ENME 816 Final Presentation

Alex Dorsey

Mechanical Engineering Department
University of Maryland Baltimore County

December, 2025

Project Objective

- Build an Interactive 3D quad-copter Simulator:
  - Non Linear Rigid Body Dynamics
  - Demonstrate Real Time Roll and Pitch Angle Estimation from Simulated Noisy IMU Data
  - Adaptive Or Fixed Gain Inner-loop control
  - Test bed for visualizing control systems effects on dynamic behavior
  - FFT of estimated attitude to adjust low pass filter cut-off frequency

Project Motivation

- Aligns with the Estimation and Control Learning Laboratory’s work at UMBC
- Visualize quad copter and the effects the control system and various parameters have on the behavior
- Make a ‘video game’ like application in Matlab

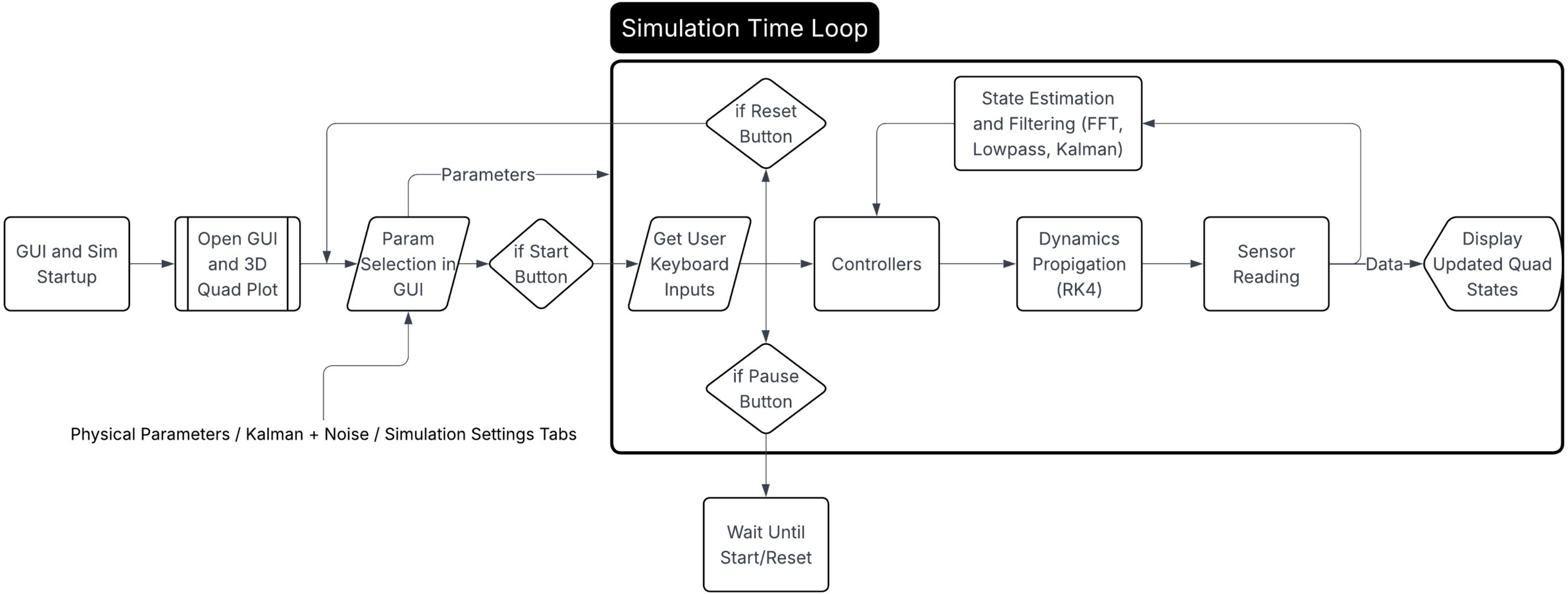
Data Source: Quadcopter Dynamics

- $m\ddot{r} = mg + f\mathcal{O}_{\mathrm{A/D}}(q)e_3$
- $\dot{q} = S(q)^{-1}\omega$
- $J\dot{\omega} + \omega \times J\omega = \tau$

Data Source

- State Vector (12x1)
  - Attitudes, body angular rates, position, and velocities (east, north, up).
  - Determined Via Propagation of ODE function
- Inputs and Disturbances
  - Keyboard User Commands
  - Process and measurement noise simulating the gyroscope and accelerator:
    - Accelerator Data = Noise*Acceleration True + Bias
    - Gyro Data = Noise*Body Rates True + Bias
    - Noise is a user definable parameter in terms of frequency and magnitude.

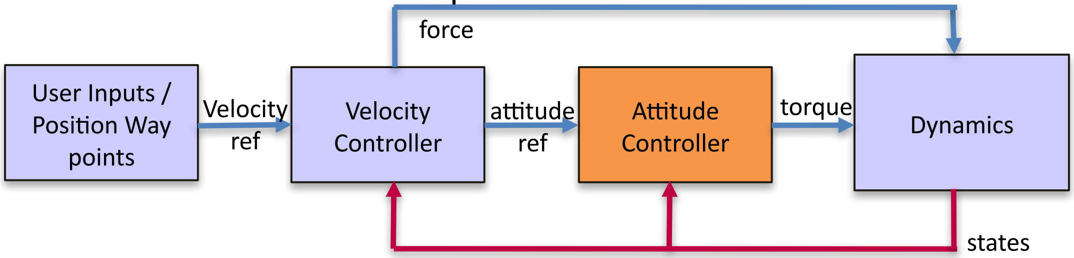
Algorithm: Flow Chart

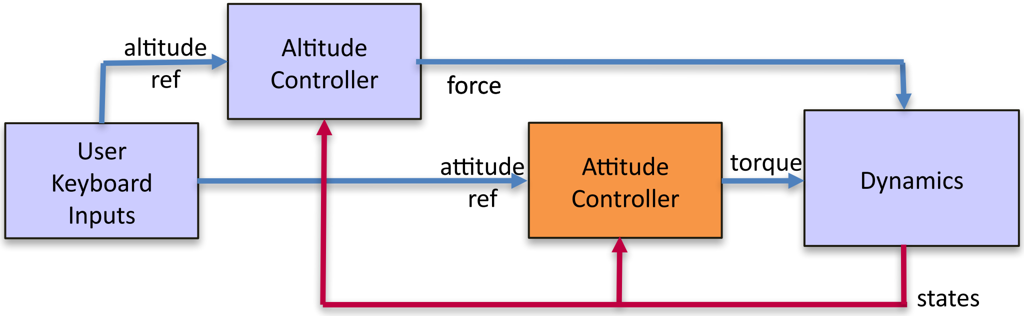


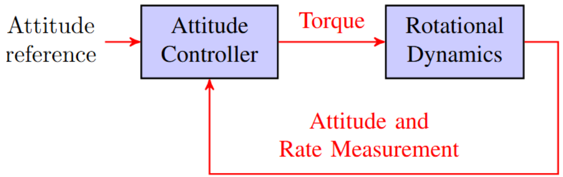
* The parameters are checked for updates each step and startup

Algorithm: Multirotor Control Flow

- Two Control Modes: Velocity Control and Attitude Control
- Velocity Control: User Commands velocity forward, velocity sideways, and altitude
- Angle Control: User commands roll, pitch, yaw, and altitude
- Way point function replaces User Keyboard Inputs and outputs Velocity Ref
- Internal Model Principle used to make controllers





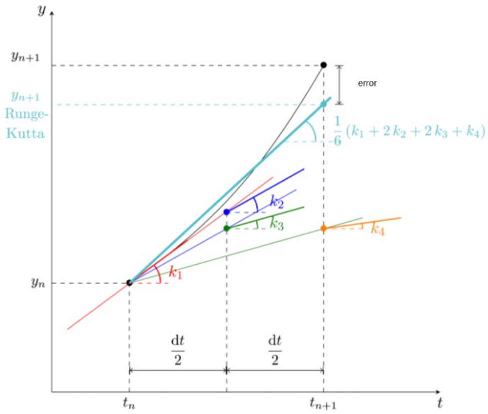


UMBC Algorithm: Multirotor Control Flow

- Velocity Controller:
  - PI controller (states are 3x1 vectors, gains are 3x3 diagonal matrices):
    - Force Vector = P [Velocity Error] + I [Velocity Error Integral]
    - Force = norm(Force Vector)
    - Force Vector is used to calculated attitude commands because the motors’ shafts must always be co-linear with the Force Vector.
- Altitude Controller
  - Force = K [Velocity Up Error] + mg
    - Attitude comes from user command -> no need for a force Vector.
- Way Points:
  - Cruising Velocity = P * Position Error
  - If norm(Position Error) < user defined distance
    - Velocity Ref = Cruising Velocity * Position Error/norm(Position Error)
  - Else
    - Velocity Ref = Cruising Velocity

Course Related Concepts

- ODE
  - Non-Linear ODE
  - ODE is propagated via Runge Kutta 4. A fourth order numerical integration technique.
- Signal Processing:
  - Kalman Filtering → estimated attitude from Noisy Data
  - Lowpass Filtering → Filter Noisy Kalman Estimate
  - FFT → Observe Frequency of Filtered Noisy Data to update Lowpass Filter
  - Many Feedback loops from states and estimated states to update next control signal (Signal Processing at its core)
- Rigid Body Visualization and GUI
  - 3D quadcopter model with rotation and translation for body and propellers
  - Dynamic Camera Following with smoothed motion (first order filter)
  - Projected Shadow on the ground plane
  - Additional Plots:
    - attitude history showing true vs estimated
    - Top Down Position History
    - FFT of estimated Angle



Varying Parameters

Demonstration

Future Work:

- Extend State Estimation to Include the full state (currently only attitude)
- Introduce wind disturbances
- Higher Fidelity Quad Copter
- Implement more adapter controllers
- Improve Visualization
- Speed up performance (needs a good CPU to run fast)