient and precise. Eigenvalue decomposition and state-space representation allows the simplification of orbital dynamics for Δvs for any maneuvers needed during transition. Test results indicate that the smaller Δv values will be achieved by satellites whose orbits have semi-major axes and inclinations similar to one another, as in the case of the ISS group. This validates the methodology in lower fuel costs and greater mission durations. Limitations are found in computational challenges for large constellations, with future directions for research aimed at the introduction of real-time adjustments and environmental inputs.**

***Keywords*—Matrix Diagonalization, Satellite Orbit Transfer, Optimization, Δv Minimization**

I. Introduction

Satellite is crucial for various mission objectives. Some are used for communication, some for navigation, weather monitoring, science research and many more. These are the reason why satellite orbit transfer is also crucial. they play a role on the deployment, maintenance, and operation of satellites by moving them from their initial orbit to their destination orbit. This maneuver is essential To make sure that satellites attain the proper altitude and inclination required to carry out their intended functions efficiently. For instance, communication satellites need to be positioned in geostationary orbits to provide consistent coverage, while Earth observation satellites are often placed in polar orbits for better global monitoring. Similarly, scientific missions may require specific orbits to study certain regions of space or Earth. Orbit transfers also help extend the lifespan of satellites by adjusting their orbits when needed, ensuring they remain operational for as long as possible. This process is a key factor in achieving the success of satellite missions and maximizing their benefits.

The success of moving satellites from one orbit to another depends greatly on precise calculations and improvements. Many factors, such as how much fuel is used, time limits, and the changing conditions in space, are important when planning these moves. One challenge of this process is reducing the Δv needed, which affects how much fuel will be used. By optimizing these transfers, a mission can achieve reduced costs and extended life of the satellite.

New advances in computer methods have greatly improved our ability to optimize satellite orbit transfers. Methods like matrix diagonalization and state-space representation of linear systems help to model and simulate orbital moves more efficiently. For example, the Clohessy-Wiltshire equations helped make exact calculations of relative motion in nearly circular orbits, which increased the accuracy of where satellites are located. Those mathematical techniques not only simplify the understanding of complex orbital motions but also open up possibilities for cooperation and multiple launches.

This paper applies matrix diagonalization to satellite orbit transfers in order to optimize performance. It shows how this can simplify dynamic systems and improve computational efficiency. With the use of eigenvalues and eigenvectors, it demonstrates that state-space models can achieve accurate satellite motion prediction and computation of minimum required velocity changes for a desired transfer. This approach is thus developing an avenue for low-cost and reliable satellite operations. This is very important in both commercial and scientific missions, where accuracy and efficiency are at large stake.

II. Theoretical Foundation

A. Eigenvalues and Eigenvectors

the word “Eigen”, originated from German, meaning “own” or “characteristic”. It describes the own values, (Eigenvalues) and own vectors (Eigenvectors) of a matrix that doesn’t change direction after transformation, only scaling by some factor. For an $n \times n$ matrix A, there is an eigenvalue of matrix A which can be real or complex scalar (λ) such that

$$Ax = \lambda x$$

for some nonzero vector $x \in R^n$. The equation is called the eigenvalue equation and the vector x in the equation is called eigenvector of A corresponding to λ.

B. Characteristic Equation

Based on the eigenvalue equation, eigenvalues and eigenvectors can be solved with this equation:

$$(\lambda I - A)x = 0$$

$x = 0$ is a trivial solution of the equation.

To make it so that $(\lambda I - A)x = 0$ and x is a nonzero vector, this has solutions if and only if λ is a solution of the characteristic equation:

$$det(\lambda I - A) = 0$$

the roots of the equation, λ, are called characteristic roots or eigenvalues.

C. Matrix Diagonalization

Diagonal matrix is an $n \times n$ matrix that consists zero elements all above and below the main diagonal.

$$D = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 3 & 0 \\ 0 & 0 & 0 & 3 \end{bmatrix}$$

Fig. 1 Example of diagonal matrix
source:
https://statlect.com/matrix-algebra/diagonal-matrix

The process of transforming an $n \times n$ matrix into a diagonal matrix that has the same basic characteristics as the original matrix is known as matrix diagonalization. The process of diagonalizing a matrix is the same as determining its eigenvalues, which are the diagonalized matrix's exact elements. Likewise, the new set of axes that correspond to the diagonal matrix is composed of the eigenvectors. A square matrix of A can be diagonalized if there is a matrix P so that $P^{-1}AP$ is a diagonal matrix. In this case, P can diagonalize matrix A, where P is a matrix consists of the eigenvectors of matrix A, and P^{-1} is the matrix inverse of P. With that in mind, a square matrix A can be decomposed:

$$A = PDP^{-1}$$

where D is a diagonal matrix constructed from eigenvalues of matrix A.

D. State-Space Representation of Linear Systems

1) State-Space Formulation

The state-space formulation is a mathematical description of physical systems, specifically dynamic systems, employing matrices. This method is particularly beneficial for systems with multiple inputs and outputs (MIMO). The standard representation of the state-space model is characterized by

$$\dot{X}(t) = AX(t) + BU(t)$$

Where:

- $X(t)$: State vector, representing the insternal state of the system. For example, the state vector include positions (x, y, z) and velocities $(\dot{x}, \dot{y}, \dot{z})$
- A: System matrix, defines the interaction among state variables.
- $U(t)$: Input vector, representing external control or influence.
- B: Input matrix, maps how inputs affect the state variables
- $\dot{X}(t)$: Derivative of the state vector, representing how the systems evolves over time.

2) Matrix Exponential Solution

The matrix exponential is a pivotal concept in linear algebra, crucial for solving systems of linear differential equations in dynamic systems. In your research paper about optimizing satellite orbit transfer efficiency, it plays a key role in elucidating the satellite's state evolution under the linearized equations of motion while in orbit.

The general solution is

$$X(t) = e^{(At)}X(0),\ X(0) = X_0$$

Where:

- $X(t)$: State vector at time.
- $X(0)$: Initial state vector.
- $e^{(At)}$: State-transition matrix, which describes the time evolution of the system.

The matrix exponential $e^{(At)}$ is defined as an infinite series:

$$e^{(At)} = I + At + (At)^2/2! + (At)^3/3! + ...$$

where:

- I: Identity matrix.
- At: Powers of A scaled by time t. This series always converges, ensuring $e^{(At)}$ is well-defined.

If A is diagonalizable, the calculation of $e^{(At)}$ is simplified to:

$$e^{(At)} = Ve^{Dt}\, V^{-1},$$

where:

- V: Matrix of eigenvectors of A
- D: Diagonal matrix of eigenvalues of A.
- e^{Dt}: Exponential of the diagonal matrix, computed element-wise as:

$$e^{Dt} = diag(e^{\lambda_1 t}, e^{\lambda_2 t}, ..., e^{\lambda_n t}),$$

where λ_i are eigenvalues of A.

E. Clohessy–Wiltshire (CW) Equations

A fundamental framework for comprehending the relative movement of two objects in orbit is offered by the Clohessy–Wiltshire equations. One object(the target) travels in a circle in this scenario, while the other (the chaser) may travel in an elliptical or circular orbit. These formulas provide a preliminary approximation of the chaser's path as seen from the target's point of view. During a rendezvous, they are especially helpful in creating moves that bring the chaser and the target close together.

1) Linearized Second-Order Equations

The CW Equations are given by:

$$\ddot{x} - 3n^2x = 0,$$
$$\ddot{y} + n^2y + 2n\dot{x} = 0,$$
$$\ddot{z} + n^2z = 0$$

Where:

- x, y, z: Relative positions of the deputy satellite in the radial, tangential, and out-of-plane directions.
- $\dot{x}, \dot{y}, \dot{z}$: Relative velocities of the deputy satellite.
- $\ddot{x}, \ddot{y}, \ddot{z}$: Relative accelerations of the deputy satellite.
- $n = \sqrt{\frac{\mu}{a^3}}$: Mean motion of the chief satellite, where:
    - μ: Gravitational parameter of the central body.
    - a: Semi-major axis of the chief satellite's orbit.

2) State-Space Conversion

To facilitate numerical computations, the CW equations are converted into first-order state-space form:

$$\dot{X} = AX,$$

Where:

- $X = [x, y, z, \dot{x}, \dot{y}, \dot{z}]^T$: State vector containing the relative positions and velocities.
- A: State-space system matrix, given by:

$$A = \begin{bmatrix} 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 3n^2 & 0 & 0 & 0 & 2n & 0 \\ 0 & 0 & 0 & -2n & 0 & 0 \\ 0 & 0 & -n^2 & 0 & 0 & 0 \end{bmatrix}.$$

Fig. 2 State-Space System matrix source: Author

This formulation allows the relative motion to be solved using matrix exponential techniques.

F. Orbit Transfer Optimization

Orbit transfer is one of the most essential concepts in satellite motion, whereby a satellite is transferred from one orbit to another. This process generally requires velocity changes (Δv), which are performed by means of propulsion maneuvers. It is therefore very important to optimize such maneuvers in order to minimize fuel consumption, increase the mission's lifetime, and perform precise orbit transfers. The theory of orbit transfer is a matter of understanding how orbits work, calculating the Δv required for different transfer methods, and techniques such as matrix diagonalization to speed things up.

In optimization of orbit transfers, the system matrix A in the CW framework is written as $A = PDP^{-1}$, where P has the eigenvectors of A, and D is a diagonal matrix made up of its eigenvalues. This transformation splits the dynamics into a set of independent pieces with each one corresponding to an eigenvalue.

Using the diagonalized form, the state-transition matrix $e^{(At)}$ is computed as:

$$e^{\mathbf{A}t} = \mathbf{V}e^{\mathbf{D}t}\mathbf{V}^{-1},$$

where e^Dt is trivial to compute since D is diagonal. This decoupling allows for efficient exact computation of the state of the satellite at any time t and design of simple maneuvers to reach some desired state. The consequence is a dramatic reduction in the Δv required, making the transfer between the orbits much more efficient.

III. Implementation

The implementation in this paper is done with Python, alongside a set of libraries. The numerical computations and matrix operations are handled through NumPy library, while the SGP4 API processes TLE data to compute positions and velocities of satellites. It depends on the Requests library to fetch real-time TLE data from Celestrak and Math for its trigonometric and general mathematical functions required for orbital calculation. JSON is used both for storing data in structured format and for outputting results. With its clear syntax, extended scientific ecosystem, and painless integration with a lot of different data sources, Python is an ideal choice for implementing and reproducing satellite orbit transfer optimization.

A. Dataset Description

The dataset that is used for the program are Two-Line Element (TLE) from a file named “stations.txt”. This file has data from Celestrak, an online resource about Earth-orbiting satellites and their orbital elements, describing the orbit for many satellites and space-related objects. The file organizes the data in groups of three lines:

- Line 1: The satellite name
- Line 2: TLE (line 1)
- Line 3: TLE (line 2)

This is the example of entry data inside stations.txt

```
ISS (ZARYA)
1 25544U 98067A   24366.81307813
.00027763  00000+0  48311-3 0  9996
2 25544  51.6382  56.2039 0006091
28.0496 332.0820 15.50544003489205
```

The breakdown in the format of the data is described below:

- Line 1: Provides the name of the satellite (ISS (ZARYA))
- Line 2: Contains data about satellite’s orbit:
    - `1`: Line number in TLE
    - `25544`: Satellite catalog number
    - `U`: Classification

    - `98067A`: Launch year and piece identifier
    - `24366.81307813`: Epoch time, representing the time the TLE was valid.
    - `.00027763`: Orbital drag coefficient.
    - `00000+0`: Placeholder for ballistic coefficient.
    - `48311-3`: Placeholder for higher-order drag terms.
    - `0`: Checksum digit
- Line 3: Contains the actual orbital elements:
    - `2`: Line number in TLE
    - `51.6382`: Inclination (i), the angle of the orbital plane relative to Earth's equatorial plane.
    - `56.2039`: Right Ascension of the Ascending Node (RAAN)
    - `0006091`: Eccentricity (e), representing the orbit's deviation from circularity.
    - `28.0496`: Argument of Perigee (ω), specifying the closest point's location in the orbit.
    - `332.0820`: Mean anomaly (M), describing the satellite's position within its orbit.
    - `15.50544003`: Mean motion (n), indicating the number of orbits per day.
    - `489205`: Checksum digit.

TLE data is significant for the program. Specifically, such information aids in determining a satellite's position and velocity utilizing the SGP4 model; this then goes a step further to obtain orbital elements such as the semi-major axis, eccentricity, inclination, and right ascension of the ascending node. These orbital elements are useful in generating the Clohessy–Wiltshire (CW) system matrix, which describes the motion of the satellite in a near-circular orbit. Additionally, the TLE data aids in the proper identification of each satellite, where names from the dataset are used to label and organize results. With such clear and accurate data, the program can calculate and enhance orbital transfers for a number of satellites effectively.

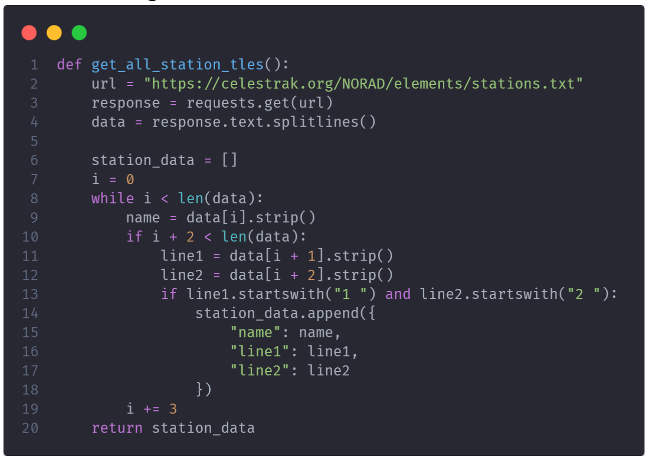
B. Program Procedure

Below is a written procedure of the code with each corresponding code snippet for the implementation.

1. Retrieving TLE Data for Satellites

The first step is retrieving TLE data for satellites from stations.txt file. This is accomplished by utilizes the requests library to fetch the data from https://celestrak.org/NORAD/elements/stations.txt. The program extracts the data line by line. Validation added to ensure that TLE first line starts with "1" and second line starts with"2". The data entries are structured into a list of dictionaries, where each dictionary consists od satellite's name and its corresponding TLE lines.

Below is the source code for the process of retrieving TLE data:



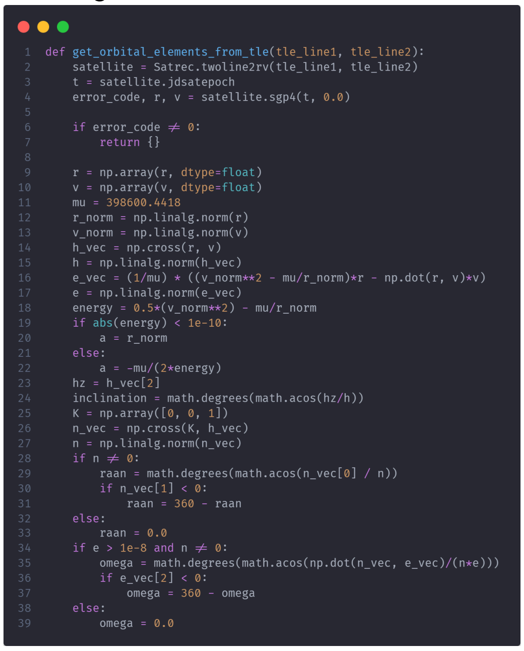
```python
def get_all_station_tles():
    url = "https://celestrak.org/NORAD/elements/stations.txt"
    response = requests.get(url)
    data = response.text.splitlines()

    station_data = []
    i = 0
    while i < len(data):
        name = data[i].strip()
        if i + 2 < len(data):
            line1 = data[i + 1].strip()
            line2 = data[i + 2].strip()
            if line1.startswith("1 ") and line2.startswith("2 "):
                station_data.append({
                    "name": name,
                    "line1": line1,
                    "line2": line2
                })
        i += 3
    return station_data
```

Fig. 3 Retrieve TLE data algorithm
source: https://github.com/Starath/MakalahAlgeo

2. *Converting TLE Data to Orbital Elements*

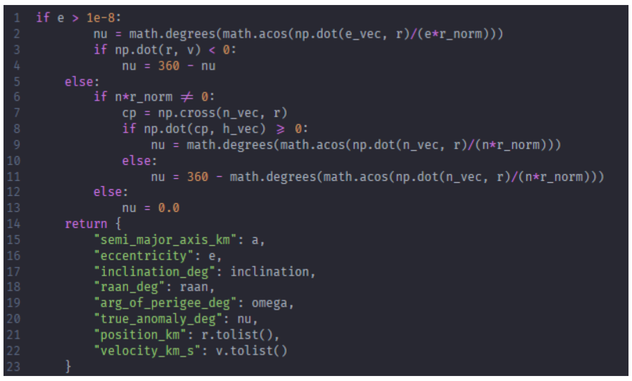
The retrieved TLE data is then proceeds to be converted into orbital elements. The two TLE lines that corresponds to each satellite are then parsed using the SGP4 library, resulting in position (r) and velocity (v) vectors at epoch time for each satellite. From these vectors, the main orbital parameters are computed: semi-major axis (a), eccentricity (e), inclination (i), and Right Ascension of Ascending Node (RAAN). The argument of perigee (ω) and true anomaly (ν) are also computed, completing the set of orbital elements. These parameters mathematically describe the orbit of a satellite. The function also handle errors and will return an empty dictionary when TLE propagation for any of the satellites fails, ensuring only valid orbital elements are passed.

Below is the source code for the process of converting the TLE data:



```python
def get_orbital_elements_from_tle(tle_line1, tle_line2):
    satellite = Satrec.twoline2rv(tle_line1, tle_line2)
    t = satellite.jdsatepoch
    error_code, r, v = satellite.sgp4(t, 0.0)

    if error_code ≠ 0:
        return {}

    r = np.array(r, dtype=float)
    v = np.array(v, dtype=float)
    mu = 398600.4418
    r_norm = np.linalg.norm(r)
    v_norm = np.linalg.norm(v)
    h_vec = np.cross(r, v)
    h = np.linalg.norm(h_vec)
    e_vec = (1/mu) * ((v_norm**2 - mu/r_norm)*r - np.dot(r, v)*v)
    e = np.linalg.norm(e_vec)
    energy = 0.5*(v_norm**2) - mu/r_norm
    if abs(energy) < 1e-10:
        a = r_norm
    else:
        a = -mu/(2*energy)
    hz = h_vec[2]
    inclination = math.degrees(math.acos(hz/h))
    K = np.array([0, 0, 1])
    n_vec = np.cross(K, h_vec)
    n = np.linalg.norm(n_vec)
    if n ≠ 0:
        raan = math.degrees(math.acos(n_vec[0] / n))
        if n_vec[1] < 0:
            raan = 360 - raan
    else:
        raan = 0.0
    if e > 1e-8 and n ≠ 0:
        omega = math.degrees(math.acos(np.dot(n_vec, e_vec)/(n*e)))
        if e_vec[2] < 0:
            omega = 360 - omega
    else:
        omega = 0.0
```

Fig. 4.1 Convert TLE data algorithm
source: https://github.com/Starath/MakalahAlgeo



```python
if e > 1e-8:
        nu = math.degrees(math.acos(np.dot(e_vec, r)/(e*r_norm)))
        if np.dot(r, v) < 0:
            nu = 360 - nu
    else:
        if n*r_norm ≠ 0:
            cp = np.cross(n_vec, r)
            if np.dot(cp, h_vec) ⩾ 0:
                nu = math.degrees(math.acos(np.dot(n_vec, r)/(n*r_norm)))
            else:
                nu = 360 - math.degrees(math.acos(np.dot(n_vec, r)/(n*r_norm)))
        else:
            nu = 0.0
    return {
        "semi_major_axis_km": a,
        "eccentricity": e,
        "inclination_deg": inclination,
        "raan_deg": raan,
        "arg_of_perigee_deg": omega,
        "true_anomaly_deg": nu,
        "position_km": r.tolist(),
        "velocity_km_s": v.tolist()
    }
```

Fig. 4.2 Convert TLE data algorithm
source: https://github.com/Starath/MakalahAlgeo

3. *Construct and Diagonalize the Clohessy–Wiltshire (CW) System Matrix*

The Clohessy–Wiltshire (CW) system matrix represents the linearized relative motion of a satellite in a near-circular orbit. Using the construct_cw_matrix() function, a 6×6 CW matrix A is created based on the mean notion:

$$n = \sqrt{\mu / a^3}$$

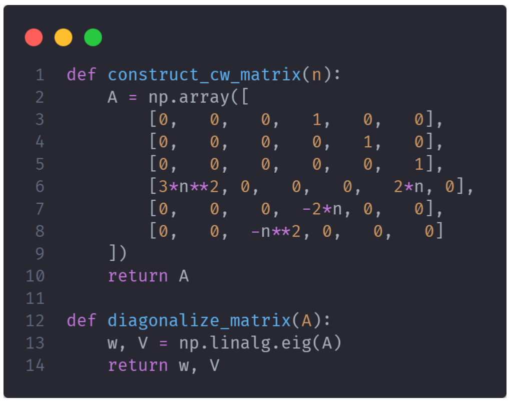
Where $\mu = 398600.4418 km^3/s^2$ is Earth's gravitational parameter and a is the semi-major axis.

Following the construction, the CW system matrix decomposed, yielding eigenvalues and eigenvectors such that

$$A = PDP^{-1}$$

This diagonalization decouples the system dynamics into independent modes, simplifying the computation of state corrections for optimizing the orbit transfer.

Below is the source code for the process of construct and diagonalize the CW matrix:



```python
def construct_cw_matrix(n):
    A = np.array([
        [0,   0,   0,   1,   0,   0],
        [0,   0,   0,   0,   1,   0],
        [0,   0,   0,   0,   0,   1],
        [3*n**2, 0,   0,   0,   2*n, 0],
        [0,   0,   0,  -2*n, 0,   0],
        [0,   0,  -n**2, 0,   0,   0]
    ])
    return A

def diagonalize_matrix(A):
    w, V = np.linalg.eig(A)
    return w, V
```

Fig. 5 Construct and diagonalize CW matrix
source: https://github.com/Starath/MakalahAlgeo

4. *Optimizing the Orbit Transfer*

After the construction and diagonalization of the Clohessy–Wiltshire CW system matrix are performed, the program optimizes the orbit transfer of the satellite. the program calculates the minimum velocity impulse (Δv) that the satellite must impart in order to change from an initial offset state (X_0) to a desired final state (X_f) in a given time of flight (t_f). The process transforms the initial and final states into modal space using eigenvectors (V). It lets the initial state roam freely over the time interval using eigenvalues (Λ). Then, it determines the required state correction in modal space. This correction is transformed back to physical space to obtain the required velocity impulse. Below are the equations governing these transformations:

$$\mathbf{X}_d = V^{-1} \cdot \mathbf{X}_0$$

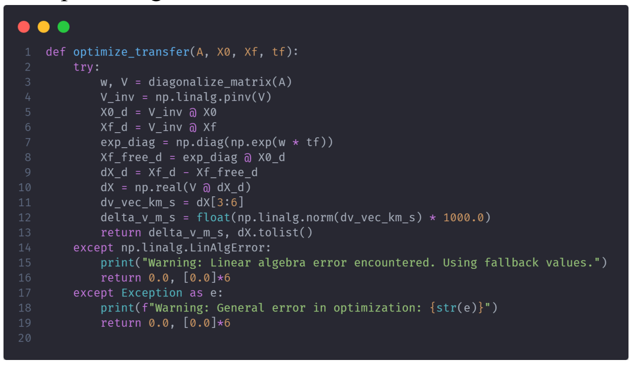
$$\mathbf{X}_{f,\text{free}} = e^{\Lambda t_f} \cdot \mathbf{X}_d$$

$$\Delta \mathbf{X}_d = \mathbf{X}_f - \mathbf{X}_{f,\text{free}}$$

$$\Delta \mathbf{X} = V \cdot \Delta \mathbf{X}_d$$

The magnitude of the velocity impulse (Δv) is then calculated from the velocity components of the state correction. This optimized Δv ensures that the satellite reaches the desired final state efficiently within the allotted time.

Below is the source code for the process of optimizing the orbit transfer:



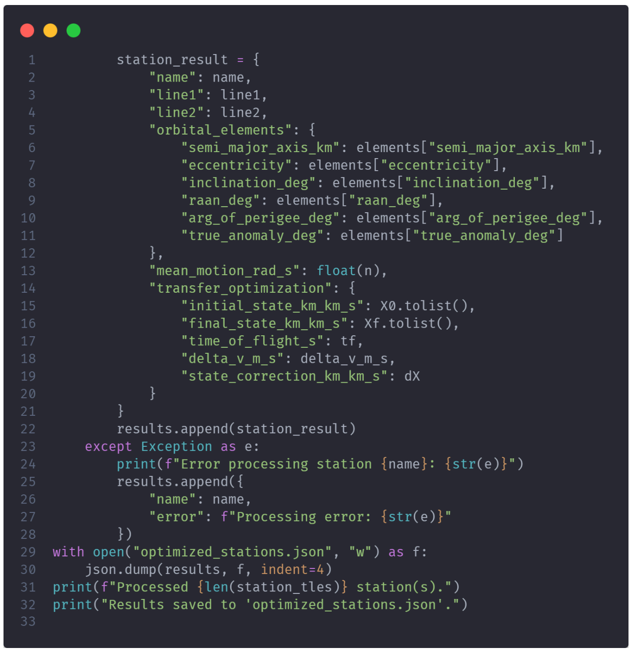
```python
def optimize_transfer(A, X0, Xf, tf):
    try:
        w, V = diagonalize_matrix(A)
        V_inv = np.linalg.pinv(V)
        X0_d = V_inv @ X0
        Xf_d = V_inv @ Xf
        exp_diag = np.diag(np.exp(w * tf))
        Xf_free_d = exp_diag @ X0_d
        dX_d = Xf_d - Xf_free_d
        dX = np.real(V @ dX_d)
        dv_vec_km_s = dX[3:6]
        delta_v_m_s = float(np.linalg.norm(dv_vec_km_s) * 1000.0)
        return delta_v_m_s, dX.tolist()
    except np.linalg.LinAlgError:
        print("Warning: Linear algebra error encountered. Using fallback values.")
        return 0.0, [0.0]*6
    except Exception as e:
        print(f"Warning: General error in optimization: {str(e)}")
        return 0.0, [0.0]*6

```

Fig. 6 Optimize orbit transfer
source: https://github.com/Starath/MakalahAlgeo

5. *Saving Optimization Results*

After the program has optimized the orbit transfers for all satellites, it formats the results and stores them onto optimized_stations.json file. The file contains detailed information for each satellite, such as its name, TLE data, derived orbital elements, mean motion, and details of the transfer improvement, including starting and ending states, time of flight, the calculated speed change (Δv), and the state adjustment needed. The program uses the json library to ensure that the results are stored in a neat and readable JSON format for subsequent data processing or visualization operations.

Below is the source code for the process of saving the optimization results:



```python
        station_result = {
            "name": name,
            "line1": line1,
            "line2": line2,
            "orbital_elements": {
                "semi_major_axis_km": elements["semi_major_axis_km"],
                "eccentricity": elements["eccentricity"],
                "inclination_deg": elements["inclination_deg"],
                "raan_deg": elements["raan_deg"],
                "arg_of_perigee_deg": elements["arg_of_perigee_deg"],
                "true_anomaly_deg": elements["true_anomaly_deg"]
            },
            "mean_motion_rad_s": float(n),
            "transfer_optimization": {
                "initial_state_km_km_s": X0.tolist(),
                "final_state_km_km_s": Xf.tolist(),
                "time_of_flight_s": tf,
                "delta_v_m_s": delta_v_m_s,
                "state_correction_km_km_s": dX
            }
        }
        results.append(station_result)
    except Exception as e:
        print(f"Error processing station {name}: {str(e)}")
        results.append({
            "name": name,
            "error": f"Processing error: {str(e)}"
        })
with open("optimized_stations.json", "w") as f:
    json.dump(results, f, indent=4)
print(f"Processed {len(station_tles)} station(s).")
print("Results saved to 'optimized_stations.json'.")
```

Fig. 7 Saving optimization results
source: https://github.com/Starath/MakalahAlgeo

IV. Experimental Results

Here is the results of optimized orbit transfer from each satellite

Name	Semi-major Axis (km)	Eccentricity	Inclination (°)	Delta V (m/s)
ISS (ZARYA)	6796.58	0.001671	51.648	1.5274
CSS (TIANHE)	6765.1	0.001021	41.47	1.5301
ISS (NAUKA)	6795.69	0.000982	51.644	1.5275
FREGAT DEB	7943.6	0.094076	51.64	1.3954
CSS (WENTIAN)	6769.54	0.001268	41.488	1.5297
CSS (MENGTIAN)	6769.54	0.001268	41.488	1.5297
CYGNUS NG-21	6795.69	0.000982	51.644	1.5275
CYSAT-1	6648.21	0.001828	51.614	1.5396
YODAKA	6765.81	0.002053	51.62	1.5301
1998-067XD	6768.12	0.001608	51.618	1.5299

Fig. 8 Optimized result data
source: https://github.com/Starath/MakalahAlgeo

The semi-major axes range from about 6648 km (like CYSAT-1) to almost 7944 km (FREGAT DEB). This may indicate that most of the vehicles either operate or have been deorbited in LEO, while there is a higher, more oval orbit for one of them. The inclination values go from 41.5° (CSS modules) to approximately 51.6° (ISS modules), thus showing the selection of rather different orbital paths for these missions. The overall eccentricities remain quite small (generally < 0.01), except for the Fregat Deb fragment, whose eccentricity is close to 0.094, highlighting its significantly elliptical orbit.

The transfer optimization part of the data shows that all simulated maneuvers were designed for a 900-second (15-minute) time of flight. The needed Δv values usually stay around 1.52 m/s for most LEO targets (like ISS segments, CSS modules, and visiting vehicles). This means they need similar low-thrust or short-burn maneuvers in the same types of orbits. The main exception is FREGAT DEB, which has a lower Δv of about 1.40 m/s. This is probably because its higher starting altitude and orbital shape change the details of the transfer design. Furthermore, the final "state_correction" vectors show small positional and velocity offsets, on the order of fractions of a kilometer and millimeters per second, to be corrected after the primary burn—emphasizing that each maneuver is finely tuned to reduce residual insertion error.

Taken together, these results underscore how mission design in LEO can converge on similar magnitudes of maneuver requirements when orbital regimes, inclinations, and target altitudes are nearly aligned. The similarity between ISS-related spacecraft and CSS modules shows that they are designed to have the same orbital height and angle. This is done to make rendezvous and docking easier. On the other hand, differences in detail, like those of FREGAT DEB, show how elliptical orbits can change the energy needed, even for short transfers. In general, the data shows that although mission goals differ, the basic Δv needs and short maneuver times of about 900 seconds are largely the same for LEO operations.

V. Conclusion

In this work, it showed how satellite orbit transfers could be optimized in terms of minimizing the required speed changes (Δv) for accurate maneuvers using matrix diagonalization. The approach via eigenvalue decomposition and state-space transformation provided effective solutions for orbit adjustments for different satellites with various orbital settings. The results showed that the satellites having close semi-major axes and inclinations to each other, like the ones in the ISS group, had very low Δv values, even down to 1.527 m/s. This again proves how matrix-based optimization techniques can lower fuel consumption and hence increase the life span of a satellite mission.

The limitation of the proposed method is that large constellations cannot be handled because high computational power is needed for such conditions. Work in the future should attempt to adopt parallel processing techniques and address non-linear changes, thus making it useful in various mission scenarios. Further, investigating real-time applications and updating the model to include dynamic environmental factors such as atmospheric drag and solar radiation pressure would likely continue to make it even more accurate and mission-applicable. This study sets up the basics for effective ways to move in space, which is important for planning satellite missions and saving money.

VI. Appendix

the JSON result file and the full source code are in this github repository below:
https://github.com/Starath/MakalahAlgeo

VII. Acknowledgement

The author would like to express his deepest gratitude to God Almighty who has helped make this paper possible. The author would also like to thank the lecturer of the Linear Algebra and Geometry course, Dr. Ir. Rinaldi Munir M.T., who has been fully dedicated to teaching this course and helping the author in understanding the foundation of Linear Algebra and Geometry so that it greatly helps the author in applying it to this paper.

References

[1] Grainger College of Engineering. (2019). “Eigenvalues and eigenvectors”. https://courses.grainger.illinois.edu/cs357/fa2019/references/ref-9-eigen/. Accessed 1st January 2025.

[2] Munir, Rinaldi. 2023. “Nilai Eigen dan Vektor Eigen (Bagian 1)”. Nilai Eigen dan Vektor Eigen (Bagian 1). Accessed 1st January 2025.

[3] Munir, Rinaldi. 2023. “Nilai Eigen dan Vektor Eigen (Bagian 2)”. Nilai Eigen dan Vektor Eigen (Bagian 2). Accessed 1st January 2025.

[4] Weisstein, Eric W. "Matrix Diagonalization." From MathWorld--A Wolfram Web Resource. https://mathworld.wolfram.com/MatrixDiagonalization.html. accessed 1st January 2025.

[5] CelesTrak. (n.d.). *Two-Line Element Sets*. Retrieved from https://celestrak.org/NORAD/elements/. Accessed 1st January 2025

[6] Rabah, Rabah & Bergeon, Benoit. (2001). On state space representation of linear discrete-time systems in Hilbert spaces. Visnyk Kharkivs’kogo Universytetu. Seriya Matematyka, Prykladna Matematyka i Mekhanika. 514. Accessed 1st January 2025.

STATEMENT

Hereby i declare this that this paper i have written is my own writing, not a reproduction or translation of someone else’s paper, and not plagiarized.

Bandung, 2 January 2025

[illegible]

Athian Nugraha Muarajuang
13523106