ETF Documentation – Controls Task

Nageswar Lakshmikanth
21AE30013
9347488450

lakshmikanth.nageswar@kgpian.iitkgp.ac.in

Dynamics of Quadrotor:

For a quadrotor Control inputs are angular speeds of 4 motors. Vertical thrust provided by each motor is,

U_i = b*w^2

Where b is a constant and w is angular velocity of the motor.

The four control inputs of the quadrotor are simplified as U1, U2, U3, U4. (Known as effective inputs)

$$\left.\begin{aligned} U_1 &= (Th_1 + Th_2 + Th_3 + Th_4)/m \\ U_2 &= l(-Th_1 - Th_2 + Th_3 + Th_4)/I_1 \\ U_3 &= l(-Th_1 + Th_2 + Th_3 - Th_4)/I_2 \\ U_4 &= C(Th_1 + Th_2 + Th_3 + Th_4)/I_3 \end{aligned}\right\}$$

Where:

u_1 : Vertical thrust generated by the four rotors
u_2 : Pitching moment
u_3 : Yawing moment
u_4 : Rolling moment
Th_i : The thrusts generated by four rotors

References used –

"Modelling and PID controller design for a quadrotor unmanned air vehicle"- https://ieeexplore.ieee.org/document/5520914

For the Mathematical modelling or Dynamics modelling of system-

There is transformation of axes from an inertial frame fixed to ground to the frame attached to the centre of gravity of the quadrotor. So the State variables are multiplied with a rotation matrix in order to obtain the parameters in the ground frame.

The rotation matrix for dynamics of quadrotor is given by,

$$R_{zxy} = \begin{bmatrix} C_\varphi C_\theta & C_\varphi S_\theta S_\psi - S_\varphi C_\psi & C_\varphi S_\theta C_\psi + S_\varphi S_\psi \\ C_\varphi S_\theta & S_\varphi S_\theta S_\psi + C_\varphi C_\psi & S_\varphi S_\theta C_\psi - C_\varphi S_\psi \\ -S_\theta & C_\theta S_\psi & C_\theta C_\psi \end{bmatrix}$$

The six equations of dynamics of Quadrotor are obtained as,

$$\left.\begin{aligned} \ddot{x} &= u_1 (Cos\phi Sin\theta Cos\psi + Sin\phi Sin\psi) - K_1 \dot{x}/m \\ \ddot{y} &= u_1 (Sin\phi Sin\psi Cos\psi - Cos\phi Sin\psi) - K_2 \dot{y}/m \\ \ddot{z} &= u_1 (Cos\phi Cos\psi) - g - K_3 \dot{z}/m \end{aligned}\right\}$$

$$\left.\begin{aligned} \ddot{\theta} &= u_2 - lK_4 \dot{\theta}/I_1 \\ \ddot{\psi} &= u_3 - lK_5 \dot{\psi}/I_2 \\ \ddot{\varphi} &= u_4 - K_6 \dot{\varphi}/I_3 \end{aligned}\right\}$$

The Aerodynamic drag in generic case is given by,

D = kv^n (where k is a constant, v – variable)

Considered linearity i.e., n=1 we get the Aerodynamic drag as,

D = k*v

Reference used for dynamic modelling –

Considering the Aerodynamic drag (used in implementation):

"Modelling and PID controller design for a quadrotor unmanned air vehicle"- https://ieeexplore.ieee.org/document/5520914

For understanding basics of Reference frames and dynamic modelling (did not use for implementation since it does not constitute Aerodynamic Factors)–

https://scholarcommons.sc.edu/etd/4069/

This quadrotor helicopter model has six outputs (x, y, z, θ, ψ, φ) while it only has four independent inputs, therefore the quadrotor is an **under-actuated** system. We are not able to control all of the states at the same time**. A possible combination of controlled outputs can be x, y, z and φ in order to track the desired positions, move to an arbitrary heading and stabilize the other two angles, which introduces stable zero dynamics into the system. (From Reference used)**

Some reference Papers considered that the rotational state space i.e., (θ, ψ, φ) and translational variable altitude (Z) can be determined, considering Y, X known.

The numerical approximation for the system can be done for the state space by solving the system simultaneously.

Method Used for numerical solution - Runge Kutta of 4th order

Here the equations are of second order and each state variable depends on the other state variables as well as the Control inputs.

So Runge Kutta method can be used for solving the system by reducing the system of equations into a new system of linear equations.

Runge kutta method can handle systems where each parameter is dependent on many parameters. Hence, this method is opted.

While solving equations (through python), at each iteration the thrust is assumed to be constant (within interval of a time step dt). This is assumed as the behaviour of control input is unknown while solving the dynamics.(user input) reducing equations to system of equations of order 1

$$\frac{d\bar{x}}{dt} = u_1(\cos\phi \sin\theta \cos\psi + \sin\phi \sin\psi) - \frac{k_1 \bar{x}}{m}$$

$$\frac{d\bar{y}}{dt} = u_1(\sin\phi \sin\psi \cos\phi - \cos\phi \sin\psi) - \frac{k_2 \bar{y}}{m}$$

$$\frac{d\bar{z}}{dt} = u_1(\cos\phi \cos\psi) - g - \frac{k_3 \bar{z}}{m}$$

$$\frac{d\bar{\theta}}{dt} = u_2 - \frac{l k_4 \bar{\theta}}{I_1}$$

$$\frac{d\bar{\psi}}{dt} = u_3 - \frac{l k_5 \bar{\psi}}{I_2}$$

$$\frac{d\bar{\phi}}{dt} = u_4 - \frac{k_6 \bar{\phi}}{I_3}$$

$$\bar{x} = \frac{dx}{dt}, \quad \bar{y} = \frac{dy}{dt}, \quad \bar{z} = \frac{dz}{dt}$$

$$\bar{\theta} = \frac{d\theta}{dt}, \quad \bar{\psi} = \frac{d\psi}{dt}, \quad \bar{\phi} = \frac{d\phi}{dt}$$

$$\frac{d\bar{x}}{dt} = f_1(\phi, \theta, \psi, \bar{x})$$

$$\frac{d\bar{y}}{dt} = f_2(\phi, \psi, \bar{y})$$

$$\frac{d\bar{z}}{dt} = f_3(\phi, \psi, \bar{z})$$

$$\frac{d\bar{\psi}}{dt} = f(\bar{\psi})$$

$$\frac{d\bar{\phi}}{dt} = f(\bar{\phi})$$

$$\frac{d\bar{\theta}}{dt} = f(\bar{\theta})$$

At a particular iterate u_i is constant.

Doing computations for all state variables parallely,

Pseudo code for dynamics solver-

Defined functions for the system of linear equations obtained by reducing the 6 equations of dynamics -> Defined Model parameters obtained from the reference paper -> Initialised all the state variables to zero -> Define control input (by user for solving dynamics equations) -> Iterated a 'for loop' in the defined time range for approximating all the state variables -> numerically approximated solution is obtained.

Model parameters used are taken from the reference -

"Modelling and PID controller design for a quadrotor unmanned air vehicle"-

https://ieeexplore.ieee.org/document/5520914

PID Implementation:

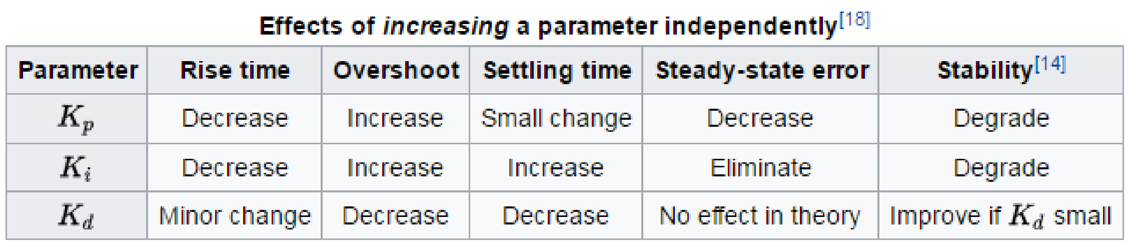
For implementing the PID controller for altitude control of Quadrotor the dynamics is simplified. The roll, pitch and yaw angles are assumed to be zero.

This simplifies the dynamics equations.

As in the above case Runge kutta method of 4th order is used for solving dynamics.

Altitude controller using PID:

A PID controller is used to stabilize the system output, tuning the Kp, Kd, Ki parameters. The required transient and steady state response is achieved using the PID controller. Dependency of system output with parameters of PID -



Effects of *increasing* a parameter independently[18]

Parameter	Rise time	Overshoot	Settling time	Steady-state error	Stability[14]
K_p	Decrease	Increase	Small change	Decrease	Degrade
K_i	Decrease	Increase	Increase	Eliminate	Degrade
K_d	Minor change	Decrease	Decrease	No effect in theory	Improve if K_d small

Pseudo Code for implementing PID controller for altitude control:

Class PID defined with init and declared all the parameters of init -> defined function resetValues in class PID to reset all the intermediate values (error, dt, etc) -> defined pid execute function -> Executed PID algorithm -> defined class solver_z for solving dynamics for altitude control -> initialised model parameters -> defined function for returning the linear system of dynamics -> implemented solver using Runge kutta method of 4th order -> In the main code initialised all the required parameters and setpoint / desired altitude to 10m -> declared lists for tracking error, altitude, velocity and control input required -> In a while loop called the function solve in class solve_z to solve dynamics -> called the function pidexecute to get the control input required to reach the desired setpoint.

In the PID class, defined a function resetvalues - This is used when the setpoint of the system is changed (as in real life application), it resets all the previously stored values of error, last sum.

In pidexecute, limited the control input parameters within a range so that it doesn't cause sudden jerk in system causing disruptions.

The initial value of control input i.e., total vertical thrust is considered as ‘g = 9.8’ inorder to avoid negative values. The desired output values of altitude are always positive. So, minimum input thrust of ‘g’ is required. (thrust per unit mass)

The last step executes the altitude control algorithm such that, the dynamics of previous iterate are used to find the required control input to reach setpoint and again the obtained control input from PID algorithm is used to find the new dynamics parameters.

Tuning Parameters of PID:

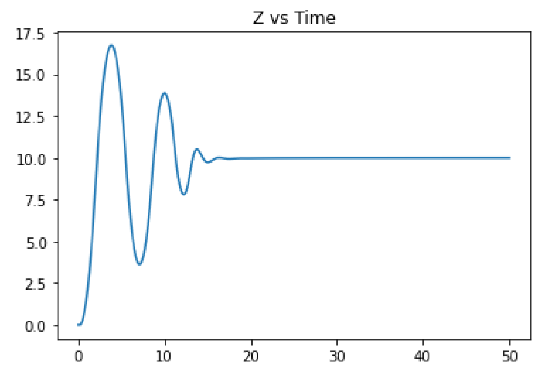
The parameters of PID are tuned generally to obtain:

- Low overshoot
- Less oscillations
- Very low steady state error

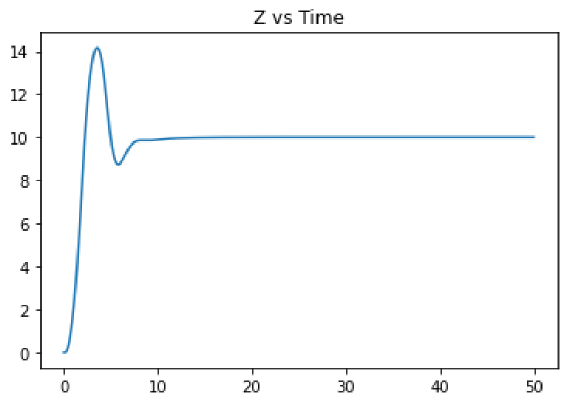
As stated in the above table-

Kp is increased so that the overshoot of the system is not high and the required time response of the system is reached. Kd value is increased inorder to reduce oscillations in the system by damping the system. Ki is increased to reduce the steady state error of the system.

At kp=5, kd = 2, ki = 2 we observe,

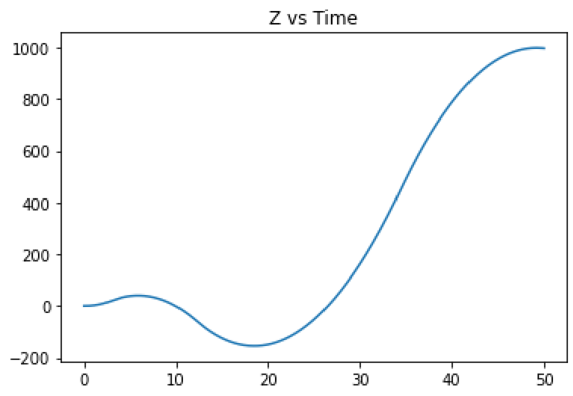


There is significant overshoot seen. So on reducing kp to 3 we get,



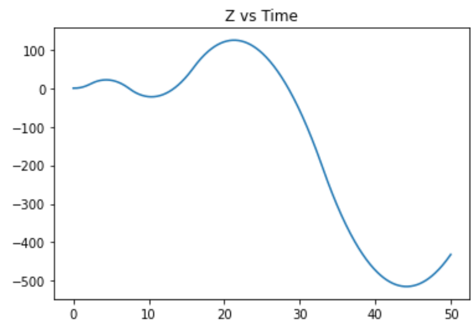
Here the overshoot is decreased yet significant,

Effect of ki is seen as, at ki = 5, kd =2 , kp = 2, the response is,



This indicates Highly overdamped situation. To overcome this on reducing ki as ki =2 we obtain desired results.

The parameter Kd is tuned from initial value of kd = 0.5, where response is,

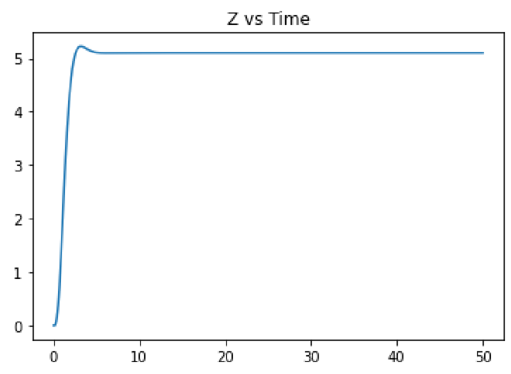


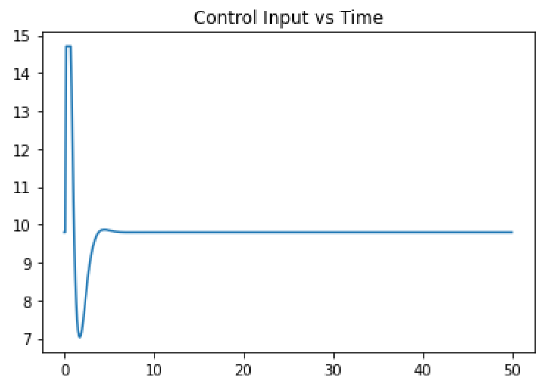
We observe that system is highly unstable accompanied by oscillations. to overcome kd is increased to kd = 2, at other parameters calculated as above.

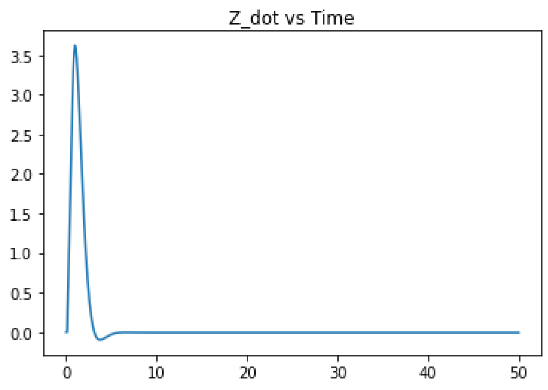
Further reducing kp = 2, ki = 1, kd = 2 we obtain the desired response.

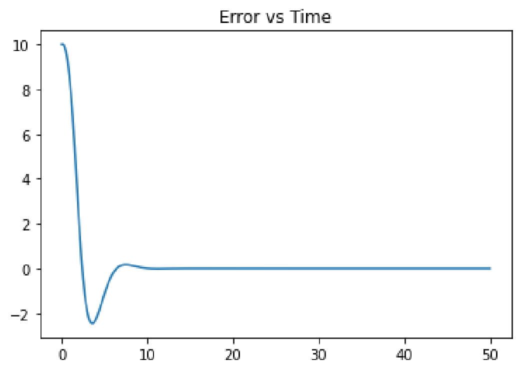
Required response is achieved at -
kp = 2
kd = 2
ki = 1

Plots obtained at optimal parameters of PID:









Reference used for PID implementation-

https://scholarcommons.sc.edu/etd/406

Further Works:

1. In the dynamic modelling the wind effects were ignored. Also some effects such as Blade flapping which occurs due to external disturbances are ignored in this model.

   reference-

   http://ai.stanford.edu/~gabeh/papers/ICRA09_AeroEffects.pdf\

2. Aerodynamic drag is assumed to be linearly dependent on velocity, which is not the case for Terrestrial regions with high density variations on the planet.
3. While solving the differential equations using Runge kutta there is an error obtained in the method, Modern Learning based methods for solving ODEs of multi variables can be opted.
4. PID tuning could be done better using better tuning techniques.