****

**Cairo University**

**Faculty of Engineering  
Aerospace Department**

**Task (2)**

NUMERICAL SOLUTION OF ODE (RK4) AIRPLANE SIMULATOR PART I

**Prepared by:** Team 15

|  |  |  |
| --- | --- | --- |
| **Name** | **Sec** | **Bn** |
| Abdallah Ahmed Abdallah | 1 | 31 |
| Mohamed Rajeh Mohamed | 2 | 10 |
| Mohamed Rabie Walaan | 2 | 11 |
| Ahmed Ramadan Kamal | 1 | 2 |
| Mohamed Ahmed Mohamed Khalil | 2 | 5 |
| Mohamed Emad Mahmoud | 2 | 20 |

**Submitted To: Eng. Mahmoud Elewa**

# **Table of contents:**

[**Table of contents:** 2](#_Toc191318111)

[Research questions: 6](#_Toc191318112)

[**Autopilot** 6](#_Toc191318113)

[**first autopilots invention** 7](#_Toc191318114)

[**inputs & outputs of an Autopilot system** 8](#_Toc191318115)

[Inputs : 8](#_Toc191318116)

[1.Flight Path Command: 8](#_Toc191318117)

[. Desired altitude 8](#_Toc191318118)

[. Desired heading orcourse (yaw) 8](#_Toc191318119)

[.Desired speed or Mach number 8](#_Toc191318120)

[. Desired vertical speed (climb rate or descent rate) 8](#_Toc191318121)

[. Desired attitude (pitch and roll angles) 8](#_Toc191318122)

[. Desired flight mode (e.g., altitude hold, navigation mode, approach mode) 8](#_Toc191318123)

[2.Sensors Feedback (for closed-loop control): 8](#_Toc191318124)

[. Current aircraft altitude 8](#_Toc191318125)

[. Current heading (yaw) 8](#_Toc191318126)

[. Current speed 8](#_Toc191318127)

[. Current vertical speed 8](#_Toc191318128)

[. Current attitude (pitch, roll) 8](#_Toc191318129)

[. Airspeed and angle of attack (if needed for specific control tasks) 8](#_Toc191318130)

[. Flight plan data (waypoints, approach, etc.) 8](#_Toc191318131)

[. Navigation data (GPS or inertial navigation systems) 8](#_Toc191318132)

[Autopilot buttons and controls : 9](#_Toc191318133)

[**the role of the pilot in an airplane equipped with an autopilot** 10](#_Toc191318134)

[**the difference between an Autopilot & SAS (stability augmentation system)** 10](#_Toc191318135)

[Stability Augmentation System (SAS): 10](#_Toc191318136)

[Autopilot : 10](#_Toc191318137)

[**the role of the onboard sensors** 11](#_Toc191318138)

[Mathematical Modeling 12](#_Toc191318139)

[Soft Sensors (Virtual Sensors) 12](#_Toc191318140)

[Observers 12](#_Toc191318141)

[Machine Learning-Based Estimation 12](#_Toc191318142)

[Kalman Filters and Variants 12](#_Toc191318143)

[**fly-by-wire flight control system** 12](#_Toc191318144)

[**open source autopilot softwares used on UAVs** 14](#_Toc191318145)

[**autopilot hardwares used on UAVs** 15](#_Toc191318146)

[VECTOR-600 : 15](#_Toc191318147)

[Pixhawk Series: 15](#_Toc191318148)

[Flight Mechanics review 17](#_Toc191318149)

[**the general rigid body dynamics (RBD) equations in 3D space.** 17](#_Toc191318150)

[**Classification of the airplanes ( EOM ) equations mathematically** 25](#_Toc191318151)

[**the (Body axes) versus the (earth or inertial axes)** 25](#_Toc191318152)

[**the pitch angle (θ) versus the angle of attack (α)** 26](#_Toc191318153)

[**attitude representations** 28](#_Toc191318154)

[**1. Euler Angles (ϕ,θ,ψ)** 28](#_Toc191318155)

[**2. Direction Cosine Matrix (DCM)** 29](#_Toc191318156)

[**3. Quaternions (q0​,q1​,q2​,q3​)** 29](#_Toc191318157)

[**4. Axis-Angle Representation (r,θ)** 30](#_Toc191318158)

[Numerical solution of ODEs 31](#_Toc191318159)

[**the numerical solving algorithms for ODEs** 31](#_Toc191318160)

[**Choosing an algorithm for solving the Airplanes EOM,** 32](#_Toc191318161)

[**Solving the system** 32](#_Toc191318162)

[**MATLABresults** 33](#_Toc191318163)

[**As shwon above, the RK4 function RK4 simulink produced the same results as the MATLAB function “ODE45”** 34](#_Toc191318164)

[**Problem definition** **Error! Bookmark not defined.**](#_Toc191318165)

[Givens : **Error! Bookmark not defined.**](#_Toc191318166)

[Initial conditions (𝑆.𝐼. 𝑢𝑛𝑖𝑡𝑠) : **Error! Bookmark not defined.**](#_Toc191318167)

[**Error! Bookmark not defined.**](#_Toc191318168)

[**RBD Equations in vector form:** **Error! Bookmark not defined.**](#_Toc191318169)

[**Figures** 39](#_Toc191318170)

[40](#_Toc191318171)

[**2. Root Mean Squared Error (RMSE)** 42](#_Toc191318172)

[42](#_Toc191318173)

[ Useful for understanding deviations in real-world terms. 42](#_Toc191318174)

[**3. Mean Absolute Error (MAE)** 42](#_Toc191318175)

[42](#_Toc191318176)

[ Less sensitive to outliers compared to MSE. 42](#_Toc191318177)

[**4. Normalized Root Mean Squared Error (NRMSE)** 42](#_Toc191318178)

[43](#_Toc191318179)

[ Useful for comparing errors across different datasets. 43](#_Toc191318180)

[**5. Pearson Correlation Coefficient (r)** 43](#_Toc191318181)

[43](#_Toc191318182)

[ Values close to 1 indicate strong correlation. 43](#_Toc191318183)

[**6. Signal-to-Noise Ratio (SNR)** 43](#_Toc191318184)

[43](#_Toc191318185)

[ Higher values indicate better similarity. 43](#_Toc191318186)

[**7. Cross-Correlation** 43](#_Toc191318187)

[43](#_Toc191318188)

[**MATLAB code** 44](#_Toc191318189)

[Time Vector Initialization 44](#_Toc191318190)

[External Forces and Moments 44](#_Toc191318191)

[Aircraft Mass and Inertia 44](#_Toc191318192)

[Initial Conditions 44](#_Toc191318193)

[Runge-Kutta 4th Order (RK4) 45](#_Toc191318194)

[Simulink RK4 45](#_Toc191318195)

[Convert Euler Angles to Degrees 45](#_Toc191318196)

[Plot Results 45](#_Toc191318197)

[**Appendix B** 46](#_Toc191318198)

[**Raungekutta4th order function** 46](#_Toc191318199)

[**get\_states\_dot Function** 46](#_Toc191318200)

[**plotting Function** 47](#_Toc191318201)

[**Appendix C** 48](#_Toc191318202)

[**Simulink model** 48](#_Toc191318203)

[**References :** 49](#_Toc191318204)

Research questions:

## **Autopilot definition and main objective**

An autopilot is a software or tool that can only manage theaircraft under certain conditions using the vehicle's hydraulic, mechanical and electronic systems. This system, which can follow the flight plan, can stabilize speed and height as well as the location of the front of the aircraft (heading). Pilots mostly lead the aircraft in a controlled manner by autopilot except for departure and landing. Autopilot is mostly used on passenger aircrafts.(It's main objective is managing the aircraft under certain conditions using the vehicle's hydraulic, mechanical andelectronic systems).

The autopilot can take part in most of the control mechanisms except takeoff. In general, it controls the movement of the aircraft around the center of gravity and directs the aircraft according to safety parameters. Route data prepared before the flight is uploaded to this software. From the moment the autopilot is instructed by the pilot, it controls the aircraft within this route. Planes; can have three different types of autopilot software: one-axis, two-axis, and three-axis. The next-generation aircraft can be guided by improved three-axis autopilots. New generation autopilots can also direct the yaw by controlling the rudder along with rotation and reclining movements. In newer systems, the autopilot can perform most of the classic flight maneuvers. The climbing flight and the descent flight are guided by the pilots except in extreme cases. Autopilot performs all operations in accordance with the pilot's commands.



## **first autopilot invention**

The first autopilot, or autopilot, was developed in 1912 by the Sperry Corporation by the inventor Elmer Ambrose Sperry, co-inventor with the German physicist Hermann Anschütz-Kaempfe of the gyro-compass, a type of modern gyroscope which indicates geographic north.

The system itself was quite simple: a gyro altitude indicator and compass were connected to the rudder, lifts and ailerons, operated hydraulically (the hydraulic system moves some actuators that, in turn, activate the three main types of control and control surfaces: tail rudder, tail lifts and wing flaps, which allow the aircraft to change direction). In addition, the system was supplemented by ADF, radiomagnetic locator, or radioaid (requiring NDB radio beacons on the ground) so that the aircraft was maintained at constant speed in straight and level flight (no yaw, no pitch, no twist) without the pilot putting hands on the joystick.The system proved its operation in a real flight in 1914. In 1920 it was first employed on a ship.

****



## **inputs & outputs of an Autopilot system**



## Inputs :

## 1.Flight Path Command:

## . Desired altitude

## . Desired heading orcourse (yaw)

## .Desired speed or Mach number

## . Desired vertical speed (climb rate or descent rate)

## . Desired attitude (pitch and roll angles)

## . Desired flight mode (e.g., altitude hold, navigation mode, approach mode)

## 2.Sensors Feedback (for closed-loop control):

## . Current aircraft altitude

## . Current heading (yaw)

## . Current speed

## . Current vertical speed

## . Current attitude (pitch, roll)

## . Airspeed and angle of attack (if needed for specific control tasks)

## . Flight plan data (waypoints, approach, etc.)

## . Navigation data (GPS or inertial navigation systems)

Outputs (Actuator Outputs):

1.Primary Control Surfaces:

.Elevator: Controls the pitch of the aircraft (up and down movement of the nose).

.Ailerons: Control the roll of the aircraft (banking the wings).

. Rudder: Controls the yaw of the aircraft (left or right movement of the nose).

2.Throttle (Engine Power):

Adjusts the engine thrust, controlling the aircraft's speed and rate of climb or descent.

3.Autopilot Steering Commands:

The autopilot computes the necessary adjustments to the flight controls (elevator, ailerons, rudder, and throttle) based on the desired flight parameters. These commands are sent to the aircraft’s flight control system, which drives the corresponding actuators.

## Autopilot buttons and controls :

FD: navigation display

HDG: allows to set the course, essential to navigate precise routes.

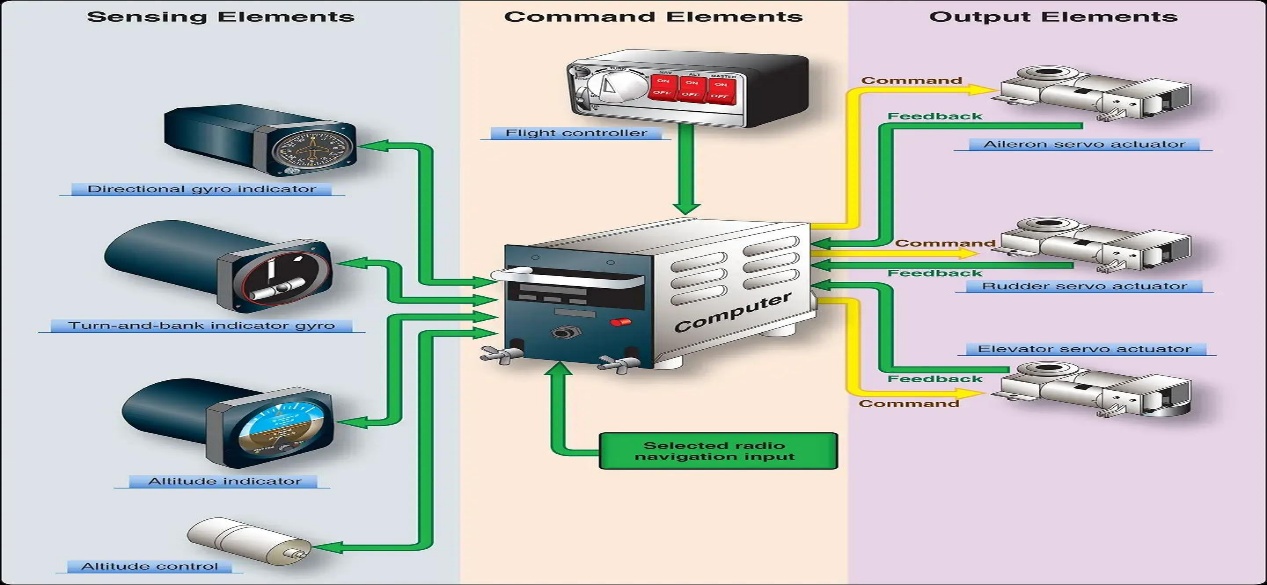
NAV: browser

ALT: tells the pilot how to track altitude, crucial for airspace management.

HDG Knob: to control direction

APPR: for approach

LVL: keeps the plane level

SPD MACH: Activates speed control, essential for efficient fuel management and flight duration.

## **The role of the pilot in an airplane equipped with an autopilot**

Before takeoff, the pilot will enter the route into the computer, giving it a start and end position and exactly how to get there. Throughout that route there are a series of points that the computer will note, each having its own speed and altitude.

The autopilot does not steer the airplane on the ground or taxi the plane at the gate. Generally, the pilot will handle takeoff and then initiate the autopilot to take over for most of the flight. In some newer aircraft models, autopilot systems will even land the plane.

**the difference between an Autopilot & SAS (stability augmentation system)**

## Stability Augmentation System (SAS):

is an inertially stabilized platform that maintains an Aero plane or helicopter's altitude and heading in a fixed position. Originally conceived as an SAS for helicopters to relieve pilot fatigue when hovering for an extended amount of time.

## Autopilot :

Many of the same functions as an SAS-equipped aircraft will be performed by autopilots. Autopilot in helicopters would function similarly to SAS. However, autopilot has a number of other features, like altitude hold. Hold your airspeed. Tracking the VOR (Localizer approaches) ILS techniques that are linked.

Point-to-point navigation, "R" navigation capabilities, Flight Management Systems. Autopilots have grown in sophistication over time, and will continue to do so as aeronautical and navigation engineers enhance their products.

Stability Augmentation System (SAS) differs from autopilot systems, which take full control of the aircraft and plan entire flights, as SAS focuses solely on maintaining stable flight without directing the aircraft's path. This system is crucial for both manned and unmanned aircraft to ensure safety and pilot control.

## **The role of the onboard sensors**

|  |  |  |
| --- | --- | --- |
| **Sensor** | **Quantities they measure** | **Typical sampling rates** |
| **GPS** | **Position (Latitude, Longitude, Altitude), Velocity** | **1 – 10 Hz** |
| **Gyroscope** | **Angular velocity (roll, pitch, yaw)** | **100 – 1000 Hz** |
| **Pitot Tube**  **(Airspeed Sensor)** | **Airspeed (Dynamic pressure)** | **10 – 50 Hz** |
| **Magnetometer** | **Heading (Magnetic North direction)** | **10 – 100 Hz** |
| **Accelerometer** | **Linear acceleration (X, Y, Z) axes** | **10 – 1000 Hz** |
| **LIDAR /**  **Rader** | **Altitude**  **(above ground)**  **Obstacle direction** | **1 – 100 Hz** |
| **Optical Flow Sensor** | **Motion relative to the ground** | **10 – 200 Hz** |
| **Sonar / Ultrasonic Sensor** | **Proximity to the ground (for landing)** | **10 – 50 Hz** |

**Estimation techniques**

If a state is not directly measured by a sensor, we can estimate it using state estimation techniques.

### Mathematical Modeling

Use known system dynamics (e.g., differential equations) to predict the unmeasured state based on measured inputs and outputs

### Soft Sensors (Virtual Sensors)

Combine multiple sensor readings with models to infer the unmeasured state.

### Observers

Luenberger Observer (for linear systems)

Nonlinear Observers (for nonlinear systems)

Sliding Mode Observers (for robust estimation in uncertain environments)

### Machine Learning-Based Estimation

Use neural networks or regression models trained on historical data to predict the state.

### Kalman Filters and Variants

Kalman Filter (KF) (for linear systems with Gaussian noise)

Extended Kalman Filter (EKF) (for nonlinear systems)

Unscented Kalman Filter (UKF) (for better nonlinear performance)

Particle Filter (for highly nonlinear and non-Gaussian systems).

## **fly-by-wire flight control system**

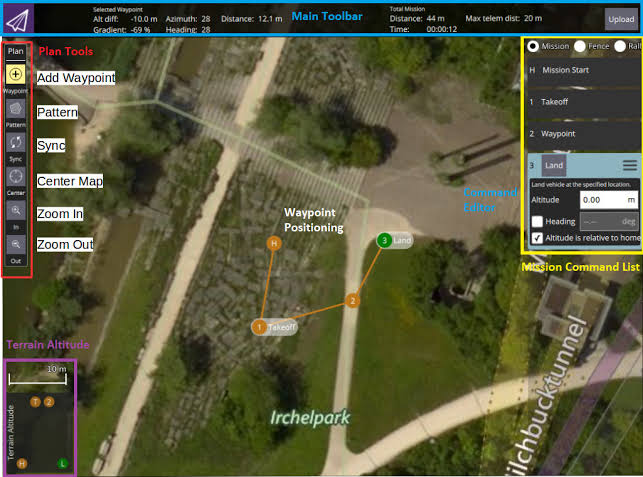
Fly-by-wire the flight control systems which use computers to process the flight control inputs made by the pilot or autopilot, and which send corresponding electrical signals to the flight control surface actuators.

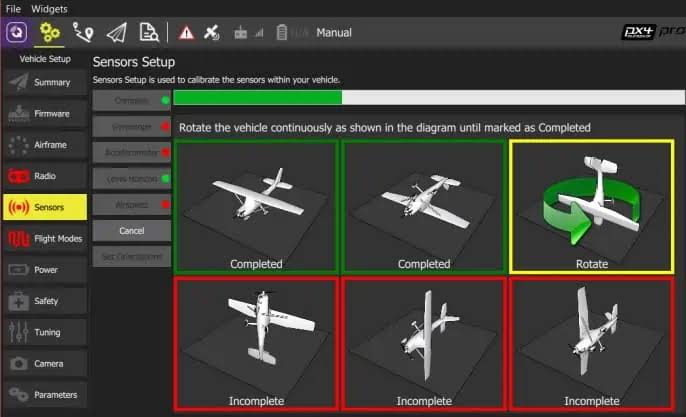
Fly-by-Wire (FBW) is the generally accepted term for those flight control systems which use computers to process the flight control inputs made by the pilot or autopilot, and send corresponding electrical signals to the flight control surface actuators. This arrangement replaces mechanical linkage and means that the pilot inputs do not directly move the control surfaces. Instead, inputs are read by a computer that in turn determines how to move the control surfaces to best achieve what the pilot wants in accordance with which of the available Flight Control Laws is active.

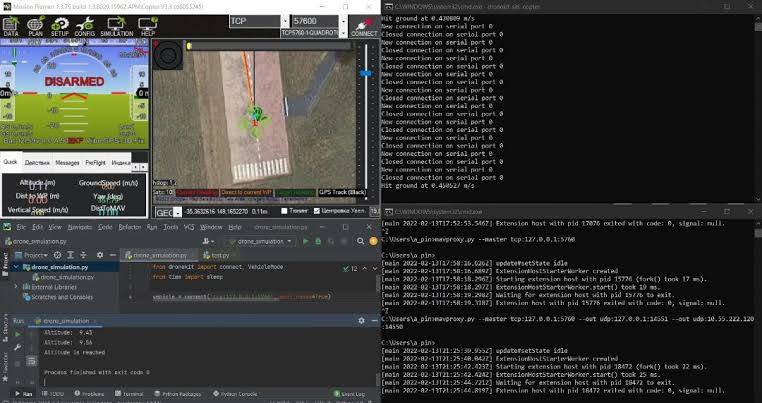
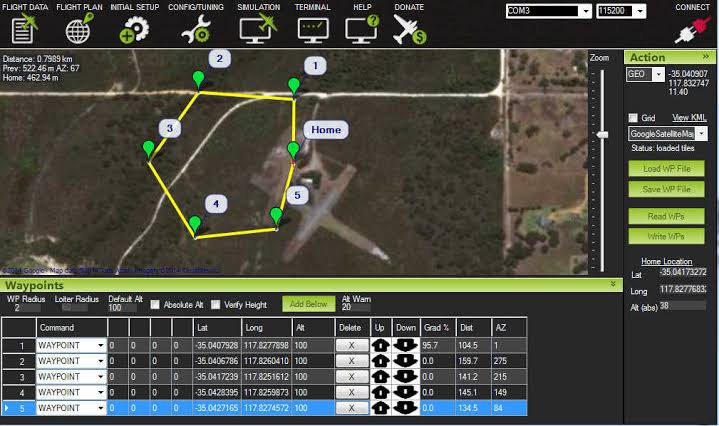
The advantages of reduced weight, improved reliability, damage tolerance, and more effective control of a necessarily highly manoeuverable aircraft, were first recognised in military aircraft design. The first aircraft to have FBW for all its flight controls in place of direct mechanical or hydraulically-assisted operation, was the F-16 in 1973. In the context of military fast jet need for agility, and therefore relatively more unstable aircraft, FBW provides the ability to ensure that unintended increases in angle of attack or sideslip are detected and rapidly, and automatically, resolved by marginally deflecting the control surfaces in the opposite way while the problem is still small. FBW also enables highly reliable flight envelope protection systems which, provided the FBW system functions at its normal level, significantly enhances safety.



## **open source autopilot softwares used on UAVs**

A number of autopilot software packages are open-source, including PX4, Dronekit and ArduPilot which are both used widely in the drone industry. Flight controller hardware may be optimized for use with one or more particular autopilot software stacks.

****



## **autopilot hardwares used on UAVs**

## VECTOR-600 :

The VECTOR-600 is UAV Navigation's most advanced autopilot. VECTOR-600 is a robust and dependable unit, with built-in physical and logical redundancy, allowing it to survive all individual sensor failures while maintaining accurate estimates of attitude and position. This dependability has helped the VECTOR-600 rapidly to become the autopilot of choice for UAV professionals who require the most advanced and reliable autopilot for fixed wing, rotary wing and VTOL UAVs.

## Pixhawk Series:

Silicon Errata

Pixhawk Standard/Supported Autopilots:

Holybro Pixhawk 4 (FMUv5)

Holybro Pixhawk 4 Mini (FMUv5)

Drotek Pixhawk 3 Pro (FMUv4pro)

mRo Pixracer (FMUv4)

Hex Cube Black (FMUv3)

CUAV Pixhack v3 (FMUv3)

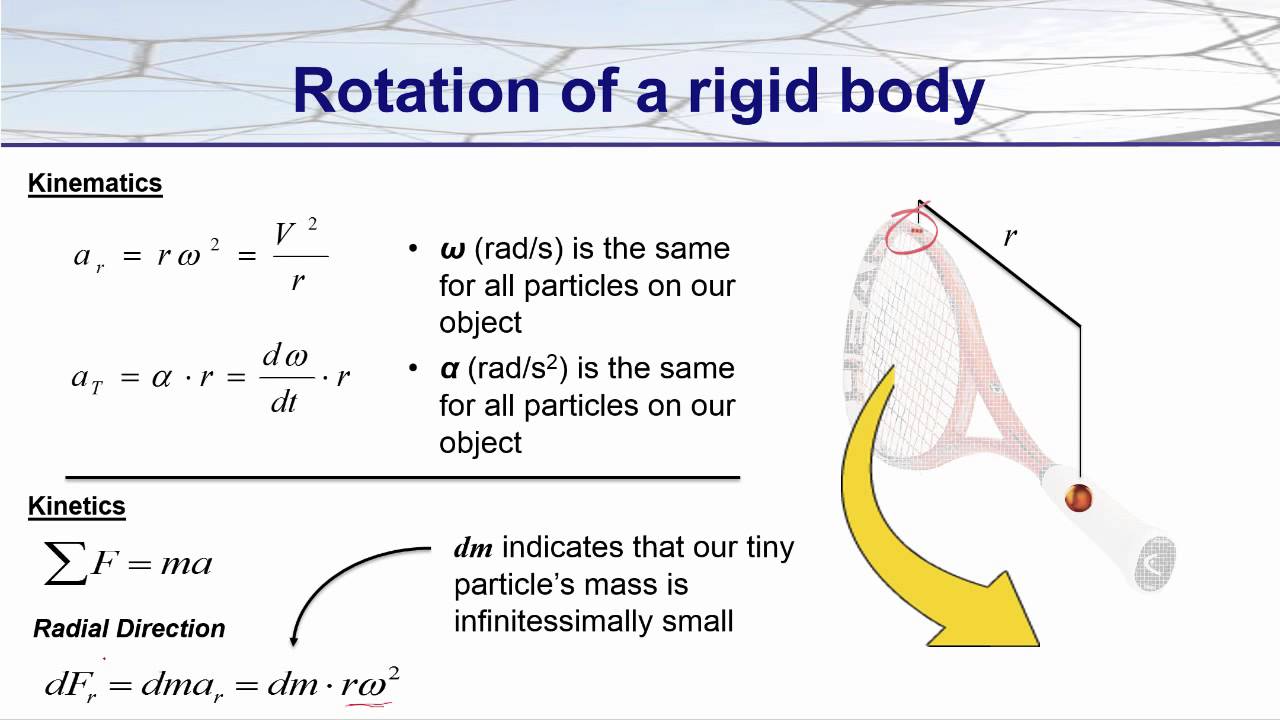
mRo Pixhawk (FMUv2)

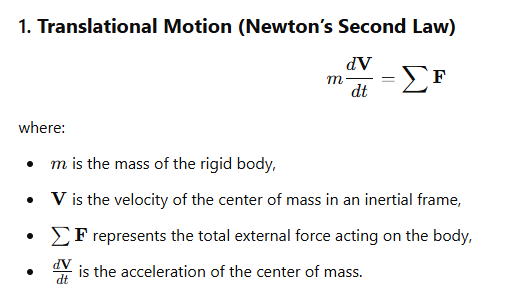
****.

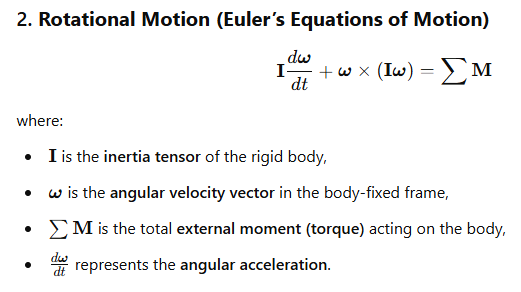
Flight Mechanics review

## **the general rigid body dynamics (RBD) equations in 3D space.**

The general rigid body dynamics (RBD) equations in 3D space describe the motion of a rigid body under the influence of external forces and moments. These equations consist of translational motion (Newton’s Second Law) and rotational motion (Euler’s Equations of Motion).



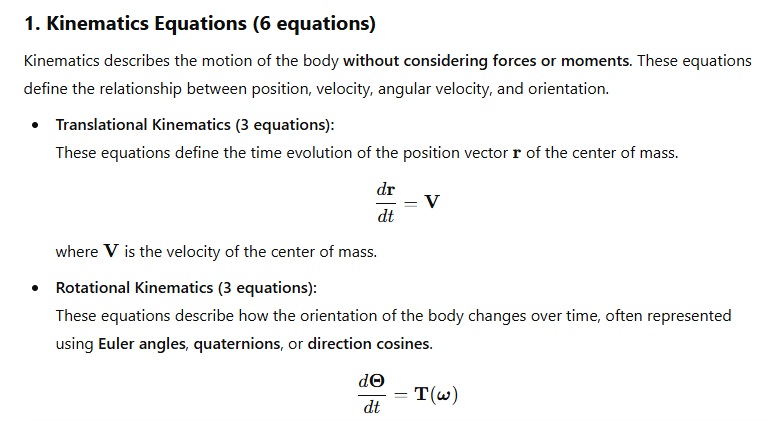


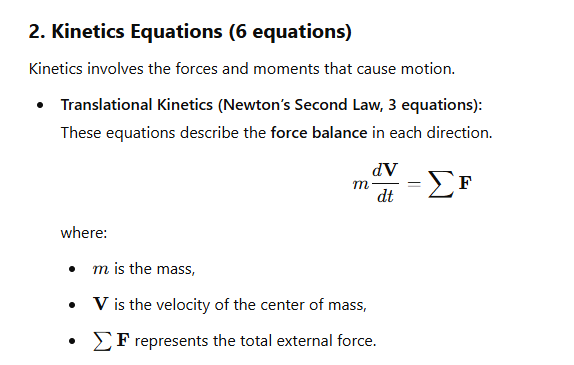


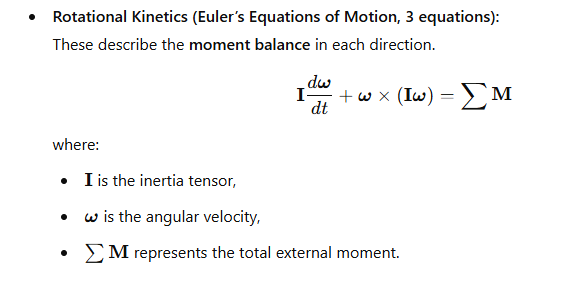
**Classification of The 12 equations of motion**

The **12 equations of motion** for a **rigid body in 3D space** consist of **kinematics** and **kinetics** equations. These are classified as follows:

|  |  |  |  |
| --- | --- | --- | --- |
| **Category** | **Variable** | **Description** | **Frame of Reference** |
| **Position (Global Location)** | x,y,z | Aircraft’s position | Earth-fixed (Inertial) |
| **Linear Velocities** | u,v,w | Forward, sideward, vertical velocity | Body-fixed |
| **Orientation (Attitude)** | ϕ,θ,ψ | Roll, pitch, yaw angles | Earth-fixed (Euler Angles) |
| **Angular Velocities** | p,q,r | Roll, pitch, yaw rates | Body-fixed |

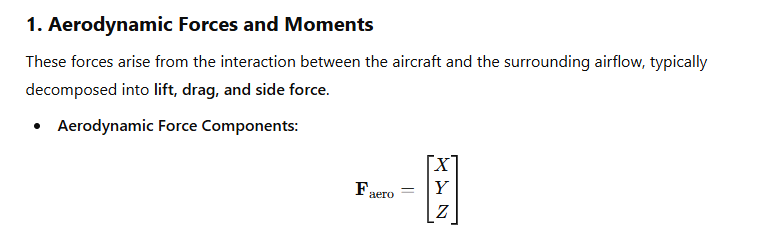
****

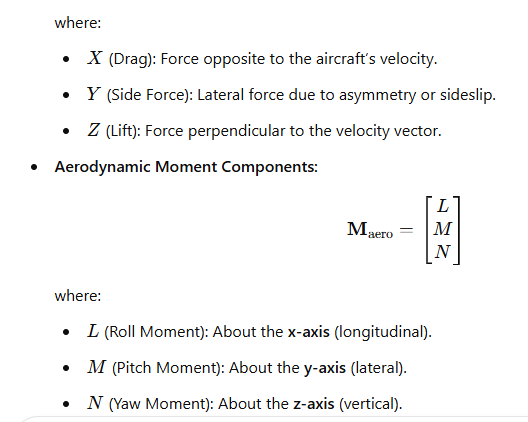
****

****

**Derivation of the Equations of Motion (EOM) for Fixed-Wing Airplanes**

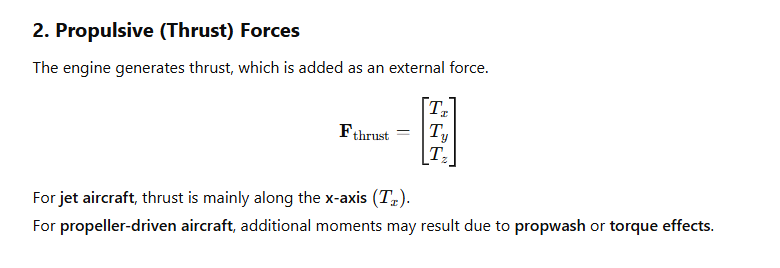
To derive the **Equations of Motion (EOM) for Fixed-Wing Airplanes**, additional **aerodynamic, propulsion, and gravitational forces** must be incorporated into the **Rigid Body Dynamics (RBD) equations**.

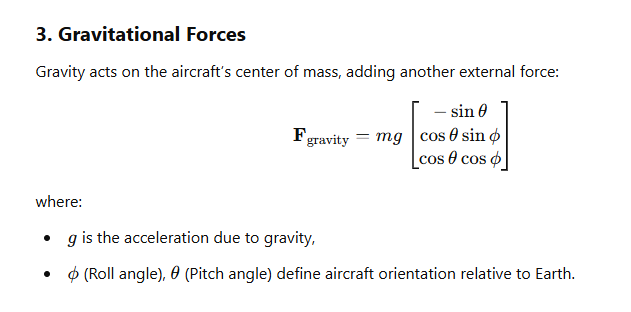
****

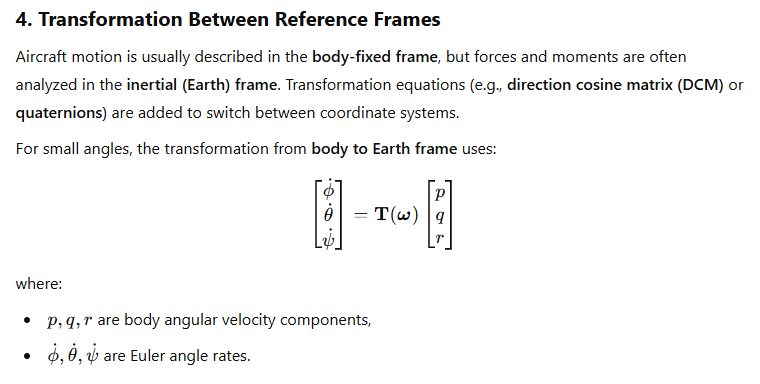
****

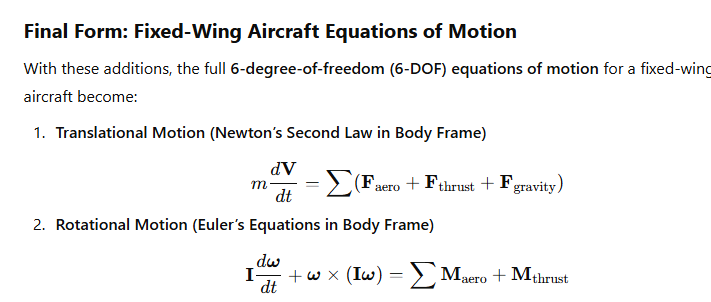
**These aerodynamic forces and moments depend on parameters such as:**

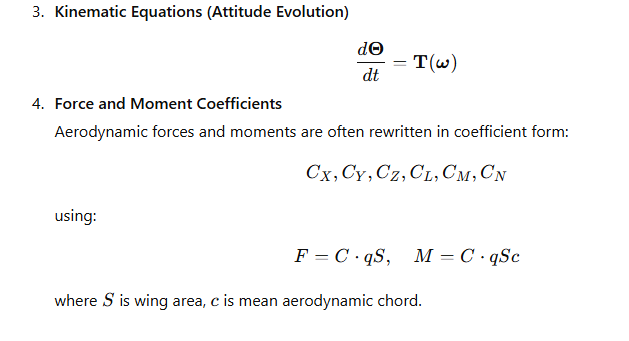
* **Angle of attack α**
* **Sideslip angle β**
* **Control surface deflections (elevator, rudder, ailerons)**
* **Dynamic pressure**



****

****

****

****

**the assumptions introduced while deriving the airplane equations of motion**

|  |  |  |
| --- | --- | --- |
| **Assumption** | **Purpose** | **When It Fails** |
| **Rigid Body** | Ignores structural deformation | Flexible aircraft, wing bending |
| **Flat Earth** | Simplifies gravity calculations | High-altitude, long-range flights |
| **Small Angles** | Linearization for easier analysis | Large maneuvers, aerobatics |
| **Symmetric Airplane** | Removes unnecessary complexity | Asymmetric damage, non-uniform loads |
| **Constant Mass & Inertia** | Avoids time-dependent equations | Fuel burn, payload changes |
| **Linear Aerodynamics** | Makes stability analysis easier | Stall, high AoA, supersonic flow |
| **Quasi-Steady Aerodynamics** | Removes wake and vortex effects | Helicopters, UAV hovering |
| **No Control Surface Dynamics** | Simplifies control modeling | High-speed jets, rate-limited actuators |
| **Standard Atmosphere** | Avoids weather effects | Gusty winds, turbulence |

## **Classification of the airplanes ( EOM ) equations mathematically**

|  |  |  |
| --- | --- | --- |
| **Category** | **Classification** | **Explanation** |
| **Order** | First-order | The EOMs describe the time rate of change of velocity and angular velocity, making them **first-order differential equations**. However, when written in terms of displacement, they become **second-order**. |
| **Type** | Ordinary Differential Equations (ODEs) | The EOMs depend on a finite number of variables (position, velocity, angles) and their derivatives concerning time, making them **ODEs** rather than PDEs (which involve spatial derivatives). |
| **Linearity** | Nonlinear | The original equations contain **quadratic terms** (e.g., products of angular velocities), making them **nonlinear**. However, they can be **linearized** for small perturbations. |
| **Coupling** | Coupled | Translational and rotational motions influence each other (e.g., roll affects yaw), making the equations **coupled**. However, they can be **decoupled** for specific flight conditions like small disturbances. |

## **the (Body axes) versus the (earth or inertial axes)**

|  |  |  |
| --- | --- | --- |
| **Feature** | **Body Axes (Aircraft Frame)** | **Earth Axes (Inertial Frame)** |
| **Fixed to** | Aircraft center of gravity | Earth's surface (assumed fixed) |
| **Orientation** | Rotates with the aircraft | Fixed relative to Earth |
| **X-Axis Direction** | Forward along fuselage | Geographic North (or another fixed direction) |
| **Y-Axis Direction** | Rightward (wing direction) | Geographic East |
| **Z-Axis Direction** | Downward (toward ground in aircraft frame) | Downward (toward Earth’s center) |
| **Used for** | Forces, moments, control inputs | Navigation, trajectory, wind effects |

## **the pitch angle (θ) versus the angle of attack (α)**

|  |  |  |
| --- | --- | --- |
| **Term** | **Pitch Angle (θ)** | **Angle of Attack )α(** |
| **Definition** | The angle between the **body X-axis** (fuselage direction) and the **Earth (inertial) horizontal**. | The angle between the **body X-axis** (fuselage direction) and the **relative wind (free stream airflow)**. |
| **Reference Frame** | Measured relative to the **Earth (inertial frame)**. | Measured relative to the **airflow (aerodynamic frame)**. |
| **Effect on Flight** | Determines the **aircraft’s nose-up or nose-down attitude** relative to the horizon. | Directly affects **lift generation** and stall behavior. |
| **Relation to Flight Path** | θ=α+γ, where γ is the **flight path angle** (angle between velocity vector and horizontal). | Affects the pressure distribution around the wing, impacting lift and drag. |
| **Example** | An aircraft flying level with a **5° nose-up pitch** has θ=5∘ | If the airflow comes at an **8° angle relative to the fuselage**, then α=8∘ |

**the sideslip angle (β) versus the heading angle (ψ)**

|  |  |  |
| --- | --- | --- |
| **Term** | **Sideslip Angle (β)** | **Heading Angle (ψ)** |
| **Definition** | The angle between the **body Y-axis** (lateral axis) and the **oncoming airflow (relative wind)**. | The angle between the **Earth X-axis** (North) and the aircraft’s **velocity vector in the horizontal plane**. |
| **Reference Frame** | Measured in the **body frame**. | Measured in the **Earth (inertial) frame**. |
| **Effect on Flight** | Indicates **yaw misalignment** between the aircraft’s motion and fuselage direction (important for stability). | Describes the **direction of travel** relative to geographic North. |
| **Relation to Yaw** | A nonzero βmeans the aircraft is experiencing **aerodynamic yawing forces** (e.g., from wind or asymmetric thrust). | ψ is affected by **wind, yaw maneuvers, and navigation corrections**. |
| **Example** | If an aircraft is moving forward but slightly **crabbed to the left** due to wind, it has β>0∘ | If an aircraft is flying **eastward**, its heading is ψ=90∘ |

|  |  |  |  |
| --- | --- | --- | --- |
| **Term** | **Frame of Reference** | **Reference Direction** | **Indicates** |
| **Pitch Angle (θ)** | Earth (inertial frame) | Earth’s horizontal | Nose-up or nose-down attitude |
| **Angle of Attack (α)** | Aerodynamic frame | Relative airflow | How air meets the wing, affecting lift |
| **Sideslip Angle (β)** | Body frame | Relative wind | Lateral motion (slipping/skidding) |
| **Heading Angle (ψ)** | Earth (inertial frame) | Geographic North | Direction of motion |

## **attitude representations**

When representing an aircraft’s orientation in space, we can use different mathematical approaches:

* **Euler Angles (Roll ϕ, Pitch θ, Yaw ψ)**
* **Direction Cosine Matrix (DCM)**
* **Quaternions (q0,q1,q2,q3q\_0, q\_1, q\_2, q\_3q0​,q1​,q2​,q3​)**
* **Axis-Angle Representation (r,θ)**

### **1. Euler Angles (ϕ,θ,ψ)**

Euler angles define orientation using **three sequential rotations** about different axes.

**Advantages:**

* **Intuitive & easy to interpret** (directly relate to aircraft roll, pitch, and yaw).
* Common in flight dynamics, autopilots, and human interfaces.
* Requires only **three parameters**, making it **compact**.

**Disadvantages:**

* **Gimbal Lock:** Loss of one degree of freedom when pitch (θ) is ±90°.
* Complex trigonometric functions required for conversions.
* Singularities in transformations make certain maneuvers problematic.

*Best for:***Small-angle approximations, human-readable attitude representation.**

### **2. Direction Cosine Matrix (DCM)**

The **DCM** (or **rotation matrix**) is a **3×3 matrix** that transforms vectors between reference frames.

**Advantages:**

* **No singularities** (avoids gimbal lock).
* Directly used in **force and moment transformations**.
* Can represent **any rotation uniquely**.

**Disadvantages:**

* Requires **9 elements** (redundant since only 3 are independent).
* Computationally expensive **(matrix multiplications required)**.
* Requires **orthonormalization** to avoid numerical drift over time.

*Best for:***Precise numerical calculations in navigation & control systems.**

### **3. Quaternions (q0​,q1​,q2​,q3​)**

A quaternion is a **4D complex number** that represents rotations without gimbal lock.

**Advantages:**

* **No gimbal lock or singularities**.
* **More computationally efficient** than DCM for rotations.
* **Compact (4 elements)** while avoiding redundancy.
* **Stable in iterative computations** (used in simulations and spacecraft).

**Disadvantages:**

* Less intuitive than Euler angles.
* Requires **normalization** to prevent drift over time.
* More complex mathematical operations (e.g., quaternion multiplication).

*Best for:***Flight simulations, spacecraft, and real-time attitude control.**

### **4. Axis-Angle Representation (r,θ)**

Defines rotation by an **axis of rotation** r=(rx,ry,rz)and a **rotation angle**θ

**Advantages:**

* **Minimal representation** for single-axis rotations.
* Directly used in **mechanical systems and robotics**.

**Disadvantages:**

* **Not ideal for continuous rotations** (e.g., flight dynamics).
* Requires **conversion** for matrix-based calculations.

*Best for:***Describing single-axis rotations or manual transformations.**

**General comparison**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Representation** | **Parameters** | **Gimbal Lock?** | **Computational Cost** | **Intuitive?** | **Best Use Case** |
| **Euler Angles** | 3 | Yes | High (trigonometry) | Yes | Aircraft control, displays |
| **DCM (Rotation Matrix)** | 9 | No | Moderate (matrix math) | No | Precise transformations |
| **DCM (Rotation Matrix)** | 4 | No | Low (efficient math) | No | Flight sims, spacecraft |
| **Axis-Angle** | 4 | No | High (for multiple transformations) | Yes | Single-axis rotations |

Numerical solution of ODEs

**the numerical solving algorithms for ODEs**

**1. Euler's Method:** This is one of the simplest numerical methods for solving ODEs. It is an explicit method which means the solution can be advanced step by step, using the value at the current step to estimate the value at the next step.

**2. Heun's Method:** Also known as the improved Euler method, this technique is a predictor-corrector method that improves upon the basic Euler method by using an average of the slopes at the beginning and the end of the interval to estimate the next value.

**3. Midpoint Method:** This method is a second-order method which uses the slope at the midpoint of the interval to estimate the next value, providing better approximation than Euler's method.

**4. Runge-Kutta Methods:** These are a family of iterative methods which include the well-known fourth-order Runge-Kutta method (RK4). RK4 is widely used due to its balance between computational effort and accuracy.

**5. Adams-Bashforth Methods:** These are multi-step methods that use the values of the solution at several previous steps to compute the next value. They are explicit methods and are often used in conjunction with other methods to start the computation.

**6. Adams-Moulton Methods:** These are implicit multi-step methods that use the value at the current step as well as previous steps to estimate the next value. They are generally more accurate and stable than explicit methods but require solving an equation at each step.

**7. Backward Differentiation Formulas (BDF):** These are implicit multi-step methods that are particularly useful for solving stiff equations. They use past values of the solution and its derivative to estimate the next value.

**8. Verlet Integration:** Commonly used in molecular dynamics simulations, this method is particularly good for solving Newton's equations of motion and is known for its stability and energy conservation properties.

**9. Leapfrog Integration:** This is a simple method that is particularly useful for problems where the velocity and position are updated in an alternating fashion, often used in the context of Hamiltonian mechanics.

**10. Symplectic Integrators:** These are designed for solving Hamiltonian systems and are particularly good at conserving the symplectic structure of the system, which is important for long-term simulations.

## **Choosing an algorithm for solving the Airplanes EOM,**

for solving the Airplanes EOM we usually use the fourth-order Runge-Kutta method (RK4).

**Initial conditions needed** : At t = 0 we need the unknowns ( u, v , w , p , q , r , φ , θ , ψ , x , y , z)

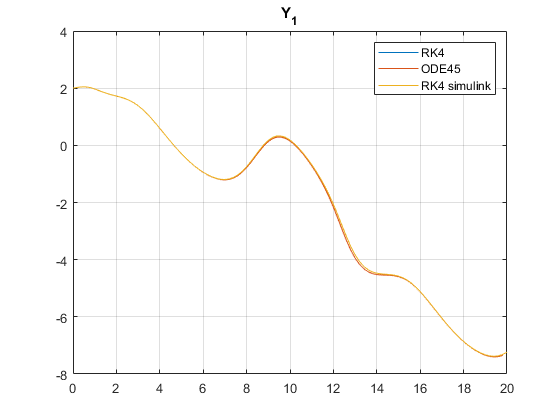
**Inputs needed in each iteration** , F\_y , F\_z , L , M , N

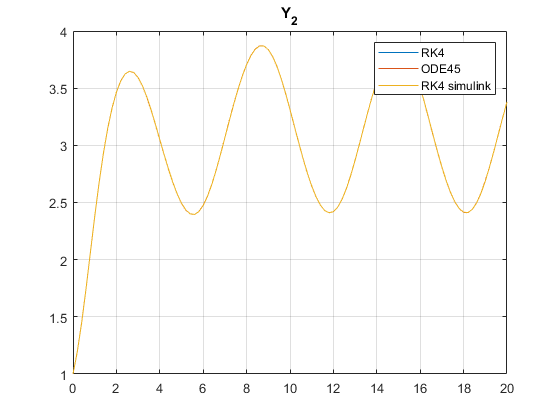
I & m are constants

**Outputs calculated in each iteration**: u, v , w , p , q , r , φ , θ , ψ , x , y , z

## **Solving the system**

### **MATLABresults**





[*Published with MATLAB® R2023b*](https://www.mathworks.com/products/matlab)

# **As shwon above, the RK4 function RK4 simulink produced the same results as the MATLAB function “ODE45”**

* RBD equations solution
  1. Problem definition

After exploring various numerical methods for solving ODEs, this task requires implementing the 4th-order Runge-Kutta (RK4) algorithm to solve the 12 rigid body dynamics (RBD) equations. The solution will be computed using fixed input forces and moments, along with given initial conditions for the 12 state variables.

* 1. Givens
  2. Initial conditions (𝑆.𝐼. 𝑢𝑛𝑖𝑡𝑠)
  3. Mathematical modeling
* As mentioned in the first task, the rigid body dynamics (RBD) equations consist of 12 coupled, nonlinear, first-order differential equations (ODEs). These equations can be categorized into six kinetic equations describing forces and moments and six kinematic equations governing motion.

**Kinetics equations**:

(4. 1)

(4.2)

**Kinematics equations**:

(4.3)

(4.4)

* 1. Problem solution
* We will utilize the vector form of the rigid body dynamics (RBD) equations to compute the 12 state variables after 25 seconds, using the given parameters and initial conditions. To derive the required 12-state formulation from the RBD equations, mathematical manipulation and rearrangement will be performed to systematically organize the kinetic and kinematic equations.

From kinetics forces equations:

(4.5)

From kinetics forces equations:

(4.5)

From kinetics moments equations:

(4.6)

From kinematics equations:

(4.7)

(4.8)

* 1. **Runge – Kutta method – 4th Order**

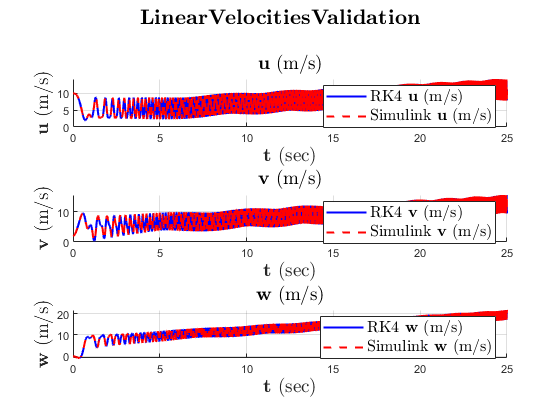
This method determines the new state at each time step based on the previous state xnx\_n using the following formula:

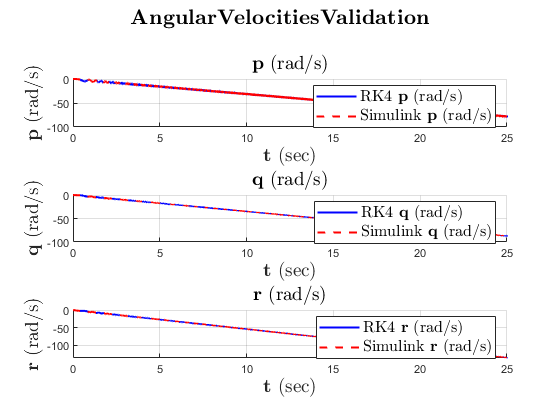
(4.9)

Where;

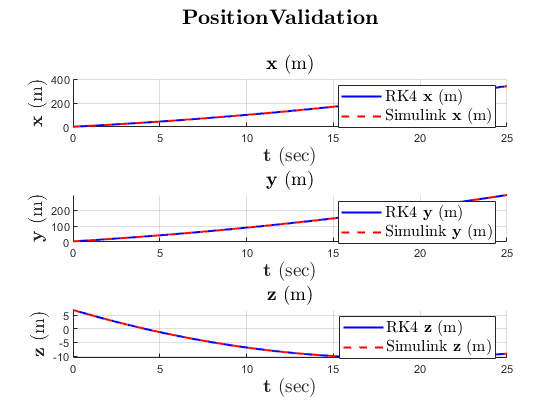
This set of equations is also applicable to multi-dimensional variables in vector form. By implementing a MATLAB code and starting from the given initial conditions, each of the 12 state variables will be computed iteratively using the Runge-Kutta method, ultimately reaching their final values by the end of the analysis period.

# **Figures**





# 

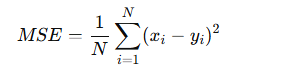


Since MATLAB code and Simulink are solving the same equations with the same solver, we got identical graphs.

**Bonus:**

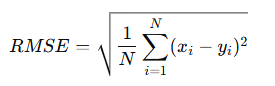
**To compare two signals (vectors) and quantify the error between them, the following mathematical expressions can be used:**

**1. Mean Squared Error (MSE)**

****

* Measures the average squared difference between corresponding elements of two vectors.
* Sensitive to large errors due to squaring.

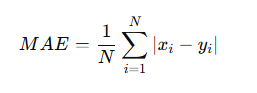
**2. Root Mean Squared Error (RMSE)**

****

 Provides an error measurement in the same unit as the original signals.

 Useful for understanding deviations in real-world terms.

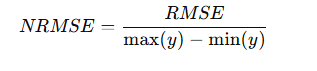
**3. Mean Absolute Error (MAE)**

****

 Measures the average absolute differences.

 Less sensitive to outliers compared to MSE.

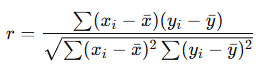
**4. Normalized Root Mean Squared Error (NRMSE)**

****

 Normalizes RMSE using the range of the reference signal.

 Useful for comparing errors across different datasets.

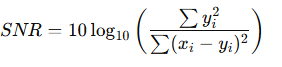
**5. Pearson Correlation Coefficient (r)**

****

 Measures the linear relationship between two signals.

 Values close to 1 indicate strong correlation.

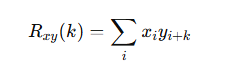
**6. Signal-to-Noise Ratio (SNR)**

****

 Evaluates how much the signal is corrupted by error.

 Higher values indicate better similarity.

**7. Cross-Correlation**

****

* Measures similarity as a function of signal displacement.
* Useful for time-shifted comparisons.

**Appendices  
Appendix A**

### **MATLAB code**

clc;  
close all;  
clearvars;

Time Vector Initialization

t0 = 0; % Initial time (s)  
tf = 25; % Final time (s)  
dt = 0.001; % Time step (s)  
t\_vec = t0:dt:tf; % Time vector

External Forces and Moments

Forces = [2; 8; 3]; % Forces (N)  
Moments = [14; 20; 7]; % Moments (N.m)

Aircraft Mass and Inertia

g = 9.81; % Gravity (m/s^2)  
m = 11; % Mass of the aircraft (kg)  
  
% Inertia Matrix (kg.m^2)  
I\_mat = [1 -2 -1;  
 -2 5 -4;  
 -1 -4 0.2];

Initial Conditions

Initial state vector: [u, v, w, p, q, r, φ, θ, ψ, x, y, z]

states\_vec(:,1) = [10, 2, 0, 2\*pi/180, pi/180, 0, 20\*pi/180, 15\*pi/180, 30\*pi/180, 2, 4, 7];  
  
% Assign initial values  
[u0,v0,w0,p0,q0,r0,phi0,theta0,psi0,x0,y\_0,z0] = deal(10,2,0,2\*pi/180,pi/180,0,20\*pi/180,15\*pi/180,30\*pi/180,2,4,7);

Runge-Kutta 4th Order (RK4)

[t\_vec\_RK4, states\_vec\_RK4] = raunge\_kutta\_4(t\_vec, states\_vec(:,1), Forces, Moments, m, I\_mat);

Simulink RK4

simOut = sim("RBD\_simulink\_model.slx");

Convert Euler Angles to Degrees

Convert RK4 results

states\_vec\_RK4(7:9, :) = rad2deg(states\_vec\_RK4(7:9, :));  
  
% Convert Simulink results  
simOut.phi\_simulink.data = rad2deg(simOut.phi\_simulink.data);  
simOut.theta\_simulink.data = rad2deg(simOut.theta\_simulink.data);  
simOut.psi\_simulink.data = rad2deg(simOut.psi\_simulink.data);

Plot Results

plotValidation(t\_vec\_RK4, states\_vec\_RK4, simOut);

**Appendix B**

**Raunge kutta4th order function**

function [t\_vec, states\_vec] = raunge\_kutta\_4(t\_vec, y0, Forces, Moments, m, I\_mat)

n = length(t\_vec);

dt = t\_vec(2) - t\_vec(1);

states\_vec = zeros(12,n);

states\_vec(:,1) =y0;

for i =1:n-1

K1 = get\_states\_dot(t\_vec(i),states\_vec(:,i), Forces, Moments, m, I\_mat);

K2 = get\_states\_dot(t\_vec(i)+.5\*dt,states\_vec(:,i)+K1\*0.5\*dt, Forces, Moments, m, I\_mat);

K3 = get\_states\_dot(t\_vec(i)+.5\*dt,states\_vec(:,i)+K2\*0.5\*dt, Forces, Moments, m, I\_mat);

K4 = get\_states\_dot(t\_vec(i)+dt,states\_vec(:,i)+K3\*dt, Forces, Moments, m, I\_mat);

states\_vec(:,i+1) = states\_vec(:,i) + dt/6 \* (K1+2.\*K2+2.\*K3+K4);

end

end

**get\_states\_dot Function**

function states\_dot = get\_states\_dot(~, states\_vec, Forces, Moments, m, I\_mat)

% get\_states\_dot Computes the time derivatives of the state variables

%

% Inputs:

% - states\_vec: State vector [u, v, w, p, q, r, phi, theta, psi, x, y, z] (12x1)

% - Forces: External forces in body frame [Fx, Fy, Fz] (3x1)

% - Moments: External moments in body frame [Mx, My, Mz] (3x1)

% - m: Mass of the vehicle (scalar)

% - I\_mat: Inertia matrix (3x3)

%

% Outputs:

% - states\_dot: Time derivative of states (12x1)

% Extract states

u = states\_vec(1);

v = states\_vec(2);

w = states\_vec(3);

p = states\_vec(4);

q = states\_vec(5);

r = states\_vec(6);

phi = states\_vec(7);

theta = states\_vec(8);

psi = states\_vec(9);

x = states\_vec(10);

y = states\_vec(11);

z = states\_vec(12);

% Rotation matrix for Euler angles (ZYX convention)

J = [ 1, sin(phi)\*tan(theta), cos(phi)\*tan(theta);

0, cos(phi), -sin(phi);

0, sin(phi)/cos(theta), cos(phi)/cos(theta)];

% Compute translational dynamics

vel\_dot = (1/m) \* Forces - cross([p; q; r], [u; v; w]);

% Compute rotational dynamics

omega\_dot = I\_mat \ (Moments - cross([p; q; r], I\_mat \* [p; q; r]));

% Compute Euler angle rates

euler\_dot = J \* [p; q; r];

% Compute position rates in the inertial frame

pos\_dot = eul2rotm([psi, theta, phi], 'ZYX') \* [u; v; w];

% Concatenate state derivatives

states\_dot = [vel\_dot; omega\_dot; euler\_dot; pos\_dot];

end

**plotting Function**

function plotValidation(t\_vec\_RK4, states\_vec\_RK4, simOut)

% Function to compare simulation results from RK4 and Simulink

figure\_titles = {'Linear Velocities Validation', ...

'Angular Velocities Validation', ...

'Euler Angles Validation', ...

'Position Validation'};

labels = {'$\bf{u}$ (m/s)', '$\bf{v}$ (m/s)', '$\bf{w}$ (m/s)', ...

'$\bf{p}$ (rad/s)', '$\bf{q}$ (rad/s)', '$\bf{r}$ (rad/s)', ...

'$\bf{\phi}$ ($^\circ$)', '$\bf{\theta}$ ($^\circ$)', '$\bf{\psi}$ ($^\circ$)', ...

'$\bf{x}$ (m)', '$\bf{y}$ (m)', '$\bf{z}$ (m)'};

simData = {simOut.u\_simulink, simOut.v\_simulink, simOut.w\_simulink, ...

simOut.p\_simulink, simOut.q\_simulink, simOut.r\_simulink, ...

simOut.phi\_simulink, simOut.theta\_simulink, simOut.psi\_simulink, ...

simOut.x\_simulink, simOut.y\_simulink, simOut.z\_simulink};

for j = 1:4 % Loop for 4 figures

figure;

sgtitle(['$\bf{' figure\_titles{j} '}$'], 'Interpreter', 'latex', 'FontSize', 16);

for i = 1:3

idx = (j-1)\*3 + i;

subplot(3,1,i); hold on;

plot(t\_vec\_RK4, states\_vec\_RK4(idx, :), 'b', 'LineWidth', 1.5);

plot(simData{idx}.time, simData{idx}.data, '--r', 'LineWidth', 1.5);

title(labels{idx}, 'Interpreter', 'latex', 'FontSize', 14);

xlabel('$\bf{t}$ (sec)', 'Interpreter', 'latex', 'FontSize', 14);

ylabel(labels{idx}, 'Interpreter', 'latex', 'FontSize', 14);

legend(['RK4 ' labels{idx}], ['Simulink ' labels{idx}], ...

'Interpreter', 'latex', 'FontSize', 12);

grid on;

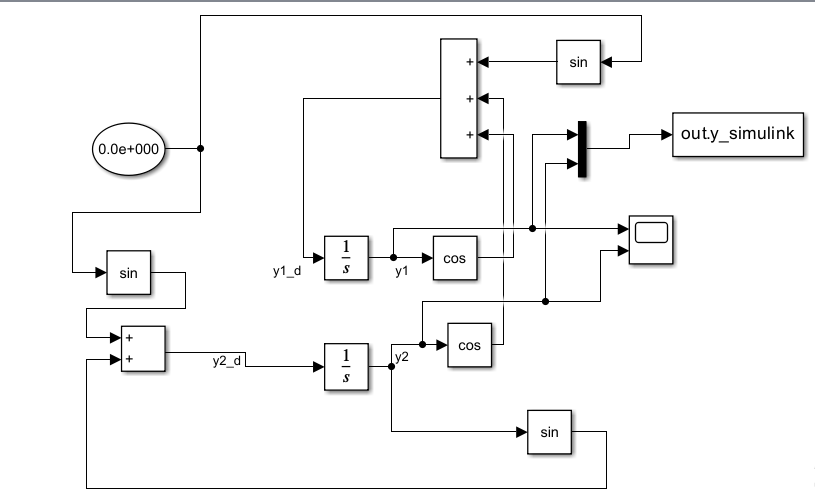
end

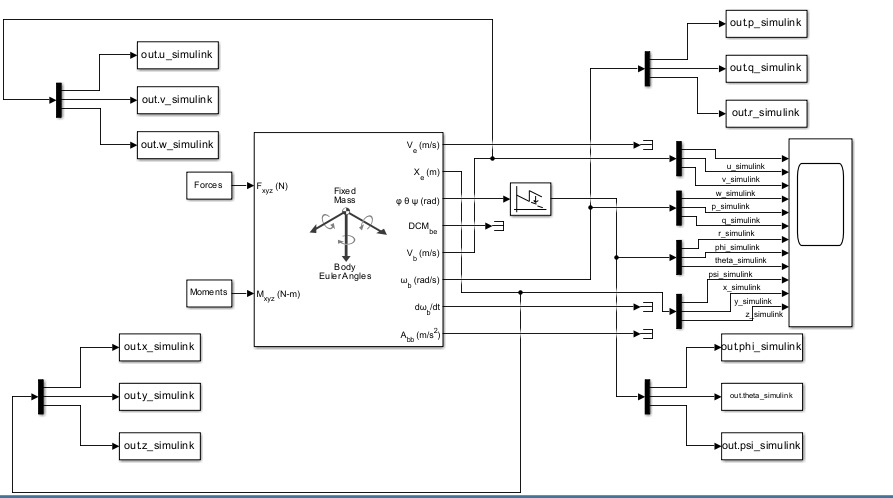
end

end

**Appendix C**

**Simulink model**



**

# **References :**

** Stevens, B. L., & Lewis, F. L. (2003).***Aircraft Control and Simulation: Dynamics, Controls Design, and Autonomous Systems* (3rd ed.). Wiley.

Covers equations of motion, stability, and attitude representations in-depth.

** Etkin, B., & Reid, L. D. (1996).***Dynamics of Flight: Stability and Control* (3rd ed.). Wiley.

Discusses rigid body dynamics, stability, and control of fixed-wing aircraft.

** Zipfel, P. H. (2007).***Modeling and Simulation of Aerospace Vehicle Dynamics* (2nd ed.). AIAA.

Provides an in-depth discussion of quaternions, DCM, and attitude control.

**NASA Technical Reports:** [**https://ntrs.nasa.gov/**](https://ntrs.nasa.gov/)

**Free lectures and notes on aircraft stability, control, and motion equations Previous years**

**Aircraft Control and Simulation" by Brian L. Stevens & Frank L. Lewis – Covers sensors used in autopilots and flight control.**

****[**https://worldaviationato.com/en/aircraft-autopilot/**](https://worldaviationato.com/en/aircraft-autopilot/)

****[**https://www.airbus.com/en/node/51851**](https://www.airbus.com/en/node/51851)

****[**https://www.unmannedsystemstechnology.com/expo/uav-autopilot-systems/**](https://www.unmannedsystemstechnology.com/expo/uav-autopilot-systems/)

****[**https://controltheorymaster.files.wordpress.com/2017/11/farid-golnaraghi-benjamin-c-**](https://controltheorymaster.files.wordpress.com/2017/11/farid-golnaraghi-benjamin-c-)**kuo-automatic-control-systems.pd**

 Bishop, C. M. (2006). *Pattern Recognition and Machine Learning*. Springer.

 Kay, S. M. (1993). *Fundamentals of Statistical Signal Processing*. Prentice Hall.

 Ljung, L. (1999). *System Identification: Theory for the User*. Prentice Hall.