These are modified pages of the thesis **Model Predictive Controller of a UAV using the LPV approach**, written by Mark Misin in order to adapt certain nuances to the Mark Misin's course **Applied Control Systems for Engineers 2 - UAV drone control**. The modifications were made by Mark Misin.

$$
\begin{aligned}
U1 &= m\ddot{z} \\
U2 &= I_x\ddot{\phi} \\
U3 &= I_y\ddot{\theta} \\
U4 &= I_z\ddot{\psi}
\end{aligned}
\tag{2.1}
$$

In case of a reference tracking problem, the closed loop controller for the UAV gives the input signals U1, U2, U3 and U4 directly to the drone. Based on these inputs, the four rotors of the UAV rotate accordingly. That means that there must be a relationship between the input signals and the angular velocities of the rotors. In the system of equations (2.2) [3], it can be seen very clearly how the control input signals are related to the angular velocities of the rotors, which are denoted as $\Omega_1,\ \Omega_2,\ \Omega_3\ and\ \Omega_4\ [rad \cdot s^{-1}]$ for the motors 1, 2, 3 and 4, respectively. The values of $c_T\ [Ns^2]\ and\ c_Q\ [Nms^2]$ are aerodynamic coefficients of thrust and drag, respectively [1]. The value of $l\ [m]$ is the distance between the center of the quadrotor and the center of a propeller [1]. Finally, the equation (2.3) adds up the rotational velocities of all the rotors [1]. Since the propellers 1 and 3 rotate counter-clockwise and the propellers 2 and 4 rotate clockwise, the motors 1 and 3 have the opposite sign compared to the motors 2 and 4

$$
\begin{aligned}
U1 &= c_T \cdot (\Omega_1^2 + \Omega_2^2 + \Omega_3^2 + \Omega_4^2) \\
U2 &= c_T \cdot l \cdot (\Omega_2^2 - \Omega_4^2) \\
U3 &= c_T \cdot l \cdot (\Omega_3^2 - \Omega_1^2) \\
U4 &= c_Q \cdot (-\Omega_1^2 + \Omega_2^2 - \Omega_3^2 + \Omega_4^2)
\end{aligned}
\tag{2.2}
$$

such that

$$
\Omega_{total} = \Omega_1 - \Omega_2 + \Omega_3 - \Omega_4 \tag{2.3}
$$

The final step needed to start building the controller for the drone is to obtain its state space equations. It is needed to have the drone's mathematical model in that form because it must only contain first order differentiation. The reason for that is because in the implementation of the controller a MATLAB® ode45 integrator will be used to integrate the system's states in time.

However, the integrator only accepts first order systems; hence, the equations of motion need to be made first order, which will be done in the next section.

2.3 The mathematical model of a quadrotor

A quadcopter has six degrees of freedom - 3 position and 3 attitude dimensions. These are $x,\ y,\ z\ and\ \phi,\ \theta,\ \psi$, respectively. The mathematical model of a UAV has to incorporate all the degrees of freedom. One way to express a mathematical model of a drone is to write it in the B-frame. The system of equations (2.4) describes the drone in the B-frame [3]. The variables u, v and w are x, y and z velocities in the B-frame $[ms^{-1}]$, respectively. The variables p, q and r are the angular velocities of $\phi,\ \theta,\ \psi$ in the B-frame $[rad\ s^{-1}]$, respectively. The Ω is the added rotation of all the rotors that comes from the equation (2.3) [1]. The constant g is the gravitational acceleration on the surface of the Earth, which is 9.81 ms^{-2}. Finally, the constant J_{TP} $[Nms^2]$ is is the total rotational moment of inertia around the propeller axis [3]

$$
\begin{aligned}
\dot{u} &= (vr - wq) + g\sin\theta \\
\dot{v} &= (wp - ur) - g\cos\theta\sin\phi \\
\dot{w} &= (uq - vp) - g\cos\theta\cos\phi + \frac{U_1}{m} \\
\dot{p} &= qr\frac{I_y - I_z}{I_x} + \frac{J_{TP}}{I_x}q\Omega + \frac{U_2}{I_x} \\
\dot{q} &= pr\frac{I_z - I_x}{I_y} - \frac{J_{TP}}{I_y}p\Omega + \frac{U_3}{I_y} \\
\dot{r} &= pq\frac{I_x - I_y}{I_z} + \frac{U_4}{I_z}
\end{aligned}
\tag{2.4}
$$

The system of equations (2.4) is in a convenient form because it only contains first order differentiation [3] . However, the problem with expressing everything in the B-frame is that now all six degrees of freedom states are velocities. However, the trajectory is given in the position values of x, y and z in the E-frame. Therefore, it is needed to have a system of equations in which the translational motion states are in the E-frame position format. The rotational motion states can stay in the B-frame as angular velocities. This Hybrid-frame (H-frame) can be seen in the equation (2.5) [3].

$$
\begin{aligned}
\ddot{x} &= (\cos\phi \sin\theta \cos\psi + \sin\phi \sin\psi)\frac{U_1}{m} \\
\ddot{y} &= (\cos\phi \sin\theta \sin\psi - \sin\phi \cos\psi)\frac{U_1}{m} \\
\ddot{z} &= -g + \cos\phi \cos\theta \frac{U_1}{m} \\
\dot{p} &= qr\frac{I_y - I_z}{I_x} + \frac{J_{TP}}{I_x} q\Omega + \frac{U_2}{I_x} \\
\dot{q} &= pr\frac{I_z - I_x}{I_y} - \frac{J_{TP}}{I_y} p\Omega + \frac{U_3}{I_y} \\
\dot{r} &= pq\frac{I_x - I_y}{I_z} + \frac{U_4}{I_z}
\end{aligned}
\tag{2.5}
$$

The H-frame contains the position variables in the E-frame. However, now the problem is that it has second order differentiation in it. The MATLAB® integrator ode45 needs first order differential equations though. In addition, the H-frame does not contain the angles ϕ, θ, ψ, which are the orientation of a drone in the E-frame. To solve these two problems, the H-frame system of equations can be expanded. The relationship between the E and B-frame can be used to create one large system of equations that contains all the six states in the B and also in the E-frame - in total, 12 states, which are: u, v, w, p, q, r, x, y, z, ϕ, θ, ψ. This system of equations would be first order and therefore suitable for the ode45 integrator. The rotational matrix under the Z-Y'-X" Euler angles convention that relates the translational velocities in the B-frame (u, v, w) to the translational velocities in the E-frame ($\dot{x}$ $\dot{y}$ $\dot{z}$) can be seen in the equation (2.6) [2]. The transformation matrix that relates the angular velocities in the B-frame (p, q, r) to the angular velocities in the E-frame ($\dot{\phi}$, $\dot{\theta}$, $\dot{\psi}$) can be seen in the equation (2.7) [3]

$$
R = \begin{bmatrix}
\cos\theta \cos\psi & \sin\phi \sin\theta \cos\psi - \cos\phi \sin\psi & \cos\phi \sin\theta \cos\psi + \sin\phi \sin\psi \\
\cos\theta \sin\psi & \sin\phi \sin\theta \sin\psi + \cos\phi \cos\psi & \cos\phi \sin\theta \sin\psi - \sin\phi \cos\psi \\
-\sin\theta & \sin\phi \cos\theta & \cos\phi \cos\theta
\end{bmatrix}
\tag{2.6}
$$

$$
T = \begin{bmatrix}
1 & \sin\phi \tan\theta & \cos\phi \tan\theta \\
0 & \cos\phi & -\sin\phi \\
0 & \sin\phi \sec\theta & \cos\phi \sec\theta
\end{bmatrix}
\tag{2.7}
$$

Now, there are all the tools needed to build a global system of equations with all the states from the both frames and that is also first order, suitable for the ode45 integrator.The global open-

loop system that will be integrated in this thesis while running the simulations can be seen in the equation (2.8). The system of equations in this configuration allows all the states (u, v, w, p, q, r, x, y, z, ϕ, θ, ψ) to be tracked. Once an initial value is given to them, the equation (2.8) computes their derivatives and then it is possible to know the state values in the next sample time period

$$
\begin{aligned}
\dot{u} &= (vr - wq) + g\sin\theta \\
\dot{v} &= (wp - ur) - g\cos\theta\sin\phi \\
\dot{w} &= (uq - vp) - g\cos\theta\cos\phi + \frac{U_1}{m} \\
\dot{p} &= qr\frac{I_y - I_z}{I_x} + \frac{J_{TP}}{I_x}q\Omega + \frac{U_2}{I_x} \\
\dot{q} &= pr\frac{I_z - I_x}{I_y} - \frac{J_{TP}}{I_y}p\Omega + \frac{U_3}{I_y} \\
\dot{r} &= pq\frac{I_x - I_y}{I_z} + \frac{U_4}{I_z} \\
\begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{z} \end{bmatrix} &= R \begin{bmatrix} u \\ v \\ w \end{bmatrix} \\
\begin{bmatrix} \dot{\phi} \\ \dot{\theta} \\ \dot{\psi} \end{bmatrix} &= T \begin{bmatrix} p \\ q \\ r \end{bmatrix}
\end{aligned}
\tag{2.8}
$$

$$\begin{bmatrix} \dot{x} \\ \ddot{x} \\ \dot{y} \\ \ddot{y} \\ \dot{z} \\ \ddot{z} \end{bmatrix} = \begin{bmatrix} 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} x \\ \dot{x} \\ y \\ \dot{y} \\ z \\ \dot{z} \end{bmatrix} + \begin{bmatrix} 0 \\ (\cos\phi \sin\theta \cos\psi + \sin\phi \sin\psi)\frac{1}{m} \\ 0 \\ (\cos\phi \sin\theta \sin\psi - \sin\phi \cos\psi)\frac{1}{m} \\ 0 \\ -\frac{g}{U_1} + \cos\phi \cos\theta \frac{1}{m} \end{bmatrix} U_1 \tag{3.1}$$

In order to have an LPV model for the angles, the last 3 equations of a system of equations (2.5) will be considered [3]. However, these equations are in the B-frame. To control the drone, the LPV model needs to contain the angles and their instantaneous changes in the E-frame. In case of angles, both of the frames are related to each other by the transformation matrix T in equation (2.7) [3].

However, here a simplifying assumption can be made that affects how the drone is controlled, insignificantly. The quadcopter cannot hover in one position if it is tilted in one direction all the time - it would start sliding down diagonally. The goal is to design a controller that stabilizes the quadcopter close to the hovering position. In this case, the angles ϕ and θ are assumed to be zero, which makes the T matrix in the equation (2.7) an identity matrix I. This converts the last three equations in the system of equations (2.5) [3] into a system of equations, in which p, q, r become $\dot{\phi}$, $\dot{\theta}$, $\dot{\psi}$, respectively [3]. It can be seen in the equation (3.2) [3], which is also second order

$$\begin{aligned} \ddot{\phi} &= \dot{\theta}\dot{\psi}\frac{I_y - I_z}{I_x} + \frac{J_{TP}}{I_x}\dot{\theta}\Omega + \frac{U_2}{I_x} \\ \ddot{\theta} &= \dot{\phi}\dot{\psi}\frac{I_z - I_x}{I_y} - \frac{J_{TP}}{I_y}\dot{\phi}\Omega + \frac{U_3}{I_y} \\ \ddot{\psi} &= \dot{\phi}\dot{\theta}\frac{I_x - I_y}{I_z} + \frac{U_4}{I_z} \end{aligned} \tag{3.2}$$

Just like it was done with the position variables, to generate an LPV format for the angles, the system of equations (3.2) [3] was expanded. The LPV format is shown in equation (3.3) [3], in which it can be seen that all the nonlinearities are encapsulated in the A matrix. The B matrix only has constant values. The A matrix is multiplied by the states and the B matrix is multiplied by the inputs

$$\begin{bmatrix} \dot{\phi} \\ \ddot{\phi} \\ \dot{\theta} \\ \ddot{\theta} \\ \dot{\psi} \\ \ddot{\psi} \end{bmatrix} = \begin{bmatrix} 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & \frac{\Omega \cdot J_{TP}}{I_x} & 0 & \dot{\theta} \cdot \frac{I_y - I_z}{I_x} \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & -\frac{\Omega \cdot J_{TP}}{I_y} & 0 & 0 & 0 & \dot{\phi} \cdot \frac{I_z - I_x}{I_y} \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & \frac{\dot{\theta}}{2} \cdot \frac{I_x - I_y}{I_z} & 0 & \frac{\dot{\phi}}{2} \cdot \frac{I_x - I_y}{I_z} & 0 & 0 \end{bmatrix} \begin{bmatrix} \phi \\ \dot{\phi} \\ \theta \\ \dot{\theta} \\ \psi \\ \dot{\psi} \end{bmatrix} + \begin{bmatrix} 0 & 0 & 0 \\ \frac{1}{I_x} & 0 & 0 \\ 0 & 0 & 0 \\ 0 & \frac{1}{I_y} & 0 \\ 0 & 0 & 0 \\ 0 & 0 & \frac{1}{I_z} \end{bmatrix} \begin{bmatrix} U_2 \\ U_3 \\ U_4 \end{bmatrix} \tag{3.3}$$

In the second control strategy with the position controller, only the equation (3.3) is used in the MPC control [3]. That is because position is handled by the feedback linearization strategy that does not require an LPV model of the system. However, in the first control strategy, where only the LPV-MPC method is used for the entire system, both LPV models are combined into one single state space system of equations as can be seen in equation (3.4). That is used for the MPC in the first control strategy, where only the LPV-MPC method is used to control the entire drone

$$\begin{bmatrix} \dot{x} \\ \ddot{x} \\ \dot{y} \\ \ddot{y} \\ \dot{z} \\ \ddot{z} \\ \dot{\phi} \\ \ddot{\phi} \\ \dot{\theta} \\ \ddot{\theta} \\ \dot{\psi} \\ \ddot{\psi} \end{bmatrix} = \begin{bmatrix} A_{LPV-(x-y-z)} & zeros(6,6) \\ zeros(6,6) & A_{LPV-(\phi-\theta-\psi)} \end{bmatrix} \begin{bmatrix} x \\ \dot{x} \\ y \\ \dot{y} \\ z \\ \dot{z} \\ \phi \\ \dot{\phi} \\ \theta \\ \dot{\theta} \\ \psi \\ \dot{\psi} \end{bmatrix} + \begin{bmatrix} B_{LPV-(x-y-z)} & zeros(6,3) \\ zeros(6,1) & B_{LPV-(\phi-\theta-\psi)} \end{bmatrix} \begin{bmatrix} U_1 \\ U_2 \\ U_3 \\ U_4 \end{bmatrix} \tag{3.4}$$

3.4 The architecture of the LPV-MPC controller

In this section, the architecture of the MPC controller is described - its schematic can be seen in Figure 3.4. Since, in the end, a second control strategy in Figure 3.2 was used, the one with the position controller, the MPC architecture described in this section is made to fit this control plan. The dotted line in red is where the control loop goes from one sample time to another. The integration using the MATLAB® ode45 integrator happens in the nonlinear model block, which represents the open loop system. The angular velocities in the B-frame and the angles themselves in the E-frame, together with the total rotational velocity of the rotors from the previous sample time period, are then sent to the continuous LPV block, where the nonlinear model is transformed into an LPV model. The angles in the E-frame are also sent directly to the cost function, because it needs the present angular values as can be seen in equation (3.22) [8].

The LPV model is continuous; however, the controller works discretely. Therefore, the LPV system needs to be discretized. The discrete LPV block takes in the A, B, C and D matrices from the continuous LPV block and discretizes them using the forward Euler method or the zero-order-hold (zoh) method - both are available in the code. The discretized matrices are then used to generate the H and F^T matrices in the H and F^T matrix generation block. They are then sent to the cost function block, which also receives a reference angle vector for the entire horizon period. The matrix H and the vector f^T are then sent to the MATLAB® quadprog optimizer, where a sequence of Δu-s are found that minimizes the cost function. The first element of that sequence is used to calculate the absolute input in the current sample time period. It can be seen in the following

$$U_{t+k} = U_{t+k-1} + \begin{bmatrix} I_{3x3} & 0_{3x3} & 0_{3x3} & 0_{3x3} \end{bmatrix} \begin{bmatrix} \Delta u1_{t+k} \\ \Delta u2_{t+k} \\ \Delta u3_{t+k} \\ \Delta u4_{t+k} \end{bmatrix} \tag{3.23}$$

for the horizon period of 4 samples, which is the case in the control strategy in Figure 3.2 It is then fed into the nonlinear model block. The absolute U-s are also used to compute the total rotational velocity of the rotors in the present sample period, which is also sent to the nonlinear model block. Then, the integration happens and the entire process starts all over again.

3.5 The position controller

This section describes the feedback linearization method in the position controller that is used in the second control strategy in Figure 3.2. The system of the second order differential equations that govern the UAV's position are [3]

$$\begin{aligned} \ddot{x} &= (\cos\phi \sin\theta \cos\psi + \sin\phi \sin\psi)\frac{U_1}{m} \\ \ddot{y} &= (\cos\phi \sin\theta \sin\psi - \sin\phi \cos\psi)\frac{U_1}{m} \\ \ddot{z} &= -g + \cos\phi \cos\theta \frac{U_1}{m} \end{aligned} \tag{3.24}$$

The system can be written out as a system of first order differential equations as it can be seen in the following [3]

$$\begin{aligned} \dot{x_1} &= x_2 \\ \dot{x_2} &= (\cos\phi \sin\theta \cos\psi + \sin\phi \sin\psi)\frac{U_1}{m} \\ \dot{x_3} &= x_4 \\ \dot{x_4} &= (\cos\phi \sin\theta \sin\psi - \sin\phi \cos\psi)\frac{U_1}{m} \\ \dot{x_5} &= x_6 \\ \dot{x_6} &= -g + \cos\phi \cos\theta \frac{U_1}{m} \end{aligned} \tag{3.25}$$

The system can be written out as a system of first order differential equations as it can be seen in equation (3.25) [3], where $x_1 = x,\ x_2 = \dot{x},\ x_3 = y,\ x_4 = \dot{y},\ x_5 = z,\ x_6 = \dot{z}$ [3].

Besides position reference values x, y, z, the planner also needs to provide the position controller with the reference velocities $\dot{x},\ \dot{y},\ \dot{z}$ and reference accelerations $\ddot{x},\ \ddot{y},\ \ddot{z}$.

Next, the errors of the position and velocity values are computed. The velocity errors are differentiated one more time to get the acceleration of the error values. These errors are calculated as [3].

$$
\begin{aligned}
e_x &= x_t^R - x_t \quad & \dot{e_x} &= \dot{x_t^R} - \dot{x_t} \quad & \ddot{e_x} &= \ddot{x_t^R} - \ddot{x_t} \\
e_y &= y_t^R - y_t \quad & \dot{e_y} &= \dot{y_t^R} - \dot{y_t} \quad & \ddot{e_y} &= \ddot{y_t^R} - \ddot{y_t} \\
e_z &= z_t^R - z_t \quad & \dot{e_z} &= \dot{z_t^R} - \dot{z_t} \quad & \ddot{e_z} &= \ddot{z_t^R} - \ddot{z_t}
\end{aligned}
\tag{3.26}
$$

The variables $\ddot{e_x},\ \ddot{e_y},\ \ddot{e_z}$ are then chosen to be a control action for the linearized state feedback control strategy as [3].

$$
\begin{aligned}
\ddot{e_x} &= k_1^x e_x + k_2^x \dot{e_x} \\
\ddot{e_y} &= k_1^y e_y + k_2^y \dot{e_y} \\
\ddot{e_z} &= k_1^z e_z + k_2^z \dot{e_z}
\end{aligned}
\tag{3.27}
$$

From equation 3.24 [3], the variables $\ddot{x_t},\ \ddot{y_t},\ \ddot{z_t}$ can be formed by subtracting second derivative errors from the second derivative reference values in x, y, and z direction respectively, like this [3]

$$
\begin{aligned}
\ddot{x_t} &= \ddot{x_t^R} - \ddot{e_x} = (\cos\phi \sin\theta \cos\psi + \sin\phi \sin\psi)\frac{U_1}{m} \\
\ddot{y_t} &= \ddot{y_t^R} - \ddot{e_y} = (\cos\phi \sin\theta \sin\psi - \sin\phi \cos\psi)\frac{U_1}{m} \\
\ddot{z_t} &= \ddot{z_t^R} - \ddot{e_z} = (-g + \cos\phi \cos\theta \frac{U_1}{m})
\end{aligned}
\tag{3.28}
$$

By choosing negative real poles, the constants in equation (3.27) can be computed [3]. Then, the values $\ddot{x_t},\ \ddot{y_t},\ \ddot{z_t}$ are determined. In order to find the angles θ and ϕ for the attitude controller, which it will then use as reference angles, the equations (3.29) [3] and (3.30) (depending on the reference yaw angle value) [3] are used, respectively. The constants a, b, c and d are calculated as [3]

$$
\theta = tan^{-1}(ac + bd) \tag{3.29}
$$

$$
\phi = tan^{-1}(\frac{\cos(\theta)(\tan(\theta)d - b)}{c}) = tan^{-1}(\frac{\cos(\theta)(a - \tan(\theta)c)}{d}) \tag{3.30}
$$

$$a = \frac{\ddot{x}_t}{\ddot{z}_t + g},\ b = \frac{\ddot{y}_t}{\ddot{z}_t + g},\ c = \cos\psi_{ref},\ d = \sin\psi_{ref} \tag{3.31}$$

$$U_1 = \frac{(\ddot{z}_t + g)\, m}{\cos\phi \cos\theta} \tag{3.32}$$

These angles, along with the ψ^R angle from the planner, will serve as reference angles for the LPV-MPC controller. However, the position controller also finds the input U1, that is directly fed into the nonlinear model. The control action U1 can be computed in equation (3.32) [3].

3.6 Implementation of the position and the LPV-MPC controller

In Figure 3.6, one can see the structure of the control strategy code. The code itself is in the Appendix B. The script consists of one MAIN file and six supportive functions. It is approximately shown with arrows in which location in the main file the functions are used. The initial constants function is a library type function in which one can find constants and certain initial values. This function also supports the other functions in this code.

The MAIN file first loads the constants and the initial values. It gets the trajectory, the reference velocities and the reference yaw angle from the trajectory generator (planner). Then, the outer loop begins where the position controller function is used. After that, the LPV-MPC loop starts that uses a function to get the discrete LPV model. The inner loop loops through the code 4 times per 1 loop of the outer loop. In the MPC simplification function, the necessary matrices for the solver are generated. After that, the solver is called. The results are then fed into the nonlinear drone model, where the integration of the open loop system happens. Finally. the results are plotted, which can be seen in the validation chapter.