Mathematical Analysis of 3D Rotations and Principal Axes

Gabriel Maldonado

December 14, 2024

Abstract

This paper covers the mathematical and computational aspects of 3D rotations and principal axes. Topics include solving ordinary differential equations (ODEs), orthogonal projections, quaternions, transformation matrices, and rotation methods like roll-pitch-yaw and Rodrigues' formula. Comparisons and applications are provided to highlight the versatility of these techniques in physics, engineering, and computer graphics.

Contents

1 Introduction **2**

2 Mathematical Foundations **3**

2.1 Euler's Angles 3

2.1.1 Rotation Matrices 3

2.1.2 Applications of Euler's Angles 5

2.1.3 Visualization of Euler's Angles 6

2.2 Rodrigues' Formula 7

2.2.1 Skew-Symmetric Matrix Representation 8

2.2.2 Applications of Rodrigues' Formula 8

2.2.3 Comparison with Roll-Pitch-Yaw Representations . . . 9

2.2.4 Visualization of Rodrigues' Formula 9

2.3 Quaternions 10

3 Numerical Methods for Solving ODEs **10**

4 Visualization and Simulation Techniques **11**

5 Challenges and Limitations **12**

6 Conclusions **13**

1 Introduction

The study of of 3D rotations is a foundation of modern computational techniques and understanding of topics used in fields such as aerospace engineering, robotics and computer graphics. Rotations form the basis for describing the orientation and movement of objects in three-dimensional space, from animating a character in a video game, to guiding a robotic arm or component, to the rotational dynamics of orbital satellites. Understanding and being able to utilize these rotations requires a combination of mathematical precision, computational efficiency, and a strong understanding of various physics/engineering considerations.

This paper focuses on the computational implementation and mathematical analysis of 3D rotations considering an emphasis on Euler's angles, transformation matrices, including Rodrigues' rotation formula, solving nonlinear ordinary differential equations (ODE's) using numerical methods such as Runge-Kutta 45 (RK45) implemented by Python's "solve ivp" toolbox. These methods provide crucial insights into the role of principal axis and it's rotational stability which are fundamental aspects of understanding the distribution of rotational inertia within rigid objects often referred to as bodies.

The main objective of studying 3D rotations of rigid objects is to develop various algorithms that visualize and simulate the rotation of an object while also simplifying the process and demonstrating the complex nature of 3D rotations. The use of comparing different rotation techniques—such as roll-pitch-yaw systems and Rodrigues' formula—this paper analyzes each methods strengths, limitations, while understanding the advantages in practical applications.

The structure of this paper is as it follows. Section 2 outlines the mathematical foundations behind the 3D rotations, including an analysis of Euler's angles, transformation matrices, and Rodrigues' formula. Sections 3 discusses numerical methods for solving ODE's, containing an interest mainly on Rk45 with a central focus on their application to rotational dynamics. Section 4 is compiled of visualization and simulation techniques incorporated to represent these mathematical models with a physical view. Finally, section 5 highlights the challenges and limitations that came up throughout the process concluding with areas dedicated to direction of future work.

2 Mathematical Foundations

2.1 Euler's Angles

Euler's angles are a set of three angles that describe the orientation of a rigid body in three-dimensional space. They are commonly used in area such as physics, computer graphics, and engineering—common in mechanics and aeronautical applications—to represent rotational motion. The three angles are:

- **Roll** (ϕ): Rotation around the x-axis.
- **Pitch** (θ): Rotation around the y-axis.
- **Yaw** (ψ): Rotation around the z-axis.

These three angles can be used to define the orientation of an objects relative to a fixed coordinate system, typically being the x, y, and z system. The order in which rotations are applied is crucial and must remain a serious consideration, as rotating an object rotating around different axes in different sequences will vary in with different results. Euler's angles are typically represented using the Tait-Bryan angles—the angle make up of the roll-pitch-yaw systemwhere the rotations are applied in the order of a yaw (ψ), pitch (θ), and roll (ϕ).

2.1.1 Rotation Matrices

The rotation matrices corresponding to the three basic rotations—roll, pitch, and yaw—in the counter-clockwise direction can be derived as follows. First consider the x, y, z coordinate system by drawing out the system allowing the axis of interest coming out of the paper or board allowing for the perpendicular or top view of the other two axes' plane. Next draw three vectors, each being the three unit vectors $\hat{\imath}$, $\hat{\jmath}$, and $\hat{k}$. Next take the same coordinate system with the two axes' visible with the top view and draw the associated unit vectors again, but rotated by some angle—ϕ for $\hat{\jmath}$ and $\hat{k}$ in the yz plane, θ for $\hat{\imath}$ and $\hat{k}$ in the xz plane, ψ for $\hat{\imath}$ and $\hat{\jmath}$ in the yz plane—while labeling the angle from the x, y, or z axes to the new vectors drawn as the respective angle regarding to the plane of rotated unit vectors. Next using trigonometry techniques regarding right triangles, write the new vector of the now rotated unit vectors being the vectors $\hat{\imath}'$, $\hat{\jmath}'$, or $\hat{k}'$. Finally, combining those new vectors each as a column of a 3×3 matrix with the resulting matrix being your rotation matrix in the clockwise direction. These matrices are used to transform the coordinates of a point in 3D space. The three rotation matrices are as follows:

Roll (ϕ): The rotation around the x-axis is represented by the matrix:

$$R_x(\phi) = \begin{pmatrix} 1 & 0 & 0 \\ 0 & \cos(\phi) & -\sin(\phi) \\ 0 & \sin(\phi) & \cos(\phi) \end{pmatrix}$$

Pitch (θ): The rotation around the y-axis is represented by the matrix:

$$R_y(\theta) = \begin{pmatrix} \cos(\theta) & 0 & \sin(\theta) \\ 0 & 1 & 0 \\ -\sin(\theta) & 0 & \cos(\theta) \end{pmatrix}$$

Yaw (ψ): The rotation around the z-axis is represented by the matrix:

$$R_z(\psi) = \begin{pmatrix} \cos(\psi) & -\sin(\psi) & 0 \\ \sin(\psi) & \cos(\psi) & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

With each of these rotation matrices being the rotation of an object about one axis with there being three axes, combining the matrices in together, typically $R_z(\psi)R_y(\theta)R_x(\phi)$, to perform the rotations in the yaw-pitch-roll sequence. There are then applied to a system/collection of vertices of an object in three-dimensions, say P to rotate the points in P to get a new set of rotated points P' of same size as P in the way:

$$P' = R_z(\psi)R_y(\theta)R_x(\phi)P$$

2.1.2 Applications of Euler's Angles

Euler's angles are widely used in the simulation of rigid body motion and are fundamental in the study of angular velocities and accelerations. In robotis, they are used to control in which ways the robotic arms are rotating and moving and therefore controlling the orientation of such arms their end effectors which allow them to perform tasks. In aerospace engineering and many flight oriented aspects Euler's angles help to model the orientation an aircraft or a spacecraft.

The way Euler's angles are effective in rotational dynamics are how they can be applied in creating a powerful system representing rotational motion of objects based on their angular mechanic properties. Euler's angles are interpreted into a system that allows for the analysis of inertia and rotational properties of an object in reference to its principal axes with a relation to the space the object remains in or in reference to another coordinate system with different orientations. This partnership of reference frames of orientation allows for an easier calculation of inertia when it comes to the object basing the calculations based on only the initial calculations. This is a powerful tool for the realm of physics and engineering alike.

This relationship of the different reference frames regarding the different orientations of the object to the space it's in allows for the creation of what is known as Euler's equations. Creating this relation by incorporating the derivative of angular momentum ($\overrightarrow{L}$)—considering angular momentum is equal to the product of angular velocity (ω) and the inertia (I) of the body: $\overrightarrow{L} = I\overrightarrow{\omega}$—with respect to another frame, being the frame of the space where the object is rotating. Relating this derivative to the space frame using:

$$\left(\frac{d\overrightarrow{L}}{dt}\right)_{space} = \left(\frac{d\overrightarrow{L}}{dt}\right)_{body} + \overrightarrow{\omega} \times \overrightarrow{L}$$

$$\overrightarrow{\tau} = \dot{\overrightarrow{L}} + \overrightarrow{\omega} \times \overrightarrow{L}$$

Further calculation and simplification leads to the formulas we know of as Euler's equations which incorporates angular velocities (ω), accelerations ($\dot{\omega}$), eternal torques (τ or M), and inertia (I) of the body. These create a system of three non linear differential equations being:

$$\begin{aligned}
I_1\dot{\omega} + (I_3 - I_2)\omega_2\omega_3 &= M_1 \\
I_2\dot{\omega} + (I_1 - I_3)\omega_3\omega_1 &= M_2 \\
I_3\dot{\omega} + (I_2 - I_1)\omega_1\omega_2 &= M_3
\end{aligned}$$

Euler's angles are a useful tool for rotations, however one limitation of Euler's angles is the phenomenon known as **gimbal lock**, which occurs when the any of the three angles ψ, θ, ϕ are rotated by $\pm 90°$. In this event two of the axes of rotation become aligned, leading to the loss of one of the rotation axis further losing a degree of freedom in the rotational representation. This issue can be avoided while considering an alternative method such as quaternions or the use of rotation matrices.

2.1.3 Visualization of Euler's Angles

Euler's angles are often visualized as a sequence of rotations about the x, y, and z axes. To picture it, consider a rigid body that undergoes rotations in the order of a yaw rotation (around the z axis) first, then a pitch rotation (around the y axis), and finally a roll rotation (around the x axis). Each of there rotations will employ a change on the orientation of the body, which then having the cumulative effect being the total orientation relative to the fixed reference fame.

In computer animation and computer graphics, Euler's angles incorporate with algorithms to simplify the way to represent rotations and are often used in conjunctions with other techniques such as quaternions and rotations matrices for more practical purposes of rotations. To visualize these rotations, consider the diagram below, which illustrates the rotations corresponding to roll, pitch, and yaw.

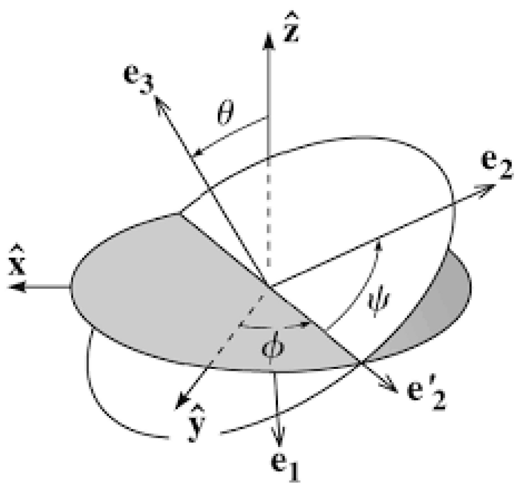

Figure 1: Visualization of Euler's angles: Roll (ϕ), Pitch (θ), and Yaw (ψ).

2.2 Rodrigues' Formula

Rodrigues' rotation formula is a rotation formula incorporating matrices similar to rotation matrices from earlier in this paper and is an efficient method for rotating a vector in three-dimensional space around a specified axis by a given angle. This formula is useful in computer graphics, robotics and other applications requiring real-time 3D transformations. Comparing to Euler's angles, which describes rotations using three sequential angles, Rodrigues' formula directly computes the rotated vector using the axis-angle representation of the the vector. The Rodrigues' formulais expressed as:

$$\mathbf{v}' = \mathbf{v}\cos\theta + (\mathbf{k} \times \mathbf{v})\sin\theta + \mathbf{k}(\mathbf{k} \cdot \mathbf{v})(1 - \cos\theta),$$

where:

- $\mathbf{v}$ is the vector to be rotated,
- $\mathbf{v}'$ is the rotated vector,
- $\mathbf{k}$ is the unit vector representing the axis of rotation,
- θ is the angle of rotation (in radians),
- $\mathbf{k} \cdot \mathbf{v}$ is the dot product,

- $\mathbf{k} \times \mathbf{v}$ is the cross product.

Rodrigues' formula combines three components to compute the rotated vector: This being the projection of $\mathbf{v}$ onto the axis $\mathbf{k}$, the orthogonal projection of $\mathbf{v}$ in the rotation plane, and a component that accounts for the rotation in that plane. This decomposition ensures an accurate and computationally efficient rotation.

2.2.1 Skew-Symmetric Matrix Representation

Rodrigues' formula can alsobe expressed using what is called a skew-symmetric matrix $\mathbf{K}$ derived from the axis of rotation $\mathbf{k}$. The matrix $\mathbf{K}$ is defined as:

$$\mathbf{K} = \begin{pmatrix} 0 & -k_z & k_y \\ k_z & 0 & -k_x \\ -k_y & k_x & 0 \end{pmatrix}.$$

The rotated vector can then be computed as:

$$\mathbf{v}' = \mathbf{v} + \sin\theta(\mathbf{K}\mathbf{v}) + (1 - \cos\theta)(\mathbf{K}^2\mathbf{v}).$$

This matrix formulation is helpful for implementing Rodrigues' formula in numerical computations, as it avoids the explicit use of trigonometric functions for vector components. In using this formula and the partnered skew-matrix in the case of rotational dynamics and specifically Euler's equations, each $\mathbf{k}$ in K are parts in a unit vector $\mathbf{k}$ which is comprised by normalizing the solutions of the three ODE's that is Euler's equations. Normalizing is taking some vector in this case the solution to Euler's equations and dividing the vector by its magnitude. As mentioned the skew-matrix is an excellent tool incorporating Euler's equations due to its nature of requiring a numerical method to solve ODE's.

2.2.2 Applications of Rodrigues' Formula

Rodrigues rotation formula is used in applications that require precise 3D transformations. In computer graphics, the formula gets used to animate objects and creating smooth rotations around arbitrary axes. Similarly, it's used in areas such as robotics in controlling the orientation robotic arms and their end effectors, and in aerospace engineering to produce simulations of satellite orientation and projected motion to control.

2.2.3 Comparison with Roll-Pitch-Yaw Representations

The Rodrigues' formula differs from the roll-pitch-yaw representation, which requires sequential rotations about coordinate axes, by the way it directly computes the rotated vector of a collection of points or a point relating to some object, by using the axis-angle pair. This method of representing rotation allows for the system to avoid issues such as gimbal lock, therefore always keeping the three degrees of freedom. Additionally, the formula is computationally more efficient for scenarios requiring rotations that are arbitrary.

2.2.4 Visualization of Rodrigues' Formula

The geometric interpretation of the Rodrigues' formula involves visualizing the process of rotating a vector around an axis within the 3D space. This approach provides a method of understanding how the formula decomposes a rotation into components along an axis and perpendicular to the same axis. Interactive models incorporating 3D tools in both Python and MATLAB can be used to further aid in demonstrating its method of rotating objects. Another diagram to visualize the derivation of the Rodrigues' formula follows:

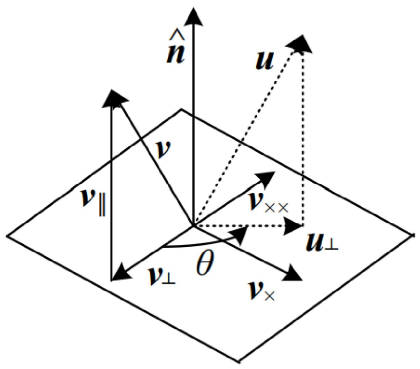

Figure 2: Geometric interpretation of Rodrigues' rotation formula.

2.3 Quaternions

Quaternions offer a compact and efficient method for representing 3D rotations. A quaternion has four components:

$$q = q_0 + q_1 i + q_2 j + q_3 k,$$

where q_0 is the real (scalar) part, and (q_1, q_2, q_3) form the imaginary (vector) part. Unit quaternions, with norm equal to 1, are commonly used for rotations:

$$q = \cos\left(\frac{\theta}{2}\right) + \mathbf{u}\sin\left(\frac{\theta}{2}\right),$$

where θ is the angle of rotation and $\mathbf{u}$ is the unit axis of rotation.

Quaternions have an advantage over Euler's angles and equations as they avoid gimbal lock, and over rotation matrices being their better efficiency since they don't require the same nine parameters as rotation matrices. They also allow smooth interpolation of rotations, known as spherical linear interpolation (SLERP), which makes them ideal for computer graphics and robotics with arm orientation and rotational movement.

Some key properties quaternions posses include:

- Quaternions can be multiplied to combine rotations.
- The conjugate of a quaternion, $q^* = q_0 - q_1 i - q_2 j - q_3 k$, reverses the rotation.
- Rotating a vector $\mathbf{v}$ can be achieved using: $q\mathbf{v}q^{-1}$.

Although quaternions were not covered in depth in this project, their ability to represent rotations without singularities or excessive computations makes them a powerful tool in rotational analysis.

3 Numerical Methods for Solving ODEs

In the context of 3D rotations specifically Euler's equations, solving ordinary differential equations (ODE's) is crucial for modeling any kind of time-dependent systems so they go hand and hand. This makes Euler's equations the perfect candidate for the use of a method that solves ODE's numerically.

Euler's equations describe the time evolution of angular velocity, acceleration, and torque in a rotational system. ODEs such as these equations often take the form:

$$\frac{d\mathbf{y}}{dt} = f(t, \mathbf{y}),$$

where $\mathbf{y}$ represents the system state (e.g., angular position or velocity) and $f(t, \mathbf{y})$ is a function that defines how the state changes over time.

To solve this type of ODE numerically, the Runge-Kutta 45 method (RK45) used by Python's "solve ivp" toolbox is useful. The RK45 method is a combination of Runge-Kutta 4 (RK4) and Runge-Kutta 5 (RK5) and improves on the these standard Runge-Kutta methods by adapting the step size to further minimize the local error. This is due to RK5's more accurate, slower calculation time and RK4's less accurate, faster calculation time Rk45 utilizes both methods to create an overall more efficient version. The way RK45 incorporates the two methods is as follows:

The method evaluates the solution using four intermediate slopes (denoted k_1, k_2, k_3, k_4) to estimate the next value of $\mathbf{y}$:

$$\mathbf{y}_{n+1} = \mathbf{y}_n + \frac{h}{6}\left(k_1 + 2k_2 + 2k_3 + k_4\right),$$

where h is the step size and k_1 through k_4 are the slopes evaluated at different points in the interval. The method also computes a fifth-order approximation k_5 to estimate the error and adjust the step size accordingly. This adaptive feature ensures a balance between accuracy and efficiency.

RK45 is particularly useful in rotational dynamics, as it provides a stable and accurate way to solve the system of ODEs representing the rotational motion of rigid bodies, especially over long time intervals. This pairs excellent with Euler's equations to efficiently adapt the solutions into simulation efforts and visualization of data.

4 Visualization and Simulation Techniques

Visualization is has an important role in understanding the dynamics of 3D rotations behavior, and an analysis of principal axes. Using computer simulations to simulate and visualize rotations creates a different aspect of understanding rotations. This allows for the ability to observe and analyze the behavior of a rigid object in space under various conditions. This further allows for comparison testing and optimization techniques. In this project, simulations were implemented using Python's Pygame toolbox to better view animation to model the behavior and rotational dynamics of objects using graphical tools with the incorporation of the various rotations methods discussed.

These methods were incorporated with various algorithms that created the rotation matrices, Euler's equations, and Rodrigues' formula components such as the skew-matrix. Then, algorithms were implemented that further calculated various scenarios using these methods while translating them to the visualization using graphical techniques such as orthogonal projections. The translation of mathematical theory involved the analysis of rigid objects regarding the objects components changing over time such as the angular velocity different time intervals.

Simulations of rotational dynamics and rotational motion frequently require numerical methods, which attributes to the use of Runge-Kutta methods such as RK45 in approximating the solution to Euler's equations. Using these numerical methods create the ability to simulate the continuous rotational motion of an object over various time intervals. These results from simulating rotation can then be displayed using 3D graphics libraries within Python and MATLAB allowing for an interactive view and method of leaning about rotational behavior.

The combination of each parts covered within this paper allow for an overarching collaboration to create proper visualization and simulation algorithms with objectives to understand the data simulated, and create overall motion of the object.

5 Challenges and Limitations

Throughout the project several challenges were encountered that influenced the depth and scope of the analysis of rotational dynamics. One of the main difficulties was the computational complexity with translating between various coordinate systems which required careful handling of rotational matrices and transformations. The complex nature of numerical methods created a barrier for understanding the true behavior of rotations of rigid objects, luckily being supplemented with graphical visuals and graph-

ical data to observe.
Other limitations where the incorporation of more simulation testing involving more rigid bodies, better analysis of inertia calculations, better analysis of center of mass calculations, more focus on every possible factor involved in rotational dynamics of a rigid body, and more in depth data collection. Part of this limitation was a lack of in depth coverage of quaternions and Lie groups, which are powerful tools for describing rotations and transformations in higher-dimensional space. Quaternions being a powerful tool, excellent for its many useful factors regarding rotational dynamics should have been leveraged more in the final analysis of the project.

From a coding perspective, having the mathematical methods and theory implemented effectively was a successful process, but the overall efficiency of such algorithms and methods used within the project were potentially lacking with the possibility of improvement. The already complex nature of rotational dynamics created a even larger layer of complexity incorporating the coding side of all the math theory and methods. This may have scarified accuracy and effective use of memory and storage.

Despite these challenges and limitations, the project successfully demonstrated the key principles of 3D rotations and the importance of the mathematical theory and methods behind them. Future work could include a more in depth look at quaternions with comparisons to the other methods mentioned, and more broad coverage of rotations geared towards more realistic scenarios.

6 Conclusions

This project aimed to create a foundation of mathematical methods and practical applications of 3D rotations and principal axes, while focusing on Euler's angles, equations, rotation matrices, Rodrigues' formula, and numerical methods necessary such as Runge-Kutta 45 method which allowed for accurate analysis of the behavior of rotating rigid objects, effectively showing the visual and simulations of scenarios regarding different applications. The strengths and limitations of the methods employed were highlighted and displayed to provide some insight into where each method might contribute to which areas.

Visualization and simulation techniques created the ideal scenario for viewing and understanding the rotations creating an overall better analysis of rotating rigid objects in three-dimensions. This methods of visual and simulating rotations was only possible with the integration of each mathematical topic covered throughout this paper. Mathematical methods and theory combined with the computational power creates a strong collaboration that creates a better understanding of reality such as rotational dynamics. Overall, this project offers valuable contributions to the study of 3D rotations and createsa basis for further on going research into these methods and comparisons of other methods while creating further opportunity to optimize these methods for more accurate and efficient simulations.

References

[1] Weisstein, Eric W. *Euler Angles*, MathWorld–A Wolfram Web Resource

[2] LibreTexts, *Euler's Equations of Motion*, Physics LibreTexts,

[3] Orthogonal Projection, *Georgia Tech Mathematics Textbook*

[4] Mathoma, *3D Rotations in General: Rodrigues Rotation Formula and Quaternion Exponentials*

[5] Sabetta Talks Math, *Euler's equations — Chapter 26 Classical Mechanics 2*