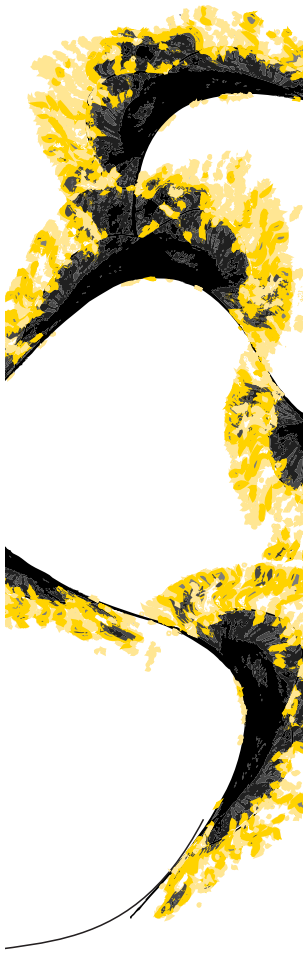
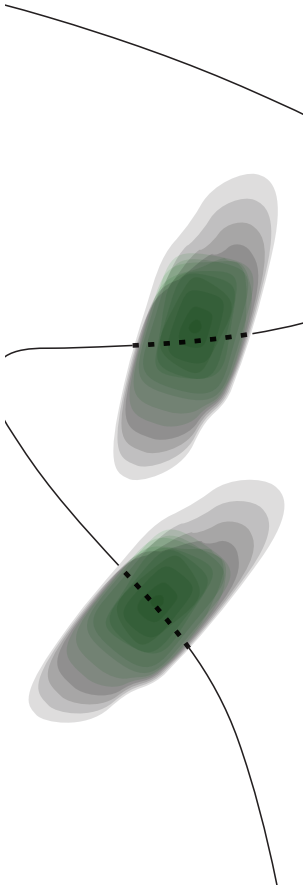


RAM ROBOTICS AND MECHATRONICS

LOAD TRANSPORTATION WITH AN ELASTIC CABLE: MODELING AND CONTROL

N. (Niels) ter Heerdt

BSC ASSIGNMENT



Committee:
prof. dr. ir. A. Franchi
dr. S. Sun
Y. Shen
dr. ing. A. Lavrenko

July, 2023

029RaM2023
Robotics and Mechatronics
EEMathCS
University of Twente
P.O. Box 217
7500 AE Enschede
The Netherlands

UNIVERSITY OF TWENTE. | TECHMED CENTRE

UNIVERSITY OF TWENTE. | DIGITAL SOCIETY INSTITUTE

***Abstract*—In this paper, research was done to see how cable elasticity affects the behaviour of a quadrotor Unmanned Aerial Vehicles (UAVs) carrying a slung load. The research consists of two main steps. In the first step, a non-linear model was derived using the Newton-Euler method. This resulted in the rotational and translational equations of motion of the system. The model was then simulated and validated using Simulink. The second step was to design a controller for the UAV. This was done by finding the equilibrium for hovering, which is used as a basis for the linearization of the non-linear system. The linearization resulted in the Jacobian matrix A and input matrix B, these were validated using the linmod function. With this A and B matrix, an LQR controller for the system was designed and implemented in Simulink. The controller was the tested by letting it track a certain position and a circular path.**

I. Introduction

Multi-rotor Unmanned Aerial Vehicles (UAVs) have been applied for load transportation, such as package delivery. In the future it might even be applied in flying taxis. However, attaching load, especially those with large sizes, directly underneath the UAV can disturb the airflow detrimentally and can consequently reduce the aerodynamic efficiency. The sling load operation is a solution to tackle this issue, in which the load is connected to the UAV through a tether. Literature already tackles this problem by neglecting the elasticity of the cable, treating it as a rigid link, hence lacking accuracy. [1] [2] There has also already been research done into elastic slung loads, but these make use of non-linear control strategies to control the system. [3]. Although there has been a lot of research, there is no literature about linear control for these dynamic systems. In order to know more about how the elasticity of the cable affects the system behaviour, consider that the system consists of a single quadrotor (UAV), a point-mass load, and an elastic cable. The load and cable can be simplified as a mass-spring-damper system. This brings about the following question: "How does the elasticity in the cable affect the system behaviour of a quadrotor UAV". In this paper a non-linear model is derived. This model was then linearized and used to design an LQR controller.

II. Dynamic model

To derive the dynamic model, the free body diagram is used. For this derivation a couple of simplifications were made namely:

- The elastic slung load is connected to the centre of mass (COM) of the quadrotor
- The frame of the drone is considered rigid
- The load is considered a point mass

The system to be considered is a quadrotor which is carrying an elastic slung load. The load is modelled by a point mass connected with a spring damper to the centre of the UAV. In figure 1 the free body diagram of the model is shown. With the Q and T vectors showing the reaction torque and thrust respectively. The rigid frame is sketched by a black cross and the force due to the cable is shown as F_{cable}. In this paper, two frames will be considered the (right hand) reference frame denoted by subscript 'e' and the (right hand) body fixed frame (BFF) which is rigidly connected to the centre of mass of the UAV, with the x- and y-axis being on the frame of the quadcopter.

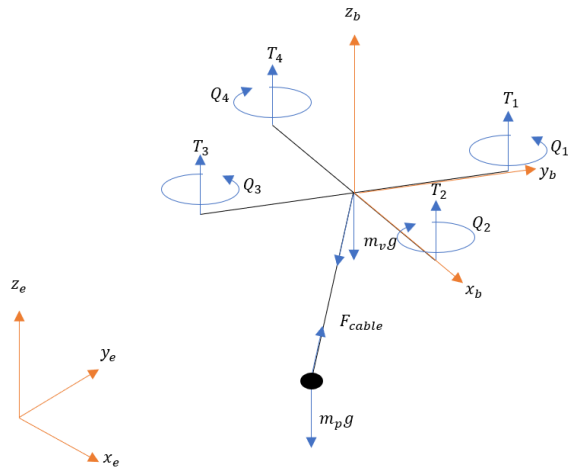


Fig. 1. Free body diagram of the system

To be able to describe the system's motion, the position of the UAV and point mass were taken at the COM expressed in the reference frame. This is expressed by the vector:

$$\mathbf{p_e} = [x_v, y_v, z_v, x_p, y_p, z_p]^T \tag{1}$$

To derive the dynamic equation the Newton Euler method is used. [4] The translational equation of motion can be described by using the sum of forces acting on the system in the reference frame. To convert between the body-fixed frame and reference frame a rotation matrix can be constructed such that $\mathbf{r_e} =^e R_b \mathbf{r_b}$. One way to describe this rotation is with the Euler angles, in this paper the Z-Y-X [5] order is used. Here, the angles ϕ,θ andψ correspond to rotations around the Z-, Y- and X- axis. The derivation for this can be found in appendix B.

$$^eR_b = \begin{bmatrix} c\psi c\theta & c\psi s\phi s\theta - c\phi s\psi & s\psi s\phi + c\psi c\phi s\theta \\ c\theta s\psi & c\psi c\phi + s\psi s\phi s\theta & c\phi s\psi s\theta - c\psi s\phi \\ -s\theta & c\theta s\phi & c\psi c\theta] \end{bmatrix} \tag{2}$$

Here, the s and c stand for sine and cosine respectively. To convert from reference to BFF the transpose of this rotation matrix is used.

A. Translational

Now that all the forces can be described in the reference frame, the translational equations of motion become:

$$\mathbf{M\ddot{p}} = \mathbf{F_g} + \mathbf{F_t} + \mathbf{F_c} \tag{3}$$

[3] Here M denotes the mass matrix described by:

$$\mathbf{M} = \begin{bmatrix} m_v I_{3x3} & \mathbf{0_{3x3}} \\ \mathbf{0_{3x3}} & m_p I_{3x3} \end{bmatrix} \tag{4}$$

With F_g describing the forces due to gravity, F_t describing the forces due to thrust and F_c describing the tension force

that is applied by the cable to the UAV and payload.
The thrust force is in the positive z- direction in BFF and thus needs to be rotated back to the reference frame using equation II. The trust generated by the rotors can be described by the use of aerodynamics where the thrust for one of the rotors is proportional to:

$$T_i = c_T * \omega_i^2 \tag{5}$$

The total thrust produced by the rotors can be calculated by: [4]

$$T_\Sigma = \Sigma_{i=1}^4 T_i = c_T \Sigma_{i=1}^4 \omega_i^2 \tag{6}$$

The thrust is always in the z direction in BFF resulting when rotated back in:

$$\mathbf{F_t} = \begin{bmatrix} (s\phi s\psi + c\phi c\psi s\theta)T_\Sigma \\ (c\phi s\psi s\theta - c\psi s\phi)T_\Sigma \\ (c\phi c\theta)T_\Sigma \\ 0 \\ 0 \\ 0 \end{bmatrix} \tag{7}$$

The force due to gravity is always pointing down in the reference frame from the COM of the UAV this then gives:

$$\mathbf{F_g} = \begin{bmatrix} 0 \\ 0 \\ -m_v g \\ 0 \\ 0 \\ -m_p g \end{bmatrix} \tag{8}$$

The tension force in the cable is modelled as a spring damper resulting in a mass spring damper system. The force that the cable is acting on the mass is negative to the force acting on the UAV due to Newton's Third Law. This results in the following:

$$\mathbf{F_c} = \begin{bmatrix} (k_a(l - l_0) + c_a v_r)\hat{\mathbf{e}} \\ -(k_a(l - l_0) + c_a v_r)\hat{\mathbf{e}} \end{bmatrix} \tag{9}$$

In equation II-A k_a is the axial stiffness of the cable and c_a is the dampening coefficient. l_0 represents the initial length of the cable and l is the actual length of the cable which can be described by: $||\mathbf{r_p} - \mathbf{r_v}||$ here $\mathbf{r_p} = [x_p y_p z_p]^T$ is the absolute position of the payload and $\mathbf{r_v} = [\mathbf{x_v y_v z_v}]^\mathbf{T}$ is the absolute position of the UAV. v_r is the absolute relative velocity which is described by: $v_r = (\dot{\mathbf{r_p}} - \dot{\mathbf{r_v}}) \cdot \hat{\mathbf{e}}$. To get the forces in the corresponding direction the tension force is multiplied by the unit direction vector $\hat{\mathbf{e}} = \frac{r_p - r_v}{l}$ where $l = \sqrt{(x_p - x_v)^2 + (y_p - y_v)^2 + (z_p - z_v)^2}$. [3]

B. Rotational

The rotational equation of the UAV stays the same as for a normal UAV this is due to the assumption that the cable is connected to the COM of the UAV. This makes the rotational equation equal to:

$$\mathbf{I} \begin{bmatrix} \dot{p} \\ \dot{q} \\ \dot{r} \end{bmatrix} = \begin{bmatrix} U_2 \\ U_3 \\ U_4 \end{bmatrix} - \begin{bmatrix} p \\ q \\ r \end{bmatrix} \times \mathbf{I} \begin{bmatrix} p \\ q \\ r \end{bmatrix} \tag{10}$$

In equation 10, the I stands for the inertia matrix which is:

$$\mathbf{I} = \begin{bmatrix} I_{xx} & 0 & 0 \\ 0 & I_{yy} & 0 \\ 0 & 0 & I_{zz} \end{bmatrix} \tag{11}$$

The p, q and r are the rotations around the BFF axes. For the inputs U2 U3 and U4 are considered which are the moments around the x-, y- and z- axis of the BFF respectively. The rotors create a reaction torque on the airframe which can be modelled by:

$$Q_i = c_Q \omega_i^2 \tag{12}$$

This will generate a certain torque around the z- axis of the BFF. The moments around the x- and y-axis can be achieved by increasing and decreasing the thrust on opposing sides, because the thrust difference between the rotors times the distance will cause a moment around the axis. The moments around the BFF are summarized in equation 13.

$$\begin{bmatrix} U_2 \\ U_3 \\ U_4 \end{bmatrix} = \begin{bmatrix} 0 & dc_t & 0 & -dc_t \\ -dc_t & 0 & dc_t & 0 \\ -c_Q & c_Q & -c_Q & c_Q \end{bmatrix} \begin{bmatrix} \omega_1^2 \\ \omega_2^2 \\ \omega_3^2 \\ \omega_4^2 \end{bmatrix} \tag{13}$$

During the simulation, the Euler angles are needed, but the physics only determines the rotations around the axis, so a mapping is needed between the rotational rate and the Euler angles. This can be achieved by considering only a small change in the Euler angle and inspecting what the influence is on the rotational rate. This results in the mapping shown in equation 14.

$$\begin{bmatrix} p \\ q \\ r \end{bmatrix} = \begin{bmatrix} 1 & 0 & -s\theta \\ 0 & c\phi & s\phi c\theta \\ 0 & -s\phi & c\phi c\theta \end{bmatrix} \begin{bmatrix} \dot{\phi} \\ \dot{\theta} \\ \dot{\psi} \end{bmatrix} \tag{14}$$

To find the Euler angle rate as a function of the rotational rate, the inverse of this rotation matrix is taken resulting in equation 15. [5]

$$\begin{bmatrix} \dot{\phi} \\ \dot{\theta} \\ \dot{\psi} \end{bmatrix} = \begin{bmatrix} 1 & s\phi t\theta & c\phi t\theta \\ 0 & c\phi & -s\phi \\ 0 & \frac{s\phi}{c\theta} & \frac{c\phi}{c\theta} \end{bmatrix} \begin{bmatrix} p \\ q \\ r \end{bmatrix} \tag{15}$$

III. Hover equilibrium

The next step after deriving the dynamic equations, is to find the equilibrium for hover. To find the equilibrium for hovering it is needed to have the accelerations, velocities and Euler angles equal to 0. This simplifies equation 3 to equation 16. This causes the vector $\hat{\mathbf{e}}$ to simplify to only a z-component, which results in $\begin{bmatrix} 0 & 0 & 1 \end{bmatrix}^T$.

$$\begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ -m_v g \\ 0 \\ 0 \\ -m_p g \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ T_\Sigma \\ 0 \\ 0 \\ 0 \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ k_a(l - l_0) \\ 0 \\ 0 \\ -k_a(l - l_0) \end{bmatrix} \tag{16}$$

This system of equations can be solved by solving for l in the last row resulting in equation 17. This can then be substituted back into the third row which then is simplified to equation 19.

$$l = -\frac{m_p g}{k_a} + l_0 \tag{17}$$

$$-m_v g + T_\Sigma + k_a(-\frac{m_p g}{k_a} + l_0 - l_0) = 0 \tag{18}$$

$$F_z = (m_v + m_p) * g \tag{19}$$

This is the required force in the z-direction for a hovering flight. To find the rotor speeds required for this force equations 6 and 13 are used. Resulting in the following set of system of equations:

$$c_t(\omega_1^2 + \omega_2^2 + \omega_3^2 + \omega_4^2) = (m_v + m_p) * g \tag{20}$$

$$dc_t(\omega_2^2 - \omega_4^2) = 0 \tag{21}$$

$$dc_t(\omega_1^2 - \omega_3^2) = 0 \tag{22}$$

$$dc_q(-\omega_1^2 + \omega_2^2 - \omega_3^2 + \omega_4^2) = 0 \tag{23}$$

The required rotor speeds all need to be the same following the last 3 equations. Solving the first equation results in:

$$\omega_1 = \omega_2 = \omega_3 = \omega_4 = \sqrt{\frac{(m_v + m_p)g}{c_t}} \tag{24}$$

The stationary point around which the equilibrium for hovering can be calculated then follows from finding the steady state of the system. This can be achieved by using the required rotor speeds as input for the simulation. This will then give the steady state coordinates for the UAV and payload. This is simulated using the matlab script described in appendix C.

$$\begin{bmatrix} 0 & 0 & z_{vss} & 0 & 0 & z_{pss} \end{bmatrix}^T \tag{25}$$

A. Validation

To validate if the designed non-linear system and hover equations are correct, two Simulink simulations were made. The first simulation implements the previously described equations of motion. The second simulation makes use of the 6 DoF (Euler angle) block out of the aerospace toolbox. This block uses a different convention for the BFF-axis, namely with the z-axis down. To account for this, all forces and moments are rotated by an angle π around the x-axis resulting in a rotation matrix:

$${}^{6DoF}R_b = \begin{bmatrix} 1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & -1 \end{bmatrix} \tag{26}$$

During these simulations, all initial conditions were set to 0 except for the $z_v = z_{vss}$ and the $z_p = z_{pss}$. The variables used are shown in table III-A.

First, it is verified that it will fall down when there is no input to the system. As can be seen in figure 2, only the z-coordinates of payload and the UAV change. As can be seen in the left and right-hand side of the figure the two graphs are exactly the same. Then to verify if the spring damper connection is working, the input is set to the value that returns from equation 24. The payload and UAV should start oscillating around their equilibrium position. The dampening effect is first neglected to see if the spring is working. As can be seen in figure 3, this corresponds to what happens; the sinusoidal signal keeps the same amplitude and the force between the UAV and payload is inverse as the z_v and z_p go in the same rate but, in negated direction.

Parameter	Value	Parameter	Value
Mv $[kg]$	10	ka $[N/m]$	100
Mp $[kg]$	10	ca $[Ns/m]$	10
g $[m/s^2]$	9.81	ct	0.1
I $[kgm^2]$	0.01	cq	0.5
l0 $[m]$	1	d $[m]$	1

TABLE I
USED PARAMETERS IN THE SIMULATION

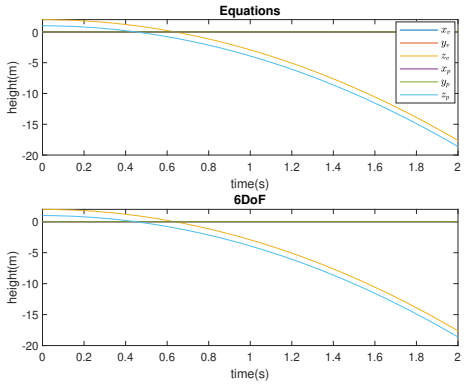


Fig. 2. System response with no input

To see if the dampening is functioning well the c_a term is now included. This gives the graph shown in figure 4. This shows that as time progresses that an equilibrium is reached in the z-direction and eventually the velocities go to 0.

The mass spring damper subsystem is now verified together with the rotational part. This is achieved by considering each axis separately and seeing if it acts correctly. First for every axis a negative moment is applied around this axis in BFF. So first a negative moment is put around the Y-axis. This should cause the Euler angle around y, the height of the UAV, the height of the payload and the value of x_p and x_v to start decreasing as the input force stays the same. This is the same behaviour that can be observed in figure 5. The top left shows the result from the equations of motion and the top right shows the 6DoF implementation. The bottom graph is the two graphs plotted on top of each other such that it can be seen that the values are the same. This is also checked with a positive moment and then for the Y and Z axis. This is shown in appendix A.

IV. Controller

A Linear Quadratic Regulator (LQR) controller was chosen, because it is widely used due to it finding the optimum control

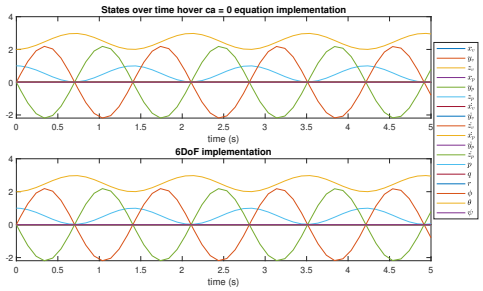


Fig. 3. System response without dampening

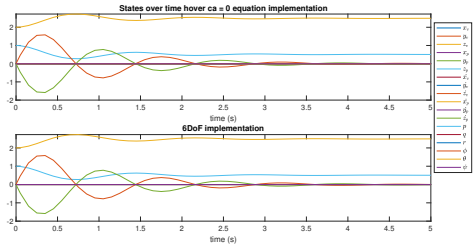


Fig. 4. System response with dampening

for linear systems. To design such a controller, it is needed that equations 3 and 10 are in the form of equation 27 and 28.

$$\dot{x} = \mathbf{A}x + \mathbf{B}u \tag{27}$$

$$y = \mathbf{C}x + Du \tag{28}$$

A. Linearization

The derived system is still non-linear while the LQR controller is applicable to linear systems. However, it is still possible to apply this control theory to a non-linear system by linearizing it around a specific point. This changes the state space equations to:

$$\dot{\mathbf{\Delta x}} = \mathbf{A\Delta x} + \mathbf{B\Delta u} \tag{29}$$

$$\Delta y = \mathbf{C\Delta x} + D\Delta u \tag{30}$$

[6] The linearization will be done around the equilibrium position for which equations were derived in section III.

$$\mathbf{\Delta x} = \begin{bmatrix} x_1 - x_1^* \\ \vdots \\ x_n - x_n^* \end{bmatrix} \tag{31}$$

Here the $\mathbf{x}^*$ is the equilibrium point. The same holds for the input u, where u^* is the equilibrium for hovering resulting in equation 32.

$$\mathbf{\Delta u} = \begin{bmatrix} u_1 - u_1^* \\ \vdots \\ u_n - u_n^* \end{bmatrix} \tag{32}$$

These new coordinates are the variations away from the equilibrium point. These can be seen as the new states, control inputs, and control outputs. To find the Jacobian matrix, a

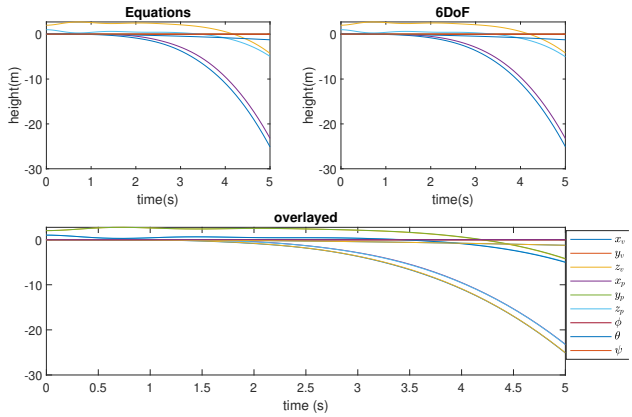


Fig. 5. Negative moment around Y-axis

partial derivative is taken over the different equations of motion with respect to all the states. This can be written down as in equation 33. Matrix **B** can be derived by taking the partial derivative of **f** with respect to the input. The function f stands for the derivative of the states so $f_1 = \dot{x}_1$ until $f_n = \dot{x}_n$ where n is the number of states

$$A = \frac{\partial \mathbf{f}}{\partial \mathbf{x}}, B = \frac{\partial \mathbf{f}}{\partial u} \tag{33}$$

[6]

To linearize the model, the next step is to choose the states. The chosen states are:

$\mathbf{x} = [x_v, y_v, z_v, x_p, y_p, z_p, \dot{x}_v, \dot{y}_v, \dot{z}_v, \dot{x}_p, \dot{y}_p, \dot{z}_p, p, q, r, \phi, \theta, \psi]$

The state function **f** results in the following:

$$\dot{x_1} = \dot{x_v} \tag{34}$$

$$\dot{x_2} = \dot{y_v} \tag{35}$$

$$\dot{x_3} = \dot{z_v} \tag{36}$$

$$\dot{x_4} = \dot{x_p} \tag{37}$$

$$\dot{x_5} = \dot{y_p} \tag{38}$$

$$\dot{x_6} = \dot{z_p} \tag{39}$$

$$\dot{x_7} = \frac{1}{m_v}((s\phi s\psi + c\phi c\psi s\theta)T_\Sigma + k_a(l - l_0)\hat{e_x} + c_a v_r \hat{e_x}) \tag{40}$$

$$\dot{x_8} = \frac{1}{m_v}((c\phi s\psi s\theta - c\psi s\phi)T_\Sigma + k_a(l - l_0)\hat{e_y} + c_a v_r \hat{e_y}) \tag{41}$$

$$\dot{x_9} = -g + \frac{1}{m_v}((c\phi c\theta)T_\Sigma + k_a(l - l_0)\hat{e_z} + c_a v_r \hat{e_z}) \tag{42}$$

$$\dot{x_{10}} = \frac{1}{m_p}(-k_a(l - l_0)\hat{e_x} - c_a v_r \hat{e_x}) \tag{43}$$

$$\dot{x_{11}}\frac{1}{m_p} = (-k_a(l - l_0)\hat{e_y} - c_a v_r \hat{e_y}) \tag{44}$$

$$\dot{x_{12}} = -g + \frac{1}{m_p}(-k_a(l - l_0)\hat{e_z} - c_a v_r \hat{e_z})) \tag{45}$$

$$\dot{x_{13}} = \frac{U_2 + (I_{yy} - I_{zz})qr}{I_{xx}} \tag{46}$$

$$\dot{x_{14}} = \frac{U_3 + (I_{zz} - I_{xx})pr}{I_{yy}} \tag{47}$$

$$\dot{x_{15}} = \frac{U_4 + (I_{xx} - I_{yy})pq}{I_{zz}} \tag{48}$$

$$\dot{x_{16}} = p + rc\psi t\theta + qs\psi s\theta \tag{49}$$

$$\dot{x_{17}} = qc\psi - rs\psi \tag{50}$$

$$\dot{x_{18}} = r\frac{c\psi}{c\theta} + q\frac{s\psi}{\theta} \tag{51}$$

For the linearization, the equilibrium point $\mathbf{x}* = [0, 0, 2.5, 0, 0, 0.5, \mathbf{0_{1x12}}]$ and $\mathbf{u}* = [(m_v + m_p)g, 0, 0, 0]$ the result of this is shown in appendix F1. This was calculated using the script in appendix D. For the validation, the function linmod is used which can linearize a Simulink system around a specified equilibrium point. When the theoretically derived equations are correct the A and B matrix results should be the same. This is the case as can be seen in appendix section F1 and F2.

B. Controllability

To see if the system is controllable, the controllability matrix is calculated which is specified as equation 52. The system is then controllable when the rank of matrix C is equal to the number of states.

$$C = \begin{bmatrix} A & A^2 ... & A^n B \end{bmatrix} \tag{52}$$

[7] This was then calculated using Matlab with the code from appendix E. This gave that the system was full rank and thus fully controllable.

C. LQR synthesis

The goal of an LQR controller is to find the optimal control by minimizing J in equation 53. [8]

$$J = \int_0^\infty (x_v^T Q x_v + u^T R u) dt \tag{53}$$

Here, Q is a nxn matrix where n is the number of states and R is an mxm matrix where m is the number of inputs. Here Q is chosen to be an nxn identity matrix. In figure 6 the implementation of the controller is shown. Here a reference state can be given as input to the system.

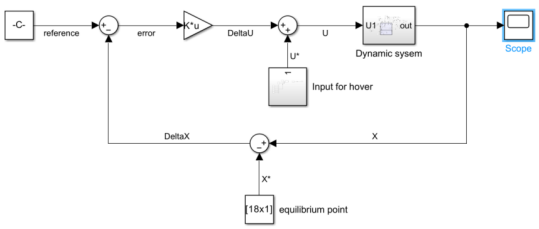


Fig. 6. Controller implementation in Simulink

D. Position control

When implementing the controller as in figure 6, it has a large steady-state error when moved away from the equilibrium position. To solve the steady-state issue an integral feedback is used. In this problem, the payload must stay at a certain position or track a trajectory. The implementation works by introducing three extra states in the controller which will be the integral of the position of the payload. This can be described by equations 54. [9]

$$\mathbf{v} = \int \begin{bmatrix} x_p \\ y_p \\ z_p \end{bmatrix} dt \tag{54}$$

By augmenting the state vector with this, the new state space model becomes equation 55. Here $G = [0_{3x3} I_{3x3}]$ which ensures that only the payload position is integrated.

$$\begin{bmatrix} \dot{\mathbf{x}} \\ \dot{\mathbf{v}} \end{bmatrix} = \begin{bmatrix} A & 0_{18x3} \\ G & 0_{3x15} \end{bmatrix} \begin{bmatrix} \mathbf{x} \\ \mathbf{v} \end{bmatrix} + \begin{bmatrix} B \\ 0_{3x4} \end{bmatrix} \mathbf{u} \tag{55}$$

The control input is also changed to:

$$\mathbf{u} = -K \begin{bmatrix} \mathbf{x} - u^* \\ \mathbf{v} - \mathbf{v_{ref}} \end{bmatrix} \tag{56}$$

$\mathbf{v_{ref}}$ is the desired position of the payload, therefore the control input to the system is changed to $u_{applied} = \mathbf{u} + u^*$. In figure 7 the implementation is shown in Simulink.

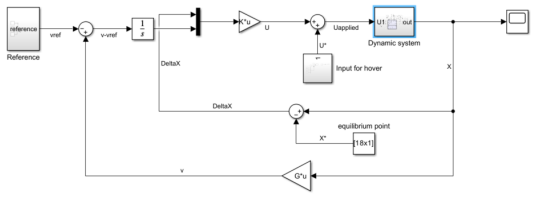


Fig. 7. Integral LQR controller in Simulink

V. Results

To see how the controller performs, tests are done. First, the LQR controller will be tested for its ability to control it around a hover position. Second, the improved controller with position control is tested by flying it to a certain position in space. At last, it is tested if the controller is able to let the package trace a circular trajectory.

A. LQR

For the first test, the reference is set at $\mathbf{0}$ and there are no disturbances present. The LQR controller was designed with $Q = I_{18x18}$ and $R = I_{4x4}$. In figure 8 the response of the system is shown after about 5 seconds it has reached its equilibrium position and is stable here.

To check whether the controller is robust, disturbances are introduced in the form of Gaussian noise. These are included as forces around the payload in x, y and z directions. As can be seen in figure 9 It is able to keep the x and y within 0.25[m] of the origin. But it is also able to control the z of the payload and of the UAV within a tolerance of 0.25[m]

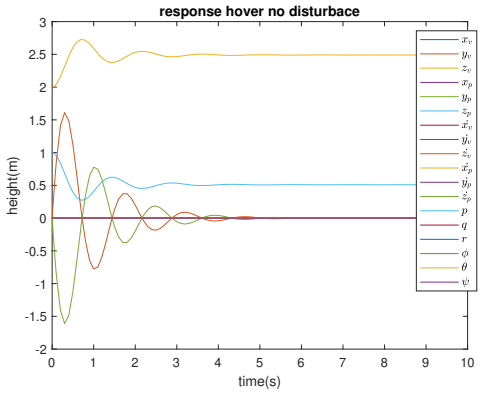


Fig. 8. Controller performance in hover no disturbance

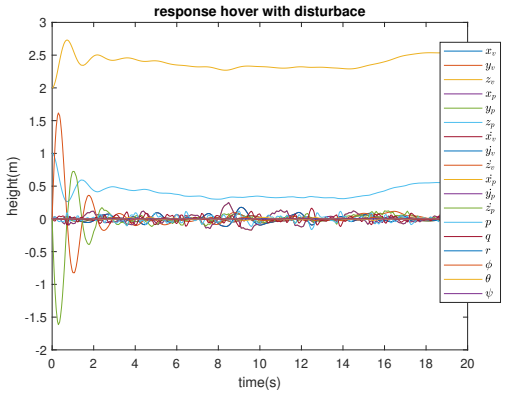


Fig. 9. Controller performance in hover with disturbance

around its equilibrium position. What also becomes evident is that in both cases the controller is not able to compensate for the oscillations caused by the initial conditions.

B. Position control

For this simulation, the initial length of the rope l_0 is increased to 10 [m]. The reference point of the payload is set at [3,5,2]. For the LQR controller, a Q was used of $Q = 100 \begin{bmatrix} I_{18x18} & 0_{18x3} \\ 0_{3x18} & 100I_{3x3} \end{bmatrix}$ and a $R = I_{4x4}$. This resulted in figure 10. What can be seen is that the payload reaches the set point in about eight seconds in which the steady-state error is zero. Here, it can also be seen that the oscillations caused by the mass spring damper system are not controlled well in the beginning.

C. Tracking a circular trajectory

To see if the designed controller is able to make the payload trace a circular trajectory with a radius of 5[m], the reference is changed to $z_{ref} = 5[m]$ and the $x_{ref} = 5 * sin(t)[m]$ and $y_{ref} = 5 * cos(t)[m]$. In figure 11, the result is shown. In this figure, the cross corresponds to the UAV at a certain time and the o to the position of the payload in time. The corresponding times are connected via a line which is the cable. As can

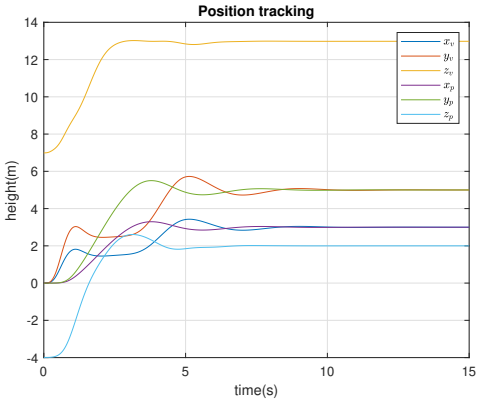


Fig. 10. System response when flying to setpoint [3,5,2]

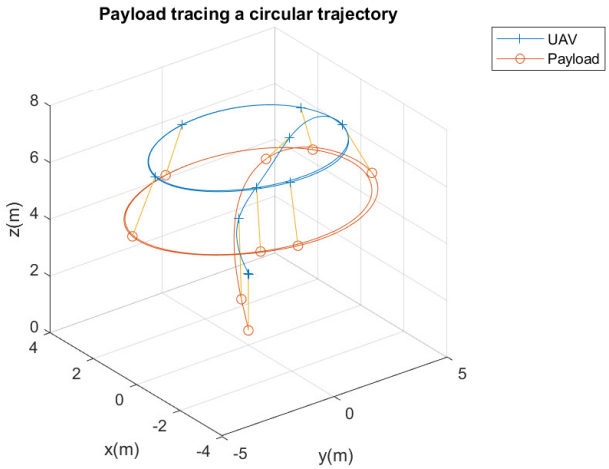


Fig. 11. Payload tracking a circular reference

be seen, it is able to follow the circle quite well after it has stabilized.

VI. Discussion

The designed controller is able to converge to the desired position quite well. The downside is that it does not take care of the initial oscillations of the system after these have damped out, then it has no problem controlling the location of the payload. The reason it is not able to take care of these oscillations is probably due to it being linearized around a certain position and that it does not include these dynamics. To solve this issue more research has to be done into for example non-linear controllers or gain scheduling LQR.

VII. Conclusion

The paper presented a detailed model of an UAV carrying an elastic-slung load. For this system, the equilibrium for hovering was determined as well as a linearized state space form. On the basis of this linearized state space, a LQR controller was designed to keep the UAV at its hover position. With the LQR controller a steady state error was experienced,

this was solved by introducing an integral feedback on the position of the payload. This was then tested by letting it hover at a point outside the equilibrium and tracing a circular trajectory which was succesfull. This paper was based on the following research question: "How does the elasticity in the cable affect the system behaviour of a quadrotor UAV". It can now be said that taking the elasticity into account can lead to new interesting trajectories by the controller being able to utilize the elasticity for getting to the desired position quickly. For future work, it would be interesting to look into the possibility of using trajectory tracking by either a gain-scheduling LQR controller or a non-linear controller and exploring a wider range of different types of cable.

REFERENCES

[1] Kamil Us, Altan Cevher, Mert Sever, and Ahmet Kirli. On the effect of slung load on quadrotor performance. *Procedia Computer Science*, 158:346–354, 01 2019.

[2] Ying Feng, Camille Alain Rabbath, and Chun-Yi Su. *Modeling of a Micro UAV with Slung Payload*, pages 1257–1272. Springer Netherlands, Dordrecht, 2015.

[3] Alexander Cicchino. *Three Formulations of Sling Load Dynamics for UAV Motion Planning and Control*. McGill University, 2018.

[4] Peter Corke Robert Mahony, Vijay Kumar. Multirotor aerial vehicles modeling, estimation and control of quadrotor. *IEEE Robotics and automation magazine*, pages 20–32, 2012.

[5] Michael Triantafyllou. Maneuvering and control of surface and underwater vehicles (13.49), 2004.

[6] University of Toronto. Ece311 - dynamic systems and control linearization of nonlinear systems, 2007.

[7] Erfan Nozari. Lecture 3: Stability, controllability observability, 2010.

[8] Faraz Ahmad, Pushpendra Kumar, Anamika Bhandari, and Pravin P. Patil. Simulation of the quadcopter dynamics with lqr based control. *Materials Today: Proceedings*, 24:326–332, 2020. International Conference on Advances in Materials and Manufacturing Applications, IConAMMA 2018, 16th -18th August, 2018, India.

[9] R. Praveen Jain. *Transportation of cable suspended load using unmanned aerial vehicles: A real-time model predictive control approach*. PhD thesis, 08 2015.

APPENDIX

A. Validation of the equations of motion

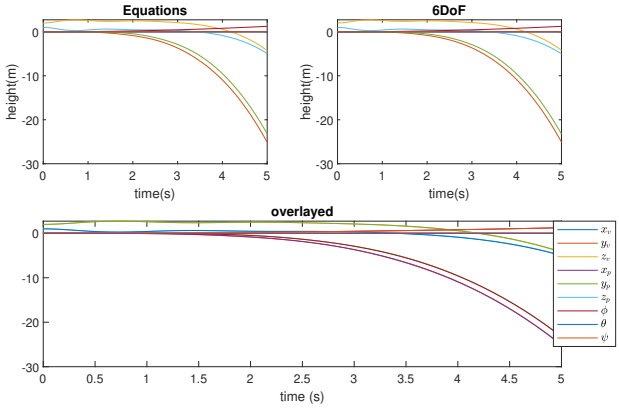


Fig. 12. Negative moment around X-axis

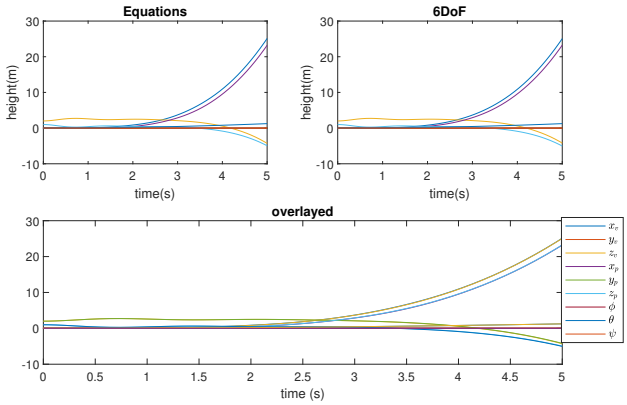


Fig. 13. Positive moment around Y-axis

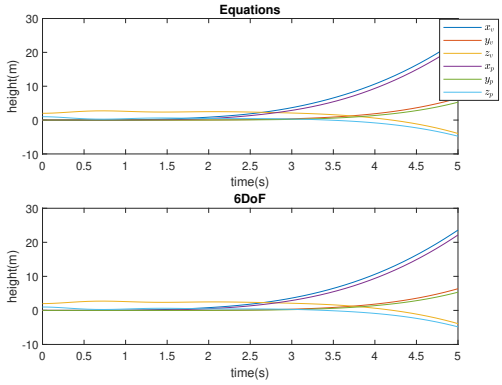


Fig. 16. negative moment around ZX-axis

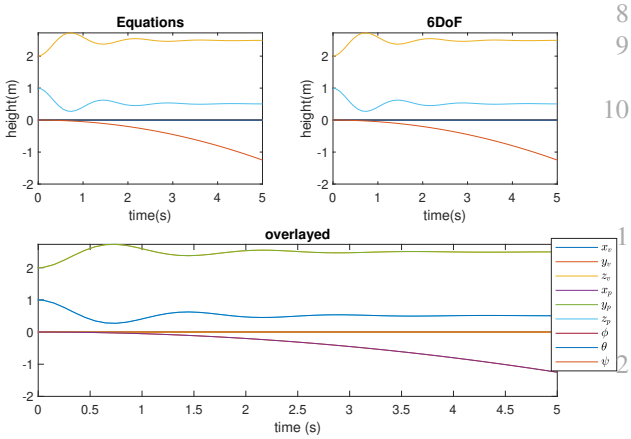


Fig. 14. Negative moment around Z-axis

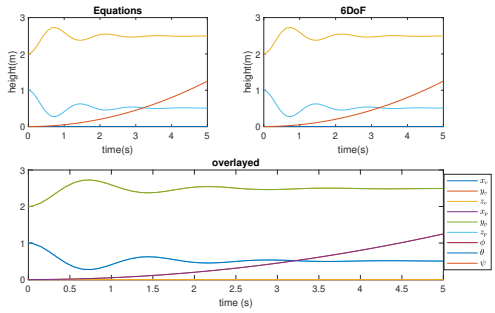


Fig. 15. Positive moment around Z-axis

B. Rotation matrix

```
%Rotation matrix
syms theta phi psi
Rz = [cos(phi) sin(phi) 0; -sin(phi) cos(
    phi) 0; 0 0 1];
Ry = [cos(theta) 0 -sin(theta); 0 1 0 ;
    sin(theta) 0 cos(theta)];
Rx = [1 0 0; 0 cos(psi) sin(psi); 0 -sin(
    psi) cos(psi)];
Rb = Rx*Ry*Rz;
Re = transpose(Rb);
```

C. Find equilibrium

```
%matlab visualization
model = 'ModelFindSteadyState';  %
    Vervang 'naam_van_je_simulink_model'
    door de werkelijke naam van je
    Simulink-model

load_system(model);

% Simuleer het model
simOut = sim(model, 'StopTime', '60', '
    SrcWorkspace', 'current');

States = transformTo2DArray(simOut.States
    .Data);
EqPoint = States(:,length(States))
```

D. Finding the A and B matrices

```
syms x_v y_v z_v x_p y_p z_p x_dot_v
    y_dot_v z_dot_v x_dot_p y_dot_p
    z_dot_p phi theta psi U1 U2 U3 U4 ca
    ka mv mp g Ixx Iyy Izz I l0 p q r
RR = [1 0 -sin(theta); 0 cos(phi) sin(phi
    )*cos(theta);0 -sin(phi) cos(phi)*cos(
    theta)]%[cos(theta) 0 -cos(phi)*sin(
    theta); 0 1 sin(phi); sin(theta) 0 cos
    (phi)*cos(theta)]
U = [U2; U3; U4];
pqr = [p;q;r];
Is = [Ixx 0 0; 0 Iyy 0; 0 0 Izz];
dotpqr = inv(Is)*(U- cross(pqr,(Is*pqr)))
Rinv = simplify(inv(RR))
dotphithetapsi = Rinv*pqr

vr = [(x_dot_p-x_dot_v); (y_dot_p-y_dot_v
    );(z_dot_p-z_dot_v)]
l = sqrt((x_p-x_v)^2+(y_p-y_v)^2+(z_p-z_v
    )^2)
ek = 1/l*[x_p-x_v;y_p-y_v;z_p-z_v]
vrk = dot(vr,ek)*ek

syms theta phi psi
Rz = [cos(psi) sin(psi) 0; -sin(psi) cos(
    psi) 0; 0 0 1]
Ry = [cos(theta) 0 -sin(theta); 0 1 0 ;
    sin(theta) 0 cos(theta)]
Rx = [1 0 0; 0 cos(phi) sin(phi); 0 -sin(
    phi) cos(phi)]
Rb = Rx*Ry*Rz
Re = simplify(inv(Rb))
vU1 = [0;0;U1];
FU1 = Re*vU1

eq1 = x_dot_v; % Eq for x_dot_v
eq2 = y_dot_v; % Eq for y_dot_v
eq3 = z_dot_v; % Eq for z_dot_v
eq4 = x_dot_p; % Eq for x_dot_p
eq5 = y_dot_p; % Eq for y_dot_p
eq6 = z_dot_p; % Eq for z_dot_p
eq7 = 1/mv*(FU1(1)+ka*(x_p-x_v)-ka*l0*(
    x_p-x_v)/(sqrt((x_p-x_v)^2+(y_p-y_v)
    ^2+(z_p-z_v)^2))+ca*vrk(1)) %Eq for
    xdotdotp
eq8 = 1/mv*(FU1(2)+ka*y_p-ka*y_v- ka*l0*(
    y_p-y_v)/(sqrt((x_p-x_v)^2+(y_p-y_v)
    ^2+(z_p-z_v)^2))+ca*vrk(2)) %Eq for
    ydotdotp
```

```
eq9 = -g + 1/mv*(FU1(3)+ka*z_p-ka*z_v+ca*vrk(3) - ka*l0*(z_p-z_v)/(sqrt((x_p-x_v)^2+(y_p-y_v)^2+(z_p-z_v)^2))) %Eq for zdotdotp
eq10 = -1/mp*(ka*(x_p-x_v)-ka*l0*(x_p-x_v)/(sqrt((x_p-x_v)^2+(y_p-y_v)^2+(z_p-z_v)^2))+ca*vrk(1)) %Eq for xdotdotv
eq11 =-1/mp*(ka*y_p-ka*y_v- ka*l0*(y_p-y_v)/(sqrt((x_p-x_v)^2+(y_p-y_v)^2+(z_p-z_v)^2))+ca*vrk(2)) %Eq for ydotdotv
eq12 =-g + 1/mp*(-(ka*z_p-ka*z_v+ca*vrk(3) + - ka*l0*(z_p-z_v)/(sqrt((x_p-x_v)^2+(y_p-y_v)^2+(z_p-z_v)^2)))) %Eq for zdotdotv
eq13 = dotpqr(1) % Eq for pdot
eq14 = dotpqr(2) % Eq for qdot
eq15 = dotpqr(3) % Eq for rdot
eq16 = dotphithetapsi(1) % Eq for phidot
eq17 = dotphithetapsi(2) % Eq for thetadot
eq18 = dotphithetapsi(3) % Eq for psidot
xdot = [eq1 eq2 eq3 eq4 eq5 eq6 eq7 eq8 eq9 eq10 eq11 eq12 eq13 eq14 eq15 eq16 eq17 eq18]
x = [x_v y_v z_v x_p y_p z_p x_dot_v y_dot_v z_dot_v x_dot_p y_dot_p z_dot_p p q r phi theta psi]
u = [U1 U2 U3 U4]
z0 = 2;

A = jacobian(xdot,x)
A = subs(A,[Ixx,Iyy,Izz,U1,ca,ka,l0,mp,mv,p,phi,psi,q,r,theta,x_p,x_v,y_p,y_v,z_p,z_v, x_dot_p,x_dot_v,y_dot_v,y_dot_p,z_dot_p,z_dot_v],[0.01,0.01,0.01,(Mv+Mp)*9.81,10,100,1,Mp,Mv,0,0,0,0,0,0,0,0,0,0,0.5,2.5,0,0,0,0,0])

var = vpa(A)
A = double(A);
B = jacobian(xdot,u)
B = subs(B,[Ixx,Iyy,Izz,mp,mv,phi,psi,theta],[0.01,0.01,0.01,Mp,Mv,0,0,0])
B = double(B);
C = ctrb(A,B)
rank(C)
```

E. Find controllable states

```
C = ctrb(a,b)
rank(C)
% Determine the controllable states
% Determine the controllable states for each input
controllable_statestill1 = false(1, 14);
controllable_statestill2 = false(1, 14);
controllable_statestill3 = false(1, 14);
controllable_statestill4 = false(1, 14);
controllable_statestill5 = false(1, 14);
controllable_statestill6 = false(1, 14);
controllable_statestill7 = false(1, 14);
controllable_statestill8 = false(1, 14);
controllable_statestill9 = false(1, 14);
controllable_statestill10 = false(1, 14);
controllable_statestill11 = false(1, 14);
controllable_statestill12 = false(1, 14);
controllable_statestill13 = false(1, 14);
controllable_statestill14 = false(1, 14);

for i = 1:18
    controllable_statestill1(i) = rank(c(1:i, :)) == 1;
    controllable_statestill2(i) = rank(C(1:i, :)) == 2;
    controllable_statestill3(i) = rank(C(1:i, :)) == 3;
    controllable_statestill4(i) = rank(C(1:i, :)) == 4;
    controllable_statestill5(i) = rank(C(1:i, :)) == 5;
    controllable_statestill6(i) = rank(C(1:i, :)) == 6;
    controllable_statestill7(i) = rank(C(1:i, :)) == 7;
    controllable_statestill8(i) = rank(C(1:i, :)) == 8;
    controllable_statestill9(i) = rank(C(1:i, :)) == 9;
    controllable_statestill10(i) = rank(C(1:i, :)) == 10;
    controllable_statestill11(i) = rank(C(1:i, :)) == 11;
    controllable_statestill12(i) = rank(C(1:i, :)) == 12;
    controllable_statestill13(i) = rank(C(1:i, :)) == 13;
    controllable_statestill14(i) = rank(C(1:i, :)) == 14;
end

% Display the results
disp('States controllable by input 1:');
disp(controllable_statestill1);
disp('States controllable by input 2:');
disp(controllable_statestill2);
disp('States controllable by input 3:');
disp(controllable_statestill3);
disp('States controllable by input 4:');
disp(controllable_statestill4);
disp('States controllable by input 5:');
```

```
disp(controllable_statestill5);
disp('States controllable by input 6:');
disp(controllable_statestill6);
disp('States controllable by input 7:');
disp(controllable_statestill7);
disp('States controllable by input 8:');
disp(controllable_statestill8);
disp('States controllable by input 9:');
disp(controllable_statestill9);
disp('States controllable by input 10:');
disp(controllable_statestill10);
disp('States controllable by input 11:');
disp(controllable_statestill11);
disp('States controllable by input 12:');
disp(controllable_statestill12);
disp('States controllable by input 13:');
disp(controllable_statestill13);
disp('States controllable by input 14:');
disp(controllable_statestill14);
```

$$B = \begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ \frac{1}{10} & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 100 & 0 & 0 \\ 0 & 0 & 100 & 0 \\ 0 & 0 & 0 & 100 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix}$$

F. Linearization

1) Result from equations

$$A = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ -5 & 0 & 0 & 5 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \frac{981}{50} & 0 \\ 0 & -5 & 0 & 0 & 5 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -\frac{981}{50} & 0 & 0 \\ 0 & 0 & -10 & 0 & 0 & 10 & 0 & 0 & -1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 5 & 0 & 0 & -5 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 5 & 0 & 0 & -5 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 10 & 0 & 0 & -10 & 0 & 0 & 1 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \end{pmatrix}$$

$$B = \begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ \frac{1}{10} & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 100 & 0 & 0 \\ 0 & 0 & 100 & 0 \\ 0 & 0 & 0 & 100 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix}$$

2) Result form linmod

$$A = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ -5 & 0 & 0 & 5 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \frac{981}{50} & 0 \\ 0 & -5 & 0 & 0 & 5 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -\frac{981}{50} & 0 & 0 \\ 0 & 0 & -10 & 0 & 0 & 10 & 0 & 0 & -1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 5 & 0 & 0 & -5 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 5 & 0 & 0 & -5 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 10 & 0 & 0 & -10 & 0 & 0 & 1 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \end{pmatrix}$$