Controls 2 Drone Project - part 1

Submitted by:

Aylon Feraru 325214492
Yair Hulin 316595628

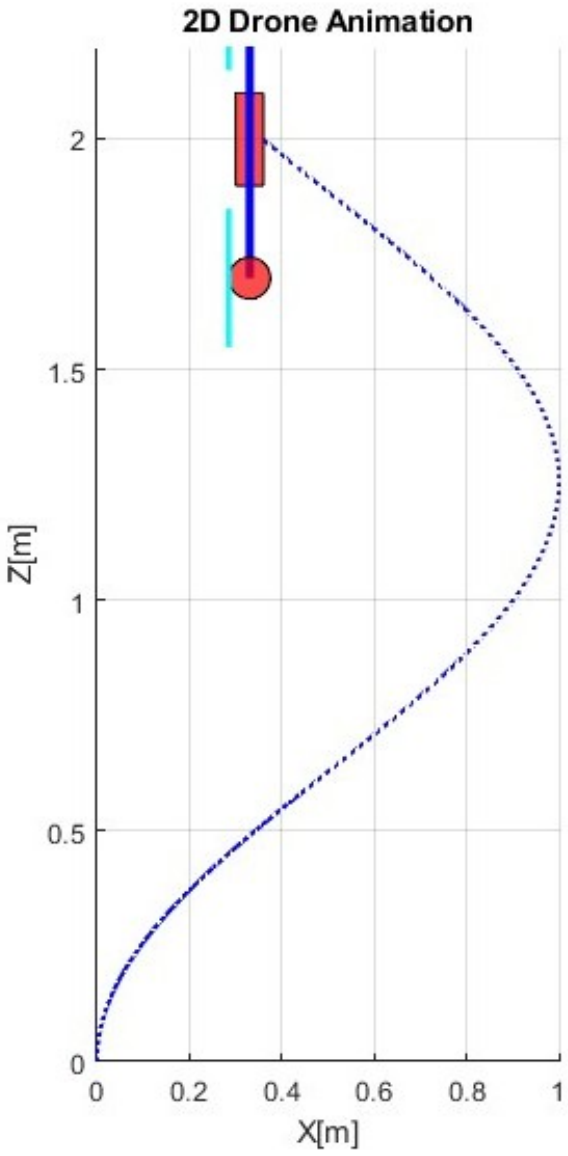
note that $\psi = 10$

Question 1.1:

- since the drone needs to rise linearly in time to $2[m]$, and it has a 10 second duration to do so, we'll set $z = 0.2 \cdot t$.
- since x needs to change sinusoidally between 0 and 1, we'll select the offset to be 0.5. additionally, we want the frequency to be $0.5[rad/s]$ so we'll define $\omega_x = 2\pi f_x = 0.5$. Since the simulation is $10[s]$ we're set to complete $5[rad]$ in that time, which is slightly less than a full cycle.(and we have.)
- The pitch should start at -90°, and end at 90°, so we'll note that taking $90\cos(t \cdot 2\pi f_{pitch} + \phi)$ with initial phase of π get us the correct start ($90\cos(\pi) = -90$) to end at 90° we'll require $\cos(20\pi f_{pitch} + \pi) = 1$ and $f = 0.05[Hz]$ fits that condition. ($90\cos(2\pi) = 90$)

```
%% define the motion track

sim_duration = 10;

sim_framerate = 0.03;

f_s = 0.25;

f_x = 0.5/(2*pi);

f_pitch = 0.05;

% % --- Edit this code (1.1)
t = 0:sim_framerate:sim_duration;
x = 0.5 + 0.5*sin((2*pi*f_x*t)-pi/2);
z = 0.2*t;
pitch = 90*cos(2*pi*f_pitch*t-pi);
% % ------------------
```

and the output is:



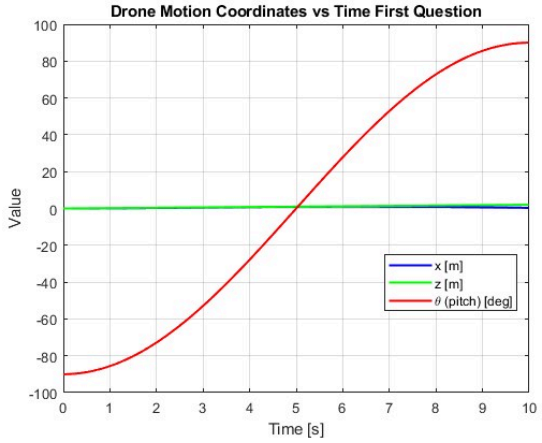
we can see that the drone behaves as required.

Question 1.2 - Graph of drone parameters

we can see that the pitch behaves exactly as expected - going from -90 to 90 degrees. z rises very slowly from 0 to 2 which is hard to see at this scale, and x indeed oscillates like a sine wave between 0 and 1.

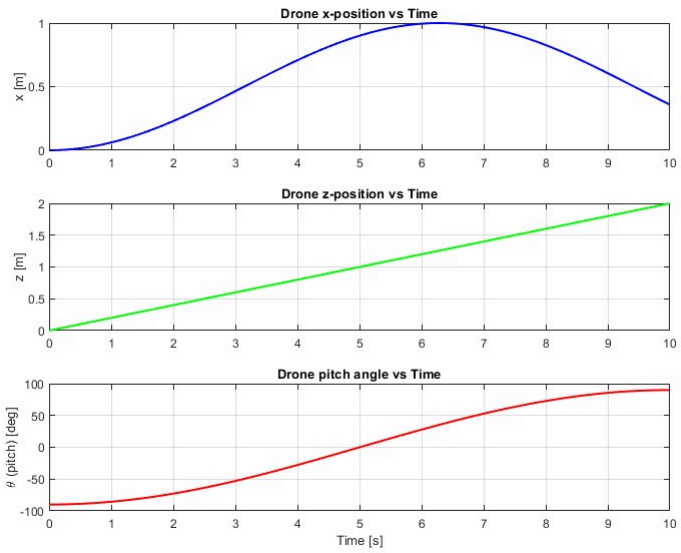
code:

```
figure;
plot(t, x, 'b', 'LineWidth', 1.5); hold on;
plot(t, z, 'g', 'LineWidth', 1.5);
plot(t, pitch, 'r', 'LineWidth', 1.5);
hold off;

xlabel('Time [s]');
ylabel('Value');
title('Drone Motion Coordinates vs Time');
legend({'x [m]', 'z [m]', '\theta (pitch) [deg]'}, 'Location', 'best');
grid on;
```



However, if instead we wanted to look at the three plots individually, I've added a graph for that as well in this report and code

```
figure;
subplot(3,1,1)
plot(t, x, 'b', 'LineWidth', 1.5)
ylabel('x [m]')
title('Drone x-position vs Time')
grid on

subplot(3,1,2)
plot(t, z, 'g', 'LineWidth', 1.5)
ylabel('z [m]')
title('Drone z-position vs Time')
grid on

subplot(3,1,3)
plot(t, pitch, 'r', 'LineWidth', 1.5)
xlabel('Time [s]')
ylabel('\theta (pitch) [deg]')
title('Drone pitch angle vs Time')
grid on
```



Question 2.1 - nonlinear system model

given the following equations:

$$\begin{aligned} m\ddot{x} &= -(F_1 + F_2)\sin(\theta) \\ m\ddot{z} &= (F_1 + F_2)\cos(\theta) - mg \\ J\ddot{\theta} &= L(F_1 - F_2) \end{aligned}$$

and

$$U = \begin{pmatrix} F_1 + F_2 \\ F_1 - F_2 \end{pmatrix}$$

we can rewrite the original equation as:

$$\begin{aligned} m\ddot{x} &= -U_1 \sin(\theta) \\ m\ddot{z} &= U_1 \cos(\theta) - mg \\ J\ddot{\theta} &= LU_2 \end{aligned}$$

so our state vector X's derivative can be expressed as:

$$\underline{\dot{X}} = \frac{d}{dt}\begin{pmatrix} x \\ \dot{x} \\ z \\ \dot{z} \\ \theta \\ \dot{\theta} \end{pmatrix} = \begin{pmatrix} \dot{x} \\ -\frac{U_1}{m}\sin(\theta) \\ \dot{z} \\ \frac{U_1}{m}\cos(\theta) - g \\ \dot{\theta} \\ \frac{L}{J}U_2 \end{pmatrix} = f(\underline{X}, \underline{U})$$

using both the definitions of the state variables, and the provided equations. Unlike X, we can express the output Y linearly, so we'd prefer to do so:

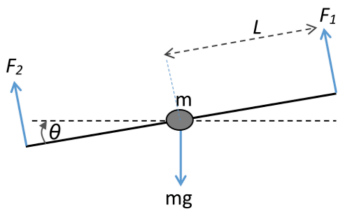
$$\underline{Y} = \begin{pmatrix} x \\ z \\ \theta \end{pmatrix} = g(\underline{X}, \underline{U}) = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \end{pmatrix} \begin{pmatrix} x \\ \dot{x} \\ z \\ \dot{z} \\ \theta \\ \dot{\theta} \end{pmatrix}$$

Question 2.2 - equilibrium points

demand:

$$\dot{X} = \underline{0} = \begin{pmatrix} \dot{x} \\ -\frac{U_1}{m}\sin(\theta) \\ \dot{z} \\ \frac{U_1}{m}\cos(\theta) - g \\ \dot{\theta} \\ \frac{L}{J}U_2 \end{pmatrix}$$

from the demands we can already infer that $\dot{x}^*, \dot{z}^*, \dot{\theta}^*$ are all zero, which is good. $U_2^* = 0$ since J and L are nonzero physical constants, also makes sense: $U_2 = F_1 - F_2$ since we want the drone hovering somewhere in place, not to spin.(picture below)



that leaves us with two state variables to find.

$$\begin{cases} 0 = -\frac{U_1}{m}\sin(\theta) \\ mg = U_1\cos(\theta) \end{cases}$$

since our drone has mass, $U_1^* \neq 0$ which means $\theta = \pi n \quad n \in \mathbb{Z}$.

(for odd ns our drone either face-plants, or our propellers output negative thrust - which isn't very physical and also not in $\theta \in (-\pi, \pi)$, which was given by the question so we'll ignore them.)
To sum up:
$\theta = 2\pi n \implies \theta = 0 \implies U_1 = mg = 2[kg] \cdot 9.8[m/s^2] = 19.6[N]$
We'll note that x, z are free variables, which makes sense - we want to balance out the forces acting on the drone to get a hover, and none of them are dependent on space in our model.

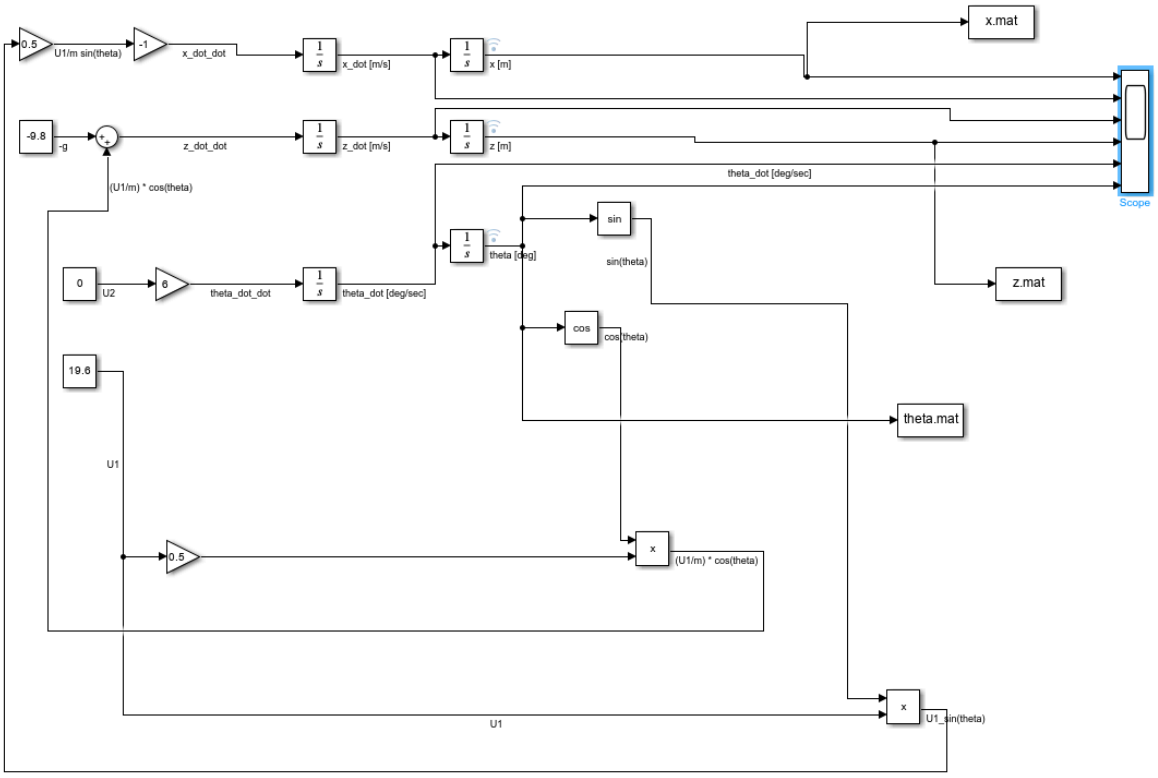
our equilibrium points then are:

$$\boxed{X^* = \begin{pmatrix} x \\ \dot{x} \\ z \\ \dot{z} \\ \theta \\ \dot{\theta} \end{pmatrix} = \begin{pmatrix} x_{free} \\ 0 \\ z_{free} \\ 0 \\ 0 \\ 0 \end{pmatrix}, \quad U^* = \begin{pmatrix} 19.6 \\ 0 \end{pmatrix}}$$

where points with subscript are free to be anything. and from now on $x^* = 2, z^* = 5$ as the questionnaire dictates.

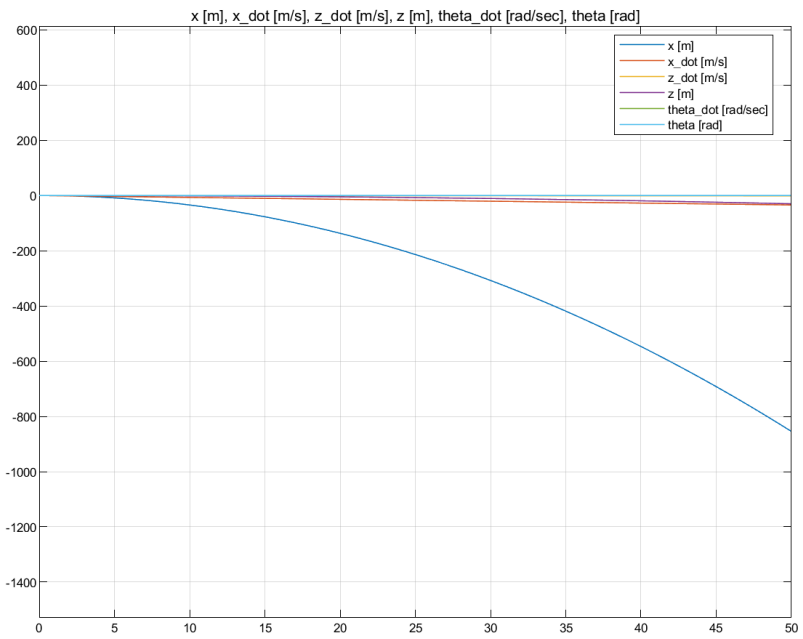
Question 2.3 + 2.4 - Simulink scheme of nonlinear model

we'll note that $L = 0.5 + 10^{-2} \cdot \psi = 0.5 + 10^{-2} \cdot 10 = 0.6$ and $J = 0.1$ from the given chart, so when we have $\frac{L}{J}U_2$ we need to substitute $6 \cdot U_2$ instead.



we can do a sanity check and input 0 on all initial conditions and the expected behavior emerges: a single constant line (everything overlaps). Since it's an equilibrium point that's also what we expect - a lack of change in state variables.

however, when we add a simple offset to the initial angle $\theta_0 = 4^\circ$ (converted to radians) the system diverges:

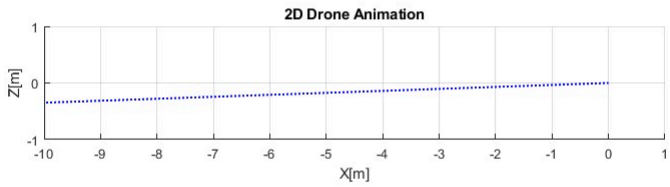


if we look at *how* it diverges, it makes physical sense - z keeps falling and x changes similarly since our angle means some of the forces that were previously counteracting mg are now redirected in the negative x-direction - we can see that the displacement curves are parabolic, and the velocity curves linear, as we would expect from the constant velocity. Note that $\dot{\theta}, \theta$ remain unchanged because in our dynamical model they're influenced by U_2 which is 0.

Question 2.5 - is the equilibrium point asymptotically stable?

No - the system diverges. An asymptotic equilibrium point fulfills $x(t) \to 0$ when $t \to \infty$, for every $x(0)$ - we can clearly see that this is not the case for our initial condition. (ran it further to 200 timestamps and it still didn't converge but diverged further, so unless the convergence time is extremely long, no - the point isn't asymptotically stable.)

Question 2.6 - drone trajectory graph:



The plot seems to make sense - we're comparing x to z and they both grow at a parabolic rate (their accelerations are constant), so the progression in the x-z plane is linear. We can see how x and z both grow in the negatives which aligns with the previous graph.

code for generating that plot:

```
close all
clearvars
clc

modelName = 'dynamical_system_question_2';
load_system(modelName);
simOut = sim(modelName);

load('x.mat');     x = ans;
load('z.mat');     z = ans;
load('theta.mat'); theta = ans;

t_sim      = x.Time;
x_sim      = x.Data;
z_sim      = z.Data;
theta_sim  = theta.Data;

save_vid = true;
drone_animation_2D(t_sim, x_sim, z_sim, theta_sim, save_vid);

figure;
plot(t_sim, x_sim, 'b', 'LineWidth', 1.5); hold on;
```

```
plot(t_sim, z_sim, 'g', 'LineWidth', 1.5);
plot(t_sim, theta_sim, 'r', 'LineWidth', 1.5);
hold off;

xlabel('Time [s]');
ylabel('Value');
title('Drone Motion Coordinates vs Time');
legend({'x [m]', 'z [m]', '\theta (pitch) [deg]'}, 'Location', 'best');
grid on;
```

3.1 Linearizing around a fixed point

we need to linearize around the desired fixed point, meaning:

$$X^* = \begin{pmatrix} x \\ \dot{x} \\ z \\ \dot{z} \\ \theta \\ \dot{\theta} \end{pmatrix} = \begin{pmatrix} 2 \\ 0 \\ 5 \\ 0 \\ 0 \\ 0 \end{pmatrix}, \quad U^* = \begin{pmatrix} 19.6 \\ 0 \end{pmatrix}$$

we'll recall and re-write more conveniently the state vector:

$$\dot{X} = \begin{pmatrix} \dot{x} \\ -\frac{U_1}{m}\sin(\theta) \\ \dot{z} \\ \frac{U_1}{m}\cos(\theta) - g \\ \dot{\theta} \\ \frac{L}{J}U_2 \end{pmatrix} = \begin{pmatrix} x_2 \\ -\frac{U_1}{m}\sin(x_5) \\ x_4 \\ \frac{U_1}{m}\cos(x_5) - g \\ x_6 \\ \frac{L}{J}U_2 \end{pmatrix} = f(\underline{X}, \underline{U})$$

$$A = \begin{pmatrix} \frac{\partial f_1}{\partial x_1} & \cdots & \frac{\partial f_1}{\partial x_6} \\ & \vdots & \\ \frac{\partial f_6}{\partial x_1} & \cdots & \frac{\partial f_6}{\partial x_6} \end{pmatrix} = \begin{pmatrix} 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & -\frac{U_1}{m}\cos(x_5) & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & \frac{U_1}{m}(-\sin(x_5)) & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}\Bigg|_{U^*, X^*} = \begin{pmatrix} 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & -9.8 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}$$

very sparse. similarly:

$$B = \frac{\partial \vec{f}}{\partial \vec{U}} = \begin{pmatrix} 0 & 0 \\ -\frac{1}{m}\sin(x_5) & 0 \\ 0 & 0 \\ \frac{1}{m}\cos(x_5) & 0 \\ 0 & 0 \\ 0 & \frac{L}{J} \end{pmatrix}\Bigg|_{X^*, U^*} = \begin{pmatrix} 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ \frac{1}{2} & 0 \\ 0 & 0 \\ 0 & 6 \end{pmatrix}$$

and since we can recall that

$$\underline{Y} = \begin{pmatrix} x \\ z \\ \theta \end{pmatrix} = g(\underline{X}, \underline{U}) = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \end{pmatrix} \begin{pmatrix} x \\ \dot{x} \\ z \\ \dot{z} \\ \theta \\ \dot{\theta} \end{pmatrix}$$

then

$$\begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \end{pmatrix} = C, \quad D = 0$$

Question 3.2 - analysis of linear approximation

let's whip up a script to test controllability and observability of the matrix:

```
% Linearized A matrix (evaluated at equilibrium)
A = [ 0   1     0    0     0   0;
      0   0     0    0  -9.8  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];

% Linearized B matrix with L/J = 6
B = [ 0  0;
      0  0;
      0  0;
    0.5  0;
      0  0;
      0  6];

% Output matrix measuring x, z, theta
C = [1 0 0 0 0 0;
     0 0 1 0 0 0;
     0 0 0 0 1 0];

% Check controllability
Co = ctrb(A, B);
rank_Co = rank(Co);
disp(['Controllability matrix rank: ', num2str(rank_Co)]);
if rank_Co == size(A,1)
    disp('System is controllable.');
else
    disp('System is not controllable.');
end

% Check observability
Ob = obsv(A, C);
rank_Ob = rank(Ob);
disp(['Observability matrix rank: ', num2str(rank_Ob)]);
if rank_Ob == size(A,1)
    disp('System is observable.');
else
    disp('System is not observable.');
end

```

and running it we get:

```
>> main
Controllability matrix rank: 6
System is controllable.
Observability matrix rank: 6
System is observable.
```

(note to self - the command: commandwindow; forces the output window open in case you minimize it away)

and because the system is controllable there are no unstable modes that state feedback can't move (since they don't exist), and similarly there are no unobservable unstable modes because all the modes are stable.
Meaning - the system is controllable, stabilizable, observable, and detectible.

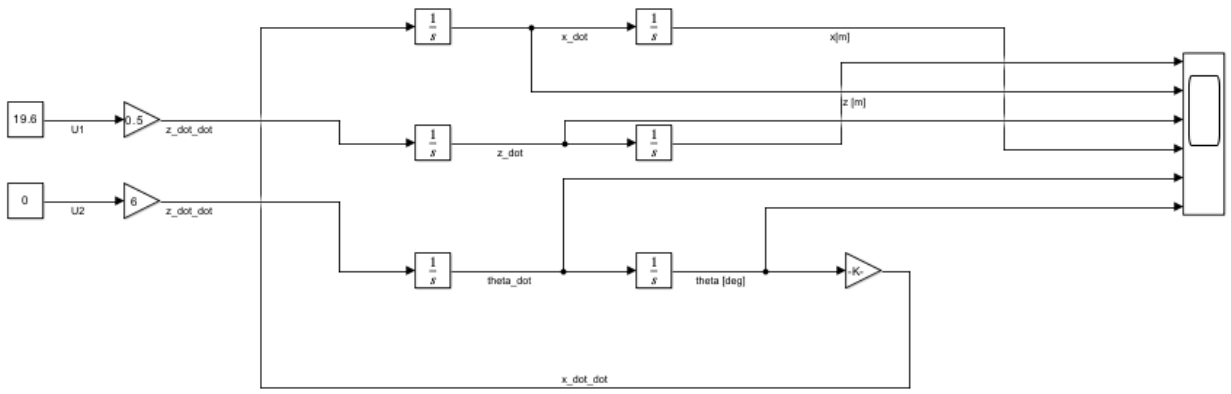
Question 3.3+3.4 - Simulink of linearized system

$$\dot{X} = \frac{d}{dx}\begin{pmatrix} x \\ \dot{x} \\ z \\ \dot{z} \\ \theta \\ \dot{\theta} \end{pmatrix} = \begin{pmatrix} 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & -9.8 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix} \begin{pmatrix} x \\ \dot{x} \\ z \\ \dot{z} \\ \theta \\ \dot{\theta} \end{pmatrix} + \begin{pmatrix} 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ \frac{1}{2} & 0 \\ 0 & 0 \\ 0 & 6 \end{pmatrix} \begin{pmatrix} U_1 \\ U_2 \end{pmatrix}$$

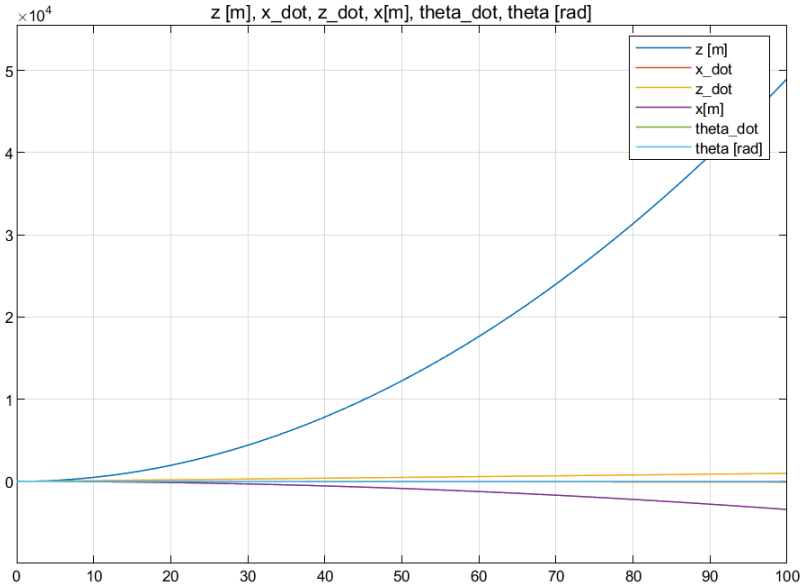
and

$$\underline{Y} = \begin{pmatrix} x \\ z \\ \theta \end{pmatrix}$$

so wiring the equations we get:

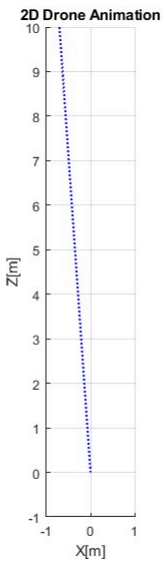


we'll note that the model is a lot simpler than we previously had, and the reasons for that are that by linearizing a lot of elements cancelled out - and half of our equations in the matrix are just identities.
running the same simulation that we had previously:



we can again see divergence, however, the direction of the divergence is different - we'll note that $z \to \infty$ now whereas previously it plummeted. The reason for that is that even though there's still tilt and not all the motor's force is directed upwards, our linear model does not take into account gravity g (got removed when we took derivatives of f) - so our drone's essentially acting in a vacuum! We'll also note that x and the other state variables are diverging in the same direction, we can see that for them the linearization is a reasonably good approximation in the short term, before our system strays too far away from the equilibrium point.

Question 3.5 - drone simulator of linear system



we can see that as expected from the scope diagram, x diverges negatively and z positively. This lines up with the scope and our physical intuition.

Question 4.1 - finding transfer functions after linearization

we'll recall that

$$\dot{X} = \begin{pmatrix} 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & -19.6 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix} \begin{pmatrix} x \\ \dot{x} \\ z \\ \dot{z} \\ \theta \\ \dot{\theta} \end{pmatrix} + \begin{pmatrix} 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ \frac{1}{2} & 0 \\ 0 & 0 \\ 0 & 6 \end{pmatrix} \begin{pmatrix} U_1 \\ U_2 \end{pmatrix}$$

extracting equations from the system and transforming them to a linear system:

$$\ddot{x} = -19.6\theta \implies x(s)s^2 = -19.6\theta(s) \implies \frac{x(s)}{\theta(s)} = \frac{-19.6}{s^2}$$

this isn't the equation we were looking to find but it might prove helpful.

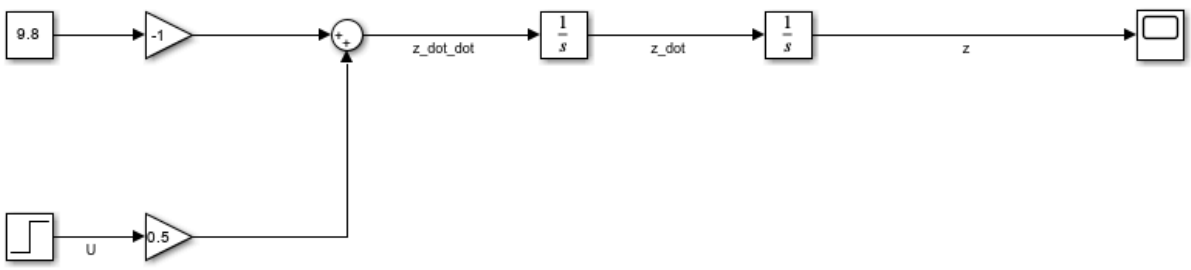
$$\ddot{\theta}(t) = 6u_2(t) \implies s^2\theta(s) = 6u_2(s) \implies \frac{\theta(s)}{u_2(s)} = \frac{6}{s^2}$$

$$\ddot{z}(t) = 0.5u_1(t) \implies \frac{z(s)}{u_1(s)} = \frac{0.5}{s^2}$$

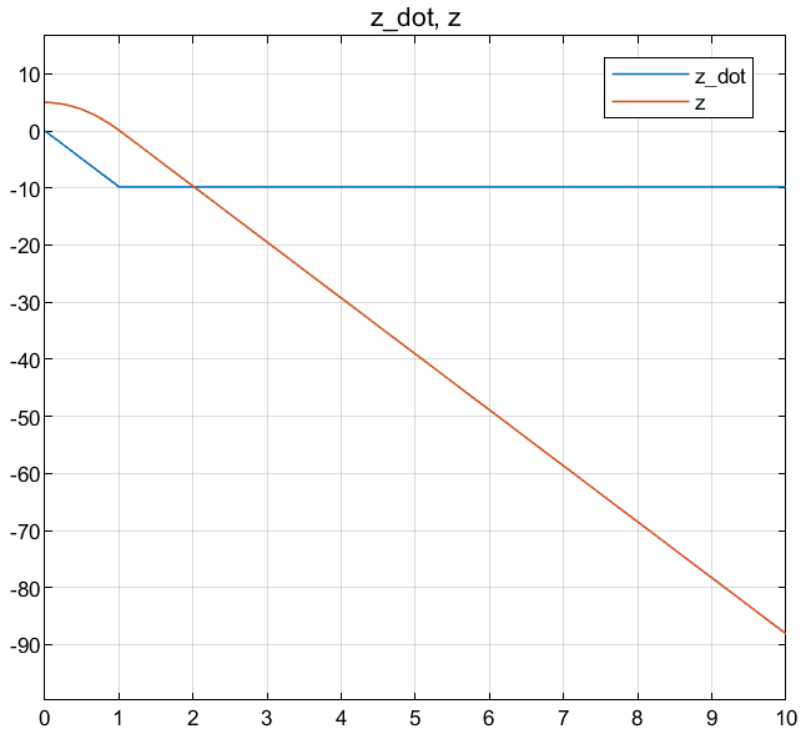
we found all the required transfer functions.

4.2 - Simulink simulation of system from u_1 to z

(we need to include gravity too)
we'll note that the nonlinear system $\ddot{z} = \frac{U_1}{m}\cos(x_5) - g$ was linearized, so when we add the constant g back we'll place it where it was in the original equation. $\ddot{z} = \frac{U_1}{2} - g$ and wire it in Simulink:



(note that they wanted the output as a result of a step input, so treating u as the control input we'll place a step of size 19.6)



analyzing the scope output:
we can see that $\dot{z}$ starts falling before the control signal kicks in (step at $t = 1$, once that happens the gravitational force is balanced out by the propulsion from the engines, but the drone has already developed some initial velocity that isn't cancelled out so the drone keeps dropping. The forces acting on the drone from $t = 1$ onwards are balanced out so the acceleration drops to 0, but the velocity remains constant and the displacement grows linearly.

Question 4.3 - open and closed loop transfer functions

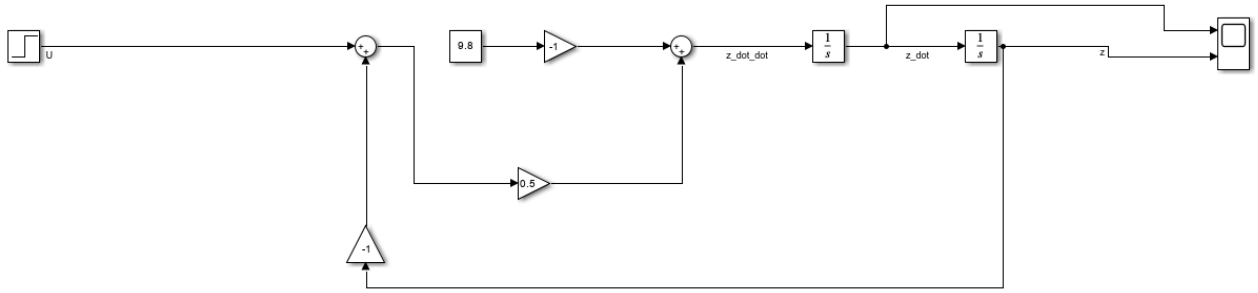
We'll note that we know $\ddot{z} = \frac{U_1}{2}$ (ignoring g in the analysis), so we already computed our transfer function -

$$\ddot{z}(t) = 0.5u_1(t) \implies \frac{z(s)}{u_1(s)} = \frac{0.5}{s^2}$$

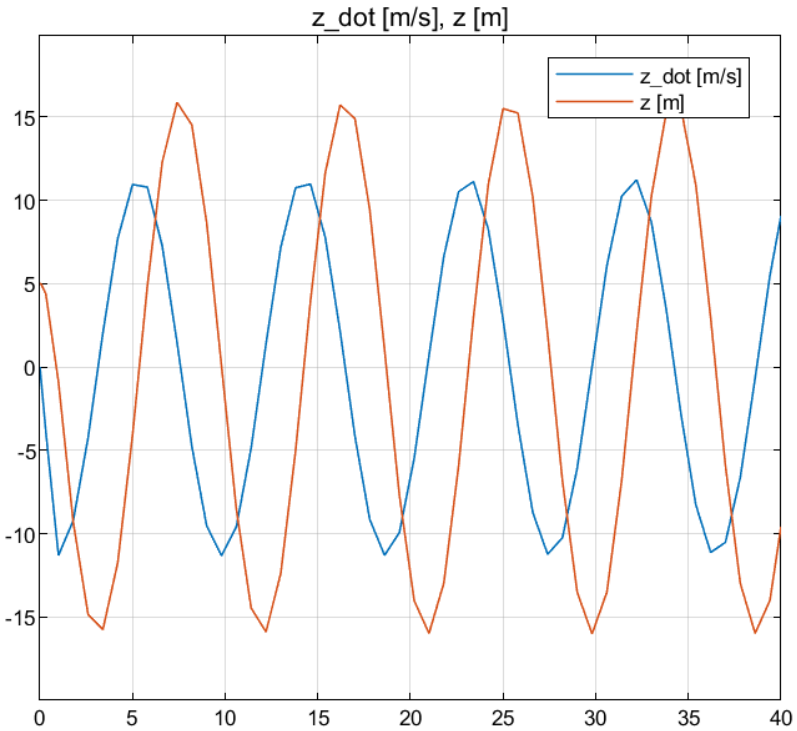
and for the closed loop function:

$$\begin{aligned} \ddot{z}(t) =& 0.5(u_1(t) - z(t)) \\ s^2 z(s) =& 0.5u_1(s) - 0.5z(s) \\ z(s)(s^2 + 0.5) =& 0.5u_1(s) \\ H_{cl}(s) = \frac{z(s)}{u_1(s)} =& \frac{0.5}{s^2 + 0.5} \end{aligned}$$

alternatively $H_{cl}(s) = H_{ol}(s)/(H_{ol}(s) + 1)$
rewiring our system accordingly:

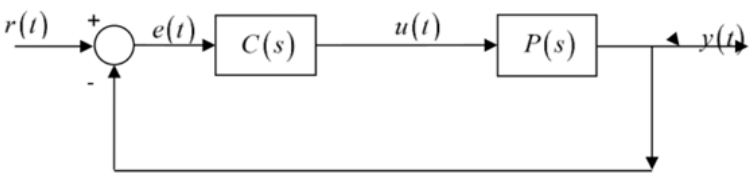


we'll note that our system now has poles at $0.25j$ and $-0.25j$ so we'd expect it to be unstable and unrestrained (to behave similarly to sines without decay)
and that's exactly what happens when we simulate that system:



we get sines that don't decay.
We will however note that this simulation is entirely unphysical since our drone "goes underground".

Question 4.4 - PID for the system



we need to design a PID where $P(s)$ is the system's original transfer function.
$C(s) = k_p + k_d s + k_i \frac{1}{s}$ is the PID controller
where the reference input $r(t)$ is a unit step.
with the following requirements:

- overshoot of 20% of steady state
- time to stabilize of at most $3[s]$ around 5% of the steady state value.

  parameters are functions of m.

To start we'll follow the instructions and convert the demands into the locations of two poles in a 2nd order system:

$$t_s(5\%) \approx \frac{3}{\zeta\omega_n}$$

$$O.S = 0.2 = \exp\left(\frac{-\pi\zeta}{\sqrt{1-\zeta^2}}\right)$$

we write a script to solve the two equations:

```
syms zeta wn ts real positive
OS = 0.2;
eq1 = OS == exp(-pi*zeta / sqrt(1 - zeta^2));
zeta_sol = vpasolve(eq1, zeta, [0, 1]);
zeta_val = double(zeta_sol);
ts_val = 3;
eq2 = ts_val == 3 / (zeta_val * wn);
wn_sol = solve(eq2, wn);
wn_val = double(wn_sol);
fprintf('Solved damping ratio ζ = %.4f\n', zeta_val);
fprintf('Solved natural frequency ω_n = %.4f rad/s (for settling time %.2fs)\n', wn_val, ts_val);
%% get poles:
sigma = -zeta_val * wn_val;
omega_d = wn_val * sqrt(1 - zeta_val^2);
s1 = sigma + 1i * omega_d;
s2 = sigma - 1i * omega_d;
fprintf('Pole 1: %.4f %+.4fi\n', real(s1), imag(s1));
fprintf('Pole 2: %.4f %+.4fi\n', real(s2), imag(s2));
% Characteristic polynomial coefficients
a = 1;
b = 2 * zeta_val * wn_val;
c = wn_val^2;
% Create the polynomial: s^2 + b*s + c
poly_coeffs = [a, b, c];

fprintf('Characteristic polynomial: s^2 + %.4fs + %.4f\n', b, c);
```

and got the following output:

```
Solved damping ratio ζ = 0.4559
Solved natural frequency ω_n = 2.1932 rad/s (for settling time 3.00s)
Pole 1: -1.0000 +1.9520i
Pole 2: -1.0000 -1.9520i
Characteristic polynomial: s^2 + 2.0000s + 4.8102
```

now that we know the poles have a real component of -1, to place a pole that won't interfere with them we'd select $s = -5$, so our desired polynomial would be:

$(s^2 + 2s + 4.8102)(s + 5) = s^3 + 7s^2 + 14.8102s + 24.051$

let's now compute the actual polynomial as a function of m and compare

coefficients:
$H(s) \triangleq C(s)P(s) = \frac{1}{ms^2+1} \cdot (k_p + k_d s + k_i \frac{1}{s})$

$$H_{closed}(s) = \frac{C(s)P(s)}{1 + C(S)P(S)} = \frac{k_p + k_d s + k_i \frac{1}{s}}{ms^2 + 1 + k_p + k_d s + k_i \frac{1}{s}} = \frac{\frac{1}{m}(k_p s + k_d s^2 + k_i)}{s^3 + (s + k_p s)/m + (k_d/m)s^2 + k_i/m}$$

$$\implies s^3 + 7s^2 + 14.8102s + 24.051 = s^3 + (s + k_p s)/m + (k_d/m)s^2 + k_i/m$$

$$\begin{cases} 7 = \frac{k_d}{m} \\ 14.8102 = (1 + k_p)/m \\ 24.051 = k_i/m \end{cases} \implies \begin{cases} k_d = 7m \\ k_p = 14.8102m - 1 \\ k_i = 24.051m \end{cases}$$

we need to analyze the *steady state error.*

let's use the final value theorem:

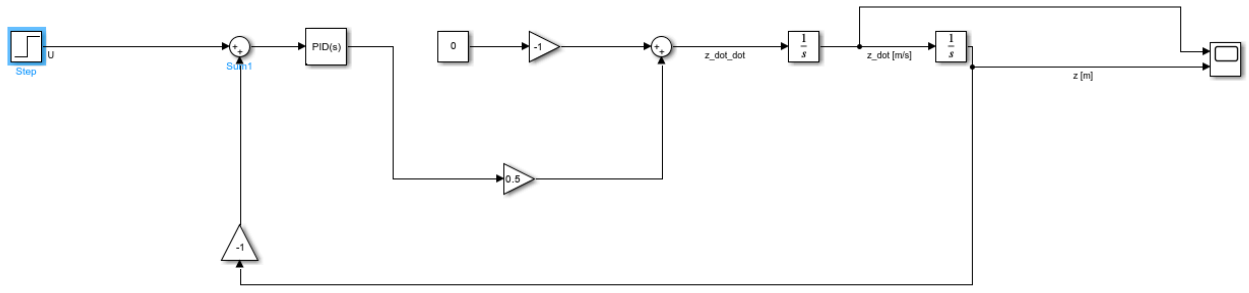
$$\lim_{t\to\infty} h_{closed}(t)r(t) = \lim_{s\to 0} s \cdot \frac{1}{s} \frac{\frac{1}{m}(k_p s + k_d s^2 + k_i)}{s^3 + (s + k_p s)/m + (k_d/m)s^2 + k_i/m} = 1$$

which is awesome! Our system tracks the reference step correctly.

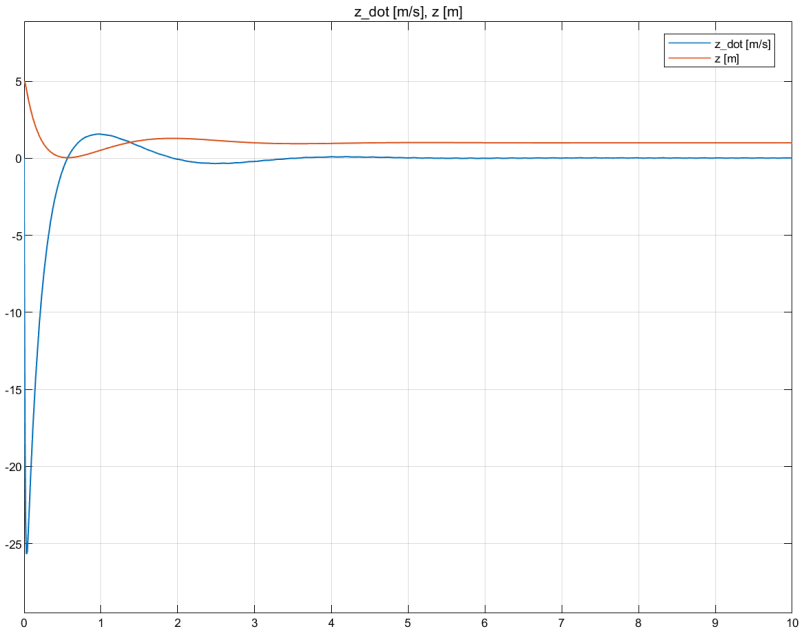
Question 4.5 - simulating the PID in Simulink

we'll note that in our specific case $k_d = 14, k_p = 28.6204, k_i = 48.102$

wiring our PID:

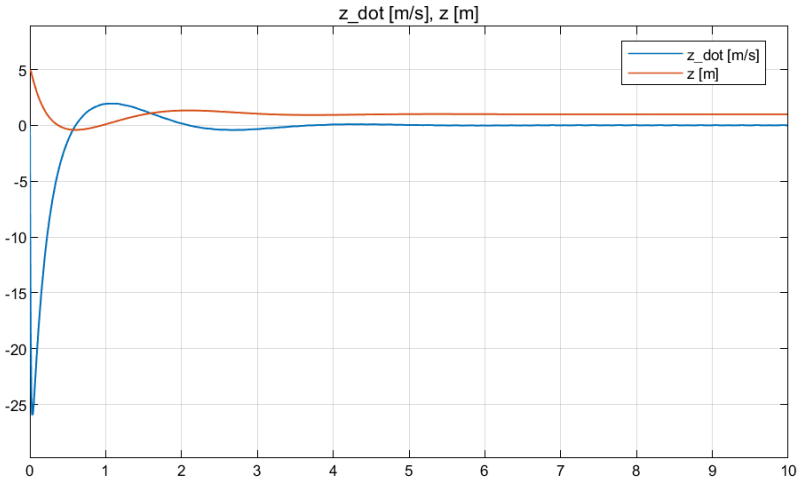


without g



we can see as expected a steady state value of $z = 1$ and that around three seconds our PID actually does reach a 5% sleeve with respect to it, however - we demanded an overshoot of 20% at most (at most 1.2), and z at its peak reached values of 1.5, which isn't too bad in terms of values but is percentage wise - the reasons for that are probably relating to the addition of another pole - while the convergence time is dominated by the two poles we decided on, the third could still somewhat contribute to the amplitude.

with g



the addition of g on the surface doesn't seem to change anything, which should be somewhat expected since we're adding a constant to a system controlled by a PID, which is robust to such changes - in particular the integrator guarantees a steady state tracking error of 0 on constants. However, a closer inspection shows that our system does perform more poorly - our overshoot now stands at 2 (100% overshoot!).

Our timed response does fulfill our requirements though - we do enter a 5% sleeve at the 3 second mark, and we don't exit it.

Question 5.1 - regulation with state feedback

we'll re-recall that:

$$\dot{X} = \begin{pmatrix} 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & -19.6 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix} \begin{pmatrix} x \\ \dot{x} \\ z \\ \dot{z} \\ \theta \\ \dot{\theta} \end{pmatrix} + \begin{pmatrix} 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ \frac{1}{2} & 0 \\ 0 & 0 \\ 0 & 6 \end{pmatrix} \begin{pmatrix} U_1 \\ U_2 \end{pmatrix}$$

and now reshape the equation to be around $X_{eq} = (2, 0, 5, 0, 0, 0)^T$, We'll note that $\delta u = U - U^*$ and $\delta x = X - X^*$, meaning, that to stabilize our new system the desired state would be $X = \underline{0}$ and $U = \underline{0}$.

5 רגולציה באמצעות משוב מצב

5.1

נרשום את משוואות המצב לאחר הלינאריזציה. ראינו בבקרה 1 כי עבור סטיות מנקודת שיווי המשקל, מתקבלת משוואת המצב הבאה:

$$\dot{\delta x} = A\delta x + B\delta u$$

כאשר: $\delta x = x - x_{eq}$ $\delta u = u - u_{eq}$

הם וקטורי המצב והכניסה החדשים.

עבור רגולציה באמצעות משוב מצב אנו רוצים שסטייה בכניסה תתקיים רק עבור סטיית משתני המצב מנקודת שיווי המשקל, לפיכך:

$$\delta u = -K\delta x$$

המצב הרצוי עבור המערכת הזו כדי לייצב את המערכת המקורית סביב נקודת שיווי המשקל, הוא שווקטור משתני המצב, ולפיכך גם הכניסה, שווים שניהם ל-0, כלומר אין סטייה מנקודת שיווי המשקל.

5.2

בהתאם לדרישות, תכננו במטלאב משוב מצב עבור המערכת וקיבלנו:

```
הקטבים שנבחרו הם:
  -4.0000 + 5.4575i  -4.0000 - 5.4575i -10.0000 + 0.0000i -11.0000 + 0.0000i -12.0000 + 0.0000i -13.0000 + 0.0000i

מטריצת ההגברים K (2x6) היא:
  101.8057   30.0546  248.3212   44.7226  -46.2307   -2.4464
 -108.0153  -37.2939   -0.9759   -0.0848   62.2940    5.2731
```

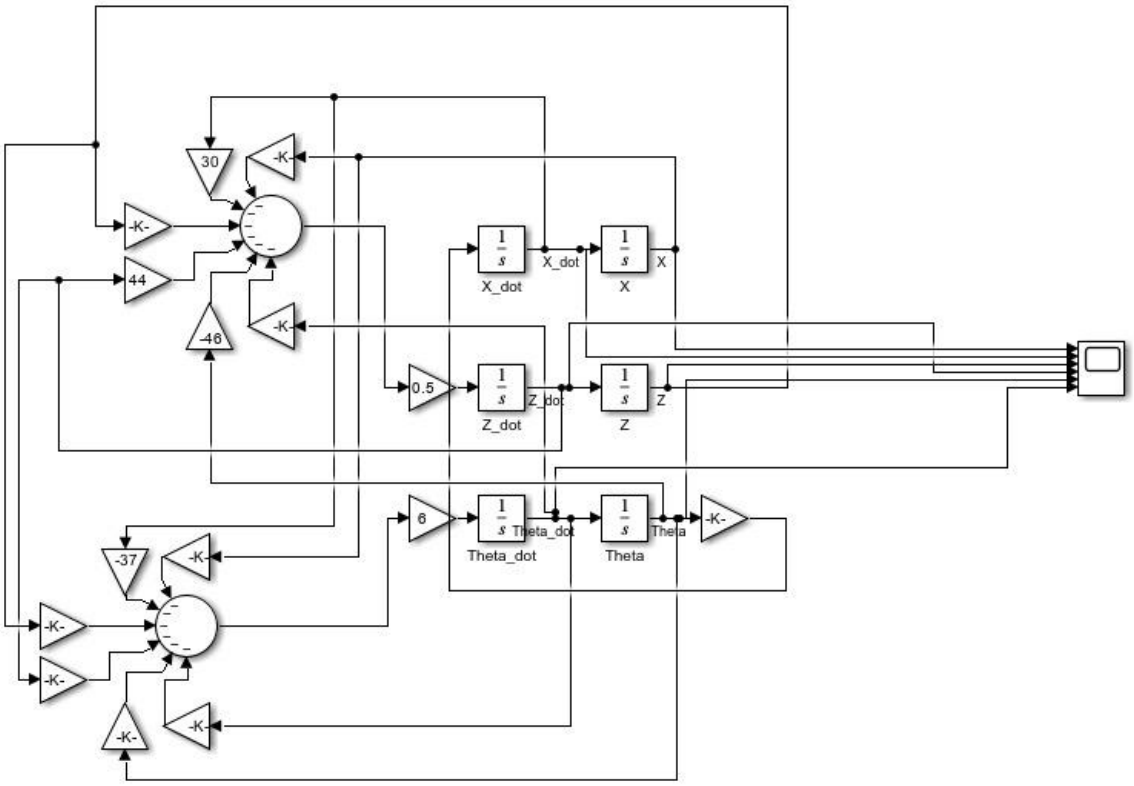
הקוד:

```
A = [0, 1, 0, 0, 0, 0;
     0, 0, 0, 0, -9.8, 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];

B = [0, 0;
     0, 0;
     0, 0;
     0.5, 0;
     0, 0;
     0, 6];
```

```
% דרישות דינמיות
Ts = 1;
OS = 10;

% חישוב ζ ו-ω_n
zeta = -log(OS/100)/sqrt(pi^2 + log(OS/100)^2);
omega_n = 4 / (zeta * Ts);

% שני קטבים דומיננטיים
p1 = -zeta * omega_n + 1i * omega_n * sqrt(1 - zeta^2);
p2 = conj(p1);

% שאר הקטבים – נמקם רחוק יותר על הציר הממשי
p3 = -10;
p4 = -11;
p5 = -12;
p6 = -13;

desired_poles = [p1, p2, p3, p4, p5, p6];

if rank(ctrb(A, B)) < size(A,1)
    error('המערכת אינה ברת ניתוב (Not controllable).');
end

% חישוב מטריצת ההגברים K
K = place(A, B, desired_poles);

% תצוגת תוצאות
disp('הקטבים שנבחרו הם:');
disp(desired_poles);

disp('מטריצת ההגברים K (2x6) היא:');
disp(K);
```

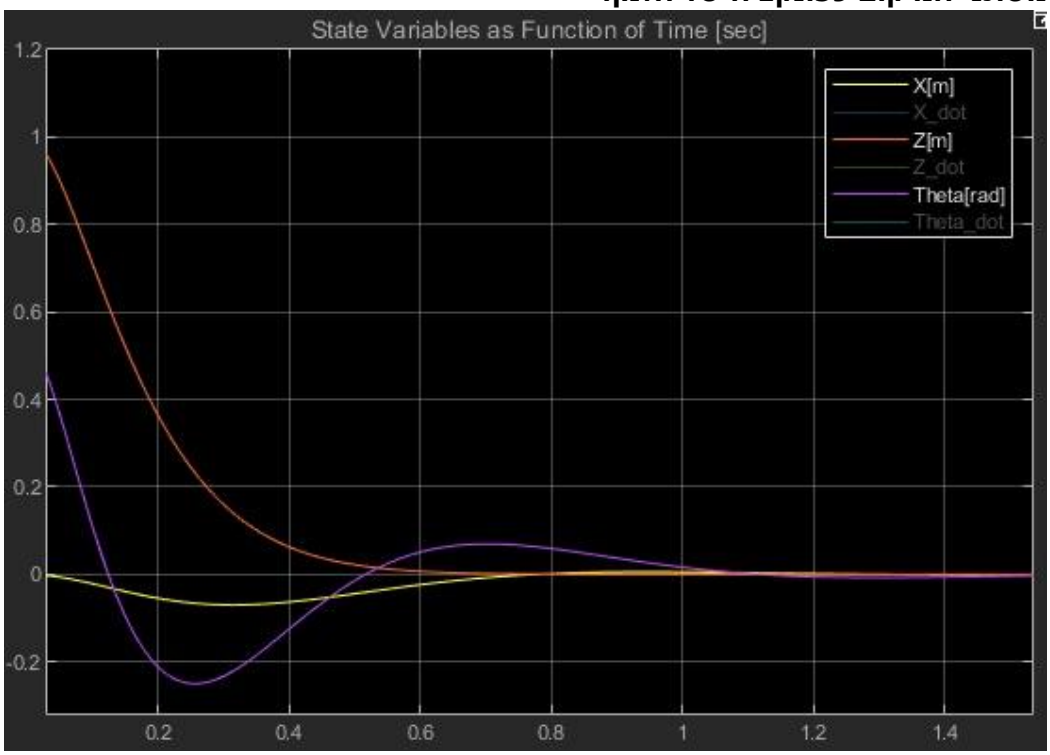
5.3

בנינו דיאגרמת בלוקים בסימולינק:



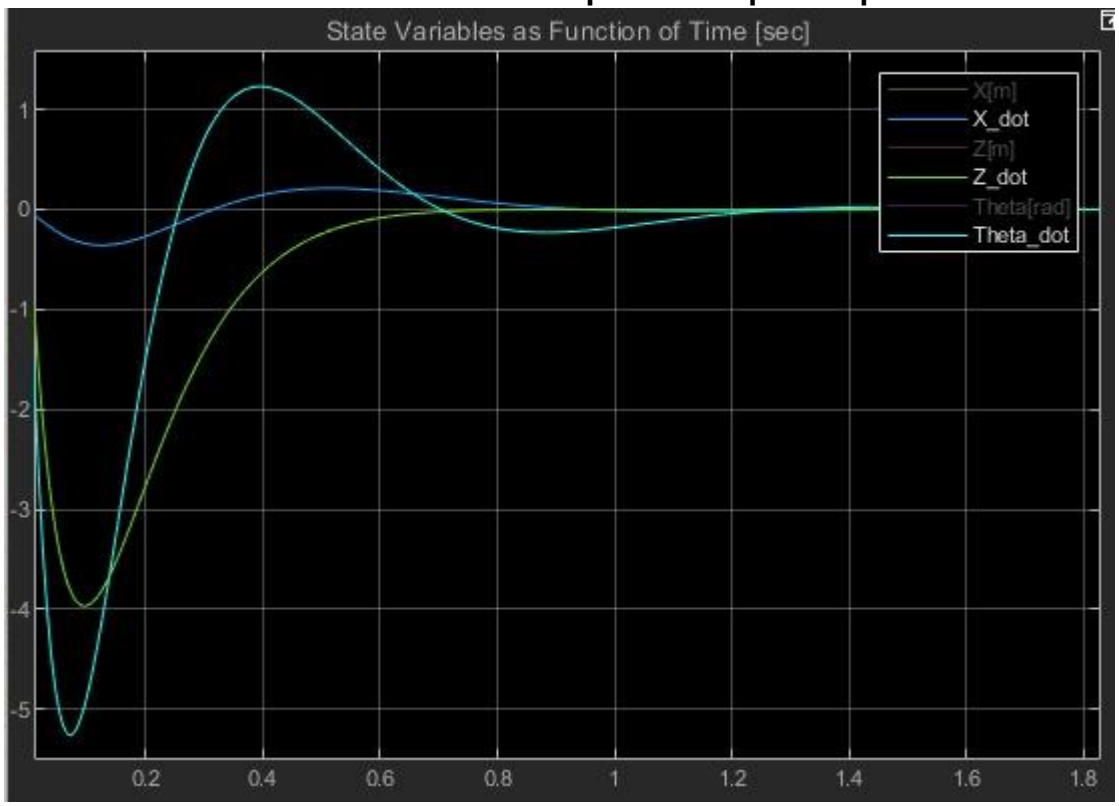
הרצנו אותה מתנאי ההתחלה הנתונים, והרצנו את המערכת. הדפסנו את התנהגות משתני המיקום בנפרד מנגזרותיהן על מנת שהגרפים יהיו קריאים:

משתני המיקום כפונקציה של הזמן:



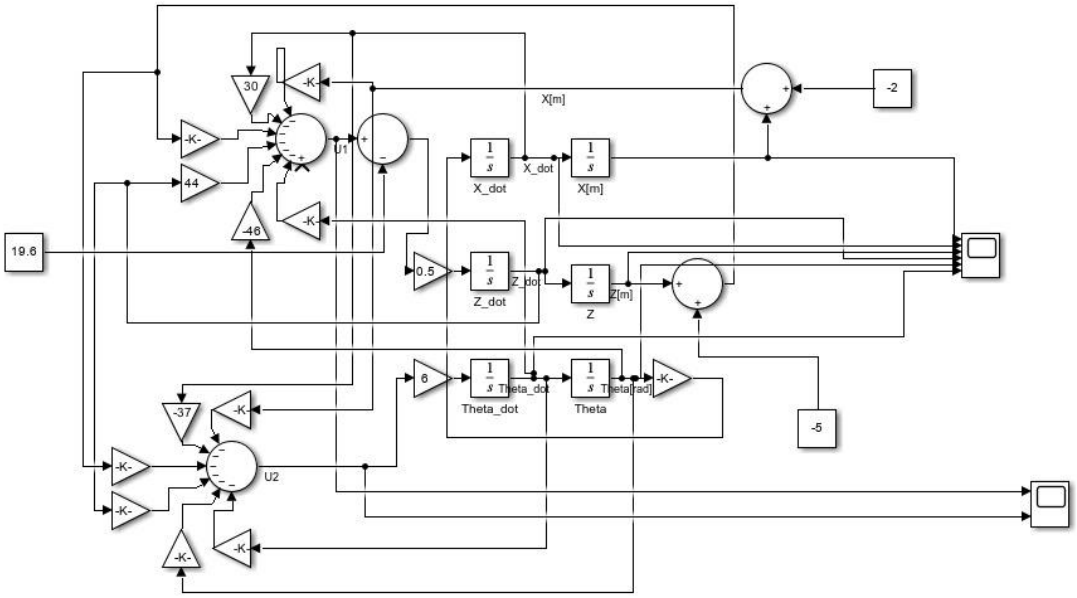
ניתן לראות כי אכן עבור משתני המיקום אנו עומדים בדרישות על זמן ההתייצבות ותגובת יתר. אמנם הזווית חורגת מתגובת היתר הרצויה, אך באמצעות שינוי הזווית אנו שולטים במיקום האופקי, ולכן אין לראות בכך פגם.
ניתן להבין מהגרף כי הרחפן מתקן את גובהו בצורה מונוטונית, בעוד האוריינטציה שלו בעלת תנודתיות עד התייצבותה על הערך הרצוי. בשל שינוי האוריינטציה, הרחפן משנה מעט גם את מיקומו האופקי עד שלבסוף הוא חוזר למיקום הרצוי.

נגזרות משתני המיקום כפונקציה של הזמן:

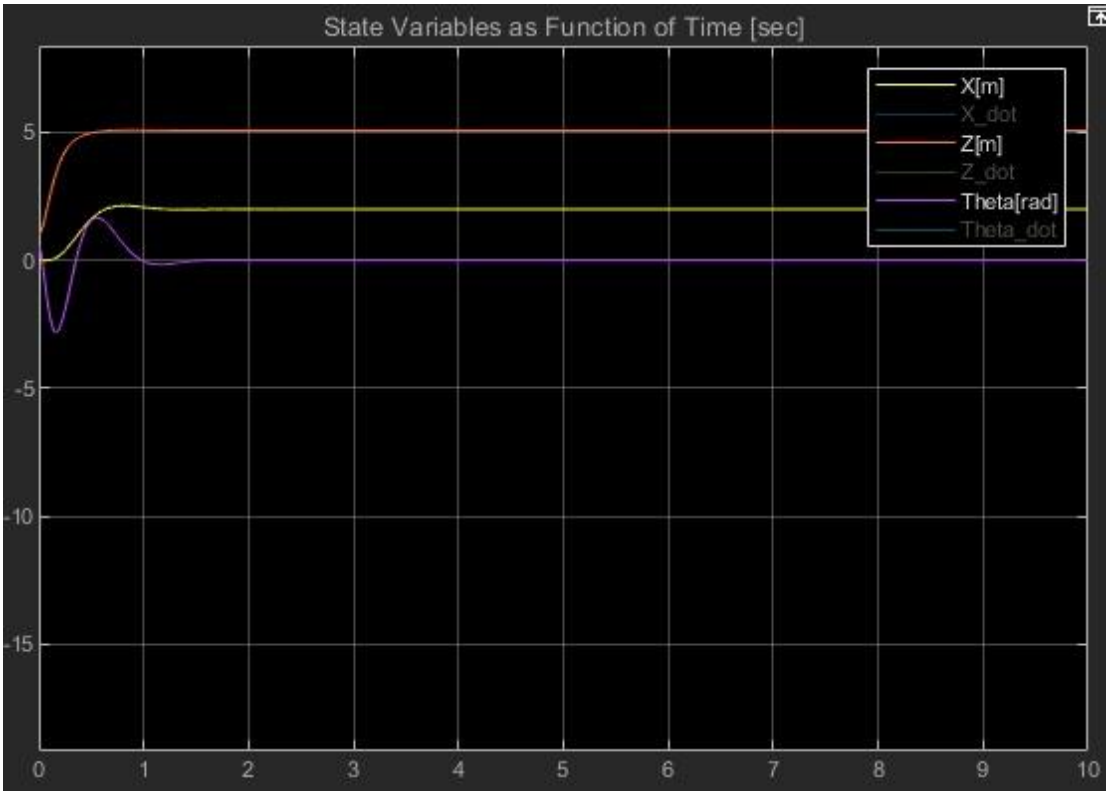


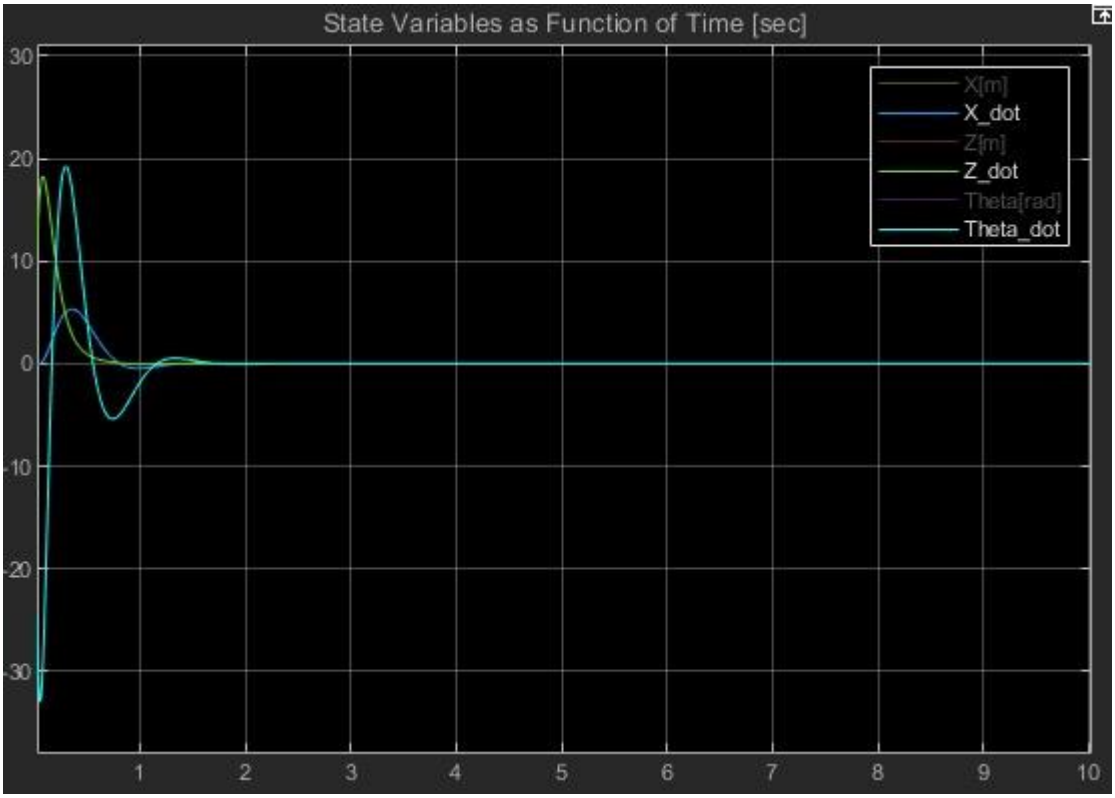
5.4

נזכיר כי בנקודת שיווי המשקל סביבה ביצענו לינאריזציה, קיבלנו כי U1 = 19.6, U2=0. נוסיף ערכים אלו במקומות המתאימים. בנוסף עלינו להוסיף את משתני המצב בערכם בשיווי המשקל. **(1)**

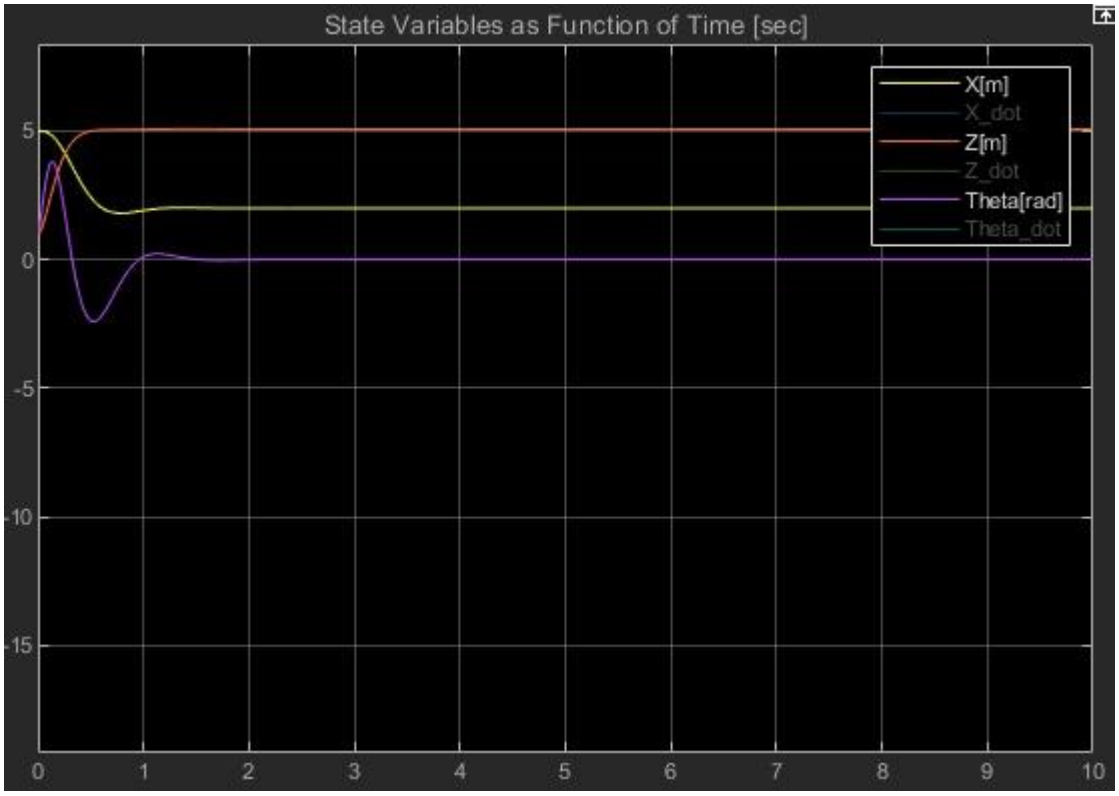


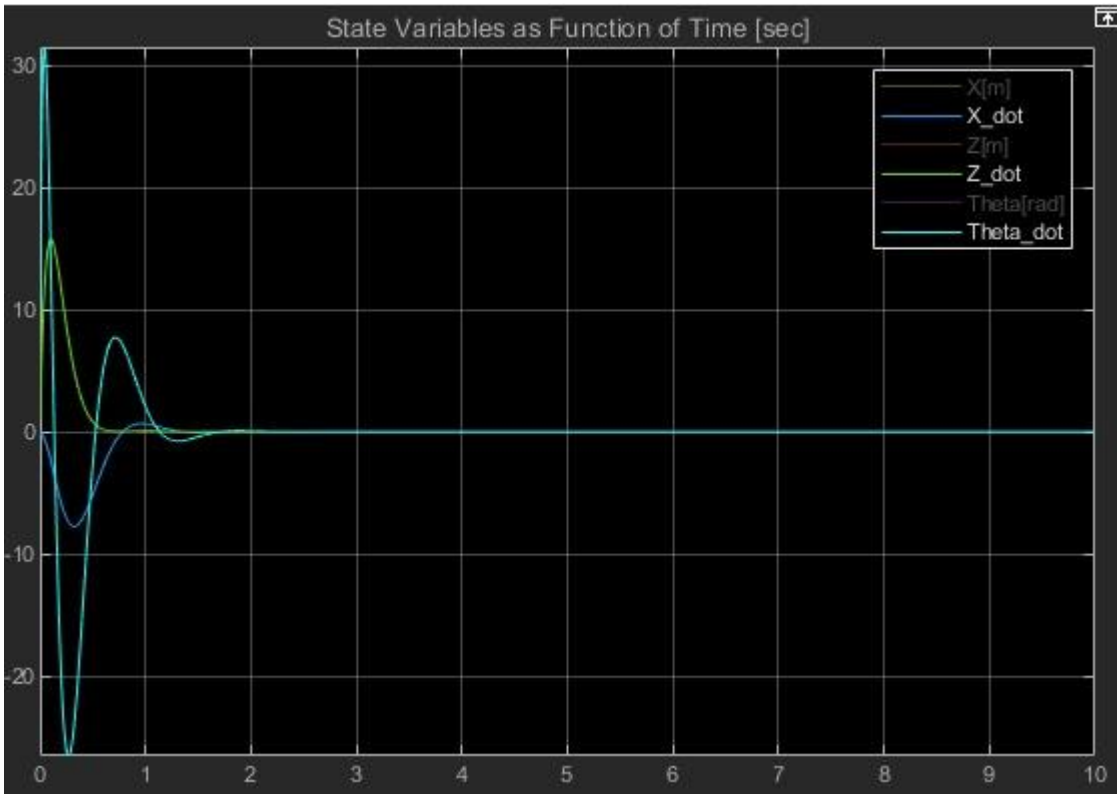
קבענו את תנאי ההתחלה השונים בהתאם לכתוב בסעיף, הרצנו את הסימולציה וקיבלנו:
(1)



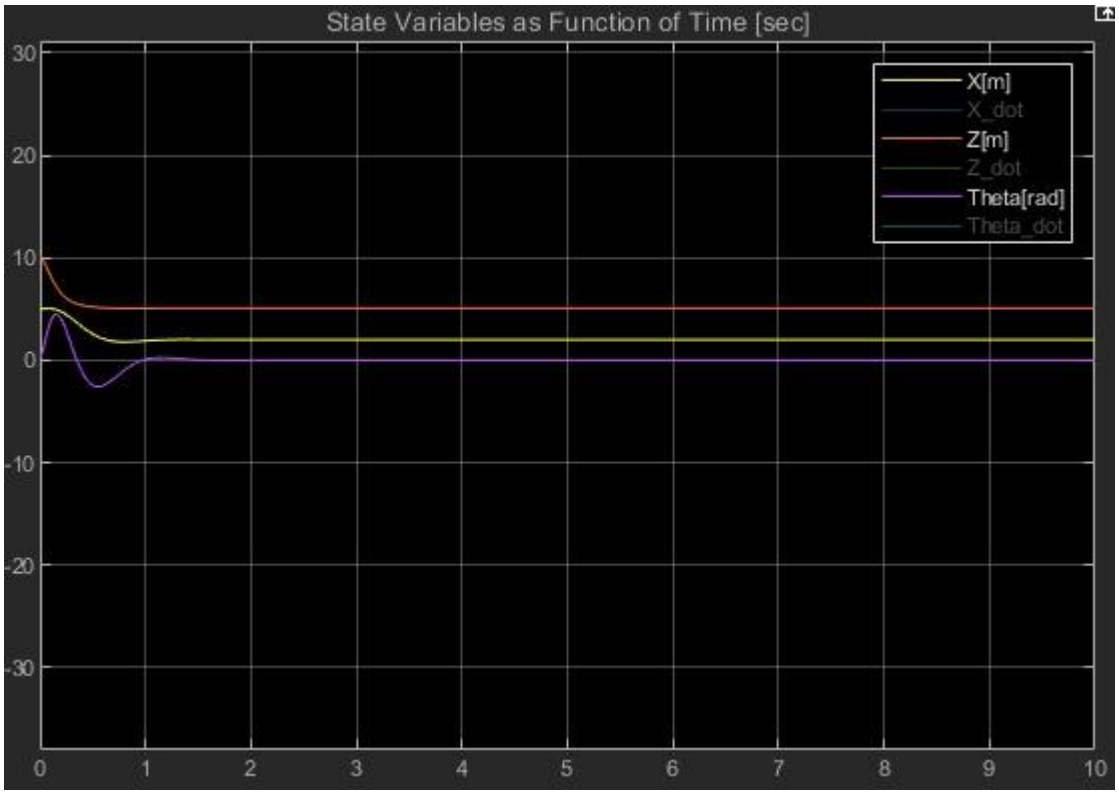


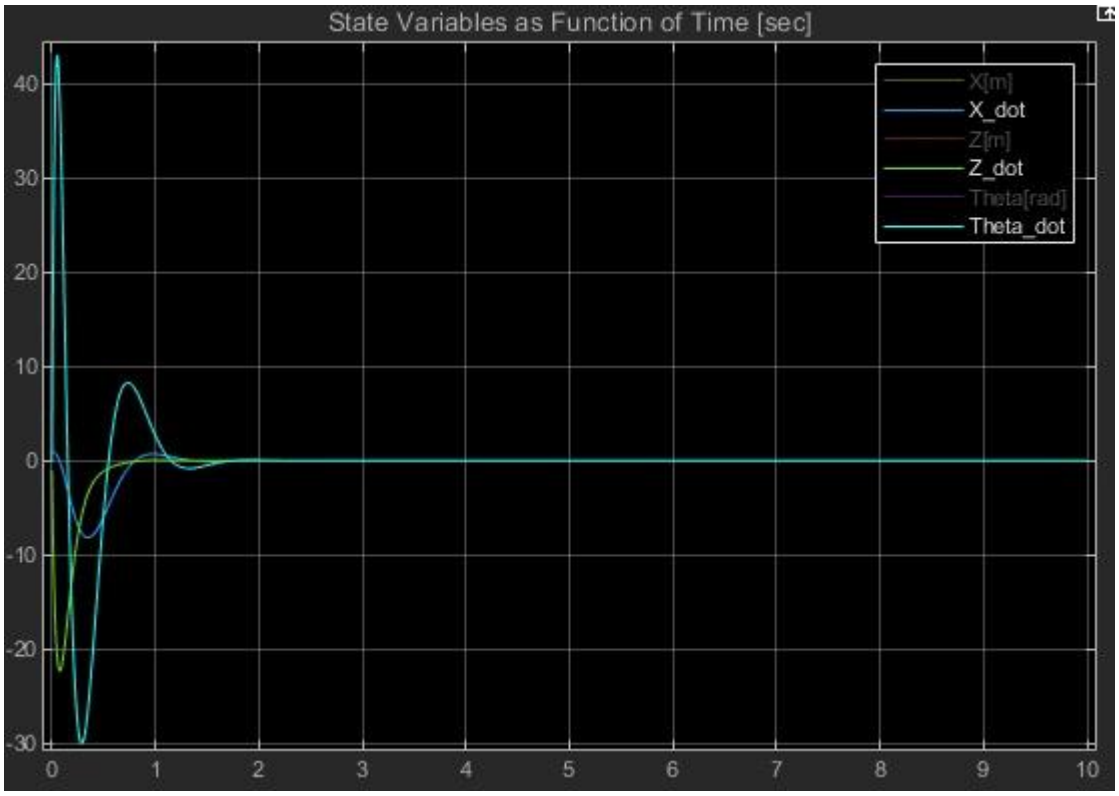
(2)


State Variables as Function of Time [sec]
X[m]
X_dot
Z[m]
Z_dot
Theta[rad]
Theta_dot
5
0
-5
-10
-15
0
1
2
3
4
5
6
7
8
9
10


State Variables as Function of Time [sec]
X[m]
X_dot
Z[m]
Z_dot
Theta[rad]
Theta_dot
30
20
10
0
-10
-20
0
1
2
3
4
5
6
7
8
9
10

(3)



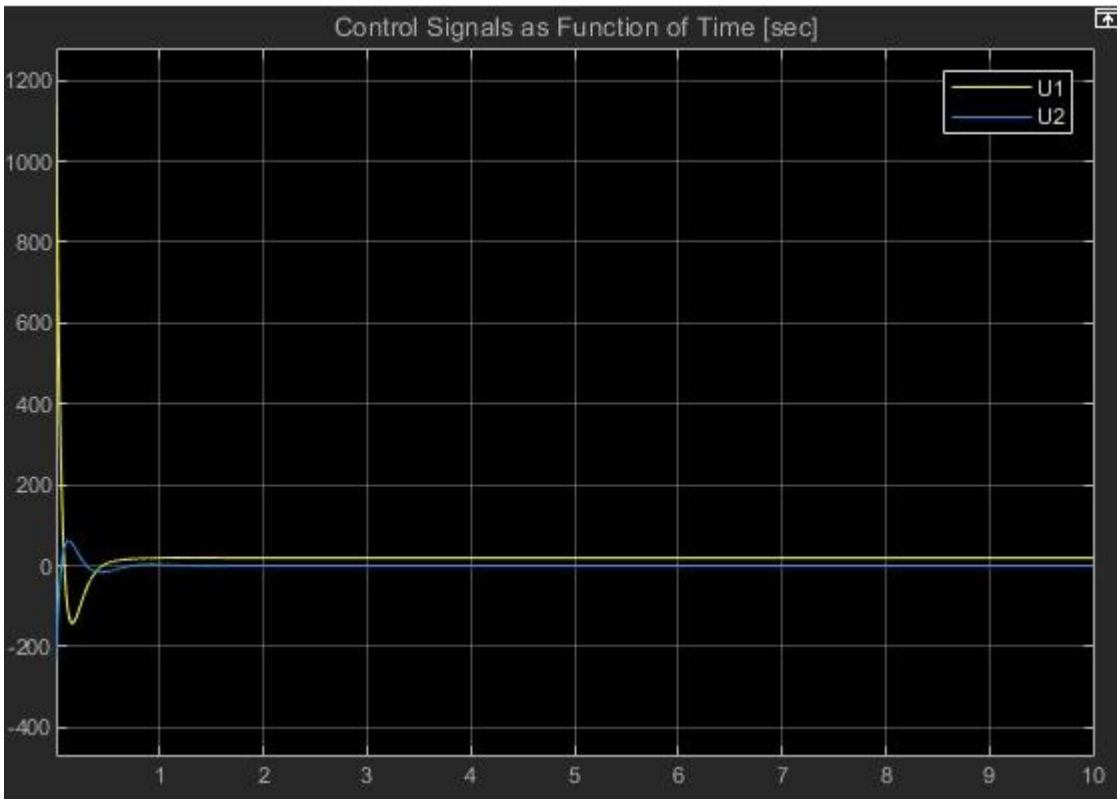


ניתן לראות כי הצלחנו לייצב את הרחפן סביב נקודת שיווי המשקל משלושת תנאי ההתחלה שנבדקו, ועמדנו בתנאים עבור משתני המיקום, למעט זווית הרחפן, כפי שהסברנו קודם, שדרכה אנו משנים מיקום אופקי ולכן היא לא יכולה להיות תמיד מונוטונית לכיוון הזווית הרצויה.

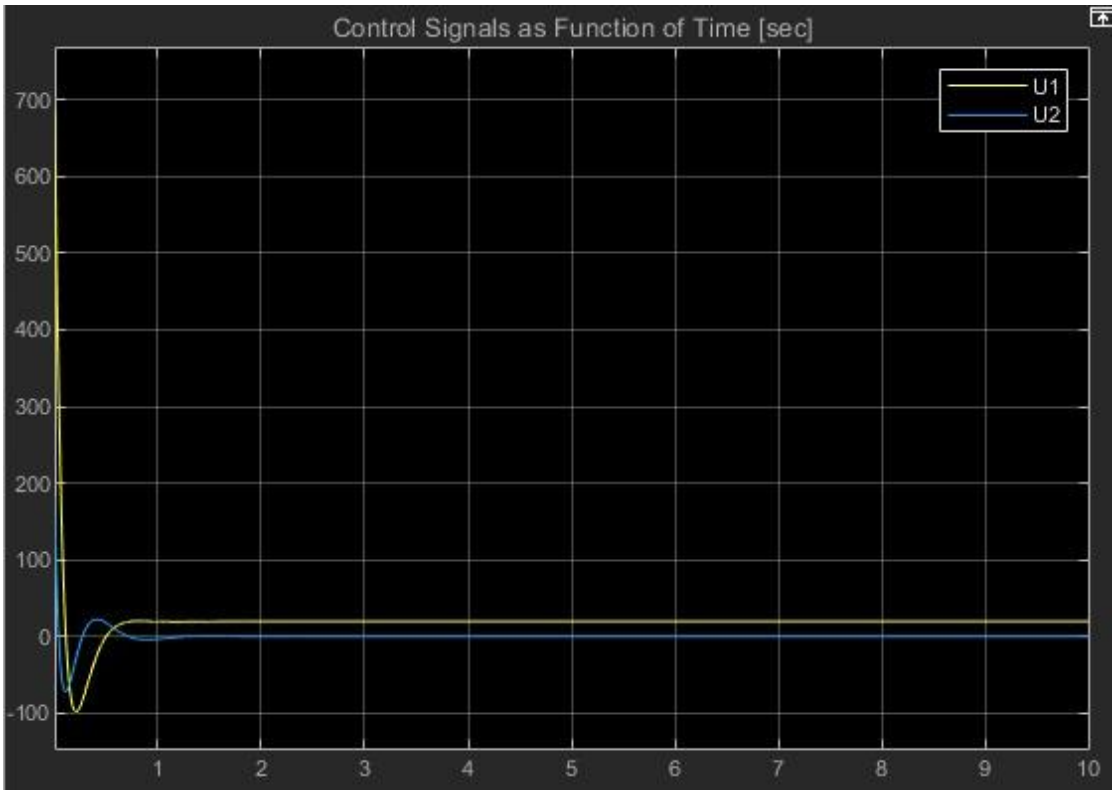
5.5

הערה משותפת ל3 תנאי ההתחלה: נראה כי אות הבקרה הראשון מתייצב על ערך שאינו 0. זאת בשל הצורך הקבוע להתגבר על כוח הכבידה אף כאשר הגענו כבר לנקודת שיווי המשקל הרצויה.

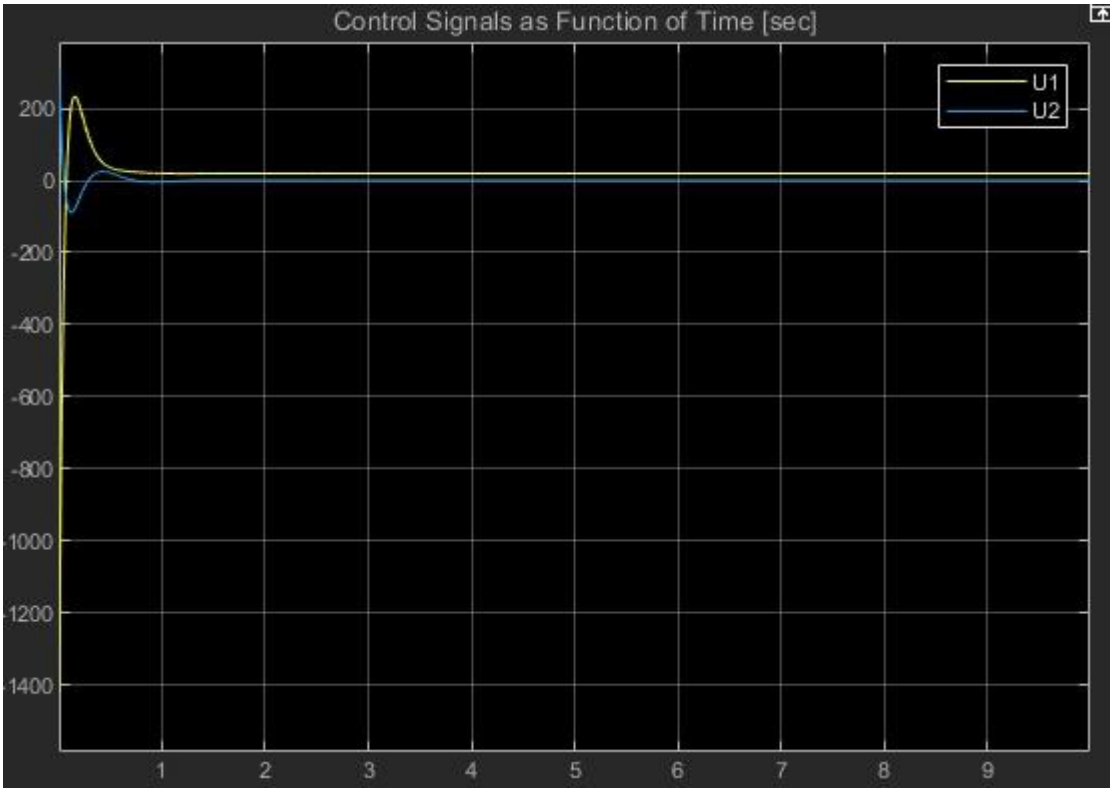
(1) הרחפן מתחיל נמוך מנקודת שיווי המשקל, לכן אות הבקרה הראשון הוא חיובי וגדול בהתחלה, עד שהרחפן מתקרב לנקודת שיווי המשקל (ואז הוא אפילו מעט שלילי, מה ששקול לכוח הפוך בכיוונו שמפעילים המדחפים, ואינו פיזיקלי במודל שלנו). הרחפן מתחיל משמאל לנקודת שיווי המשקל ולכן אות הבקרה השני מתחיל שלילי כדי להטות את הרחפן לזווית ימינה וכך הוא יזוז ימינה בכח המדחפים.



(2) הרחפן מתחיל נמוך מנקודת שיווי המשקל לכן אות הבקרה הראשון הוא חיובי וגדול מאוד בהתחלה, עד שהוא מתקרב לגובה הרצוי ואז האות קטן (ואף נהיה מעט שלילי).
הרחפן מתחיל ימינה מנקודת שיווי המשקל ולכן אות הבקרה השני מתחיל חיובי על מנת להטות את הרחפן שמאלה וכך המדחפים יגרמו לתזוזתו שמאלה לעבר נקודת שיווי המשקל.

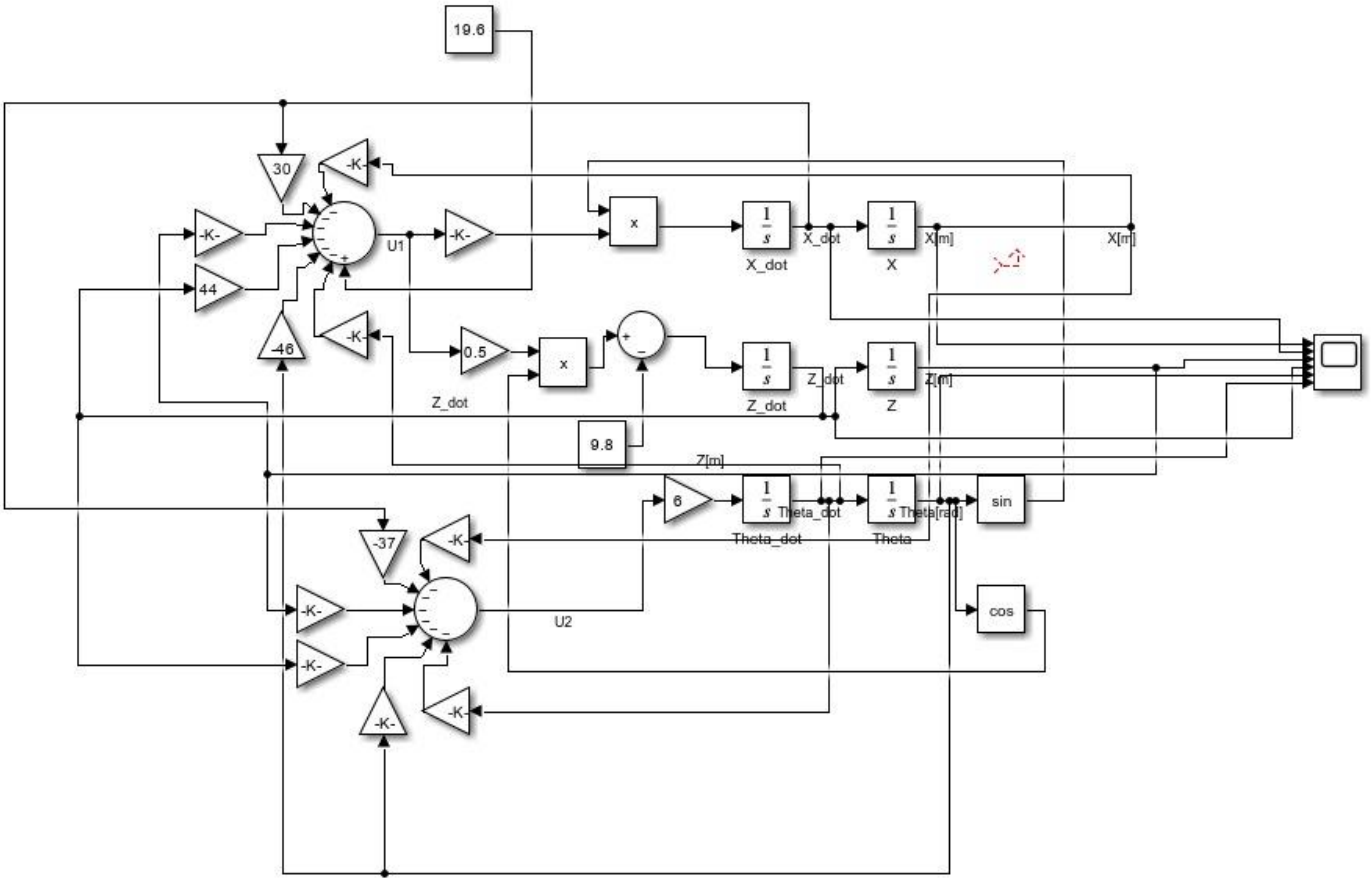


(3) הרחפן מתחיל מעל נקודת שיווי המשקל, לכן אות הבקרה הראשון מתחיל שלילי (שוב ההערה שהערנו קודם), עד שהוא מתקרב לנקודת שיווי המשקל ואז הוא גדל ואף נהיה חיובי מעט. הרחפן מתחיל מימין לנקודת שיווי המשקל ולכן אות הבקרה השני מתחיל חיובי על מנת להטות את הרחפן שמאלה וכך הוא ינוע לעבר נקודת שיווי המשקל.



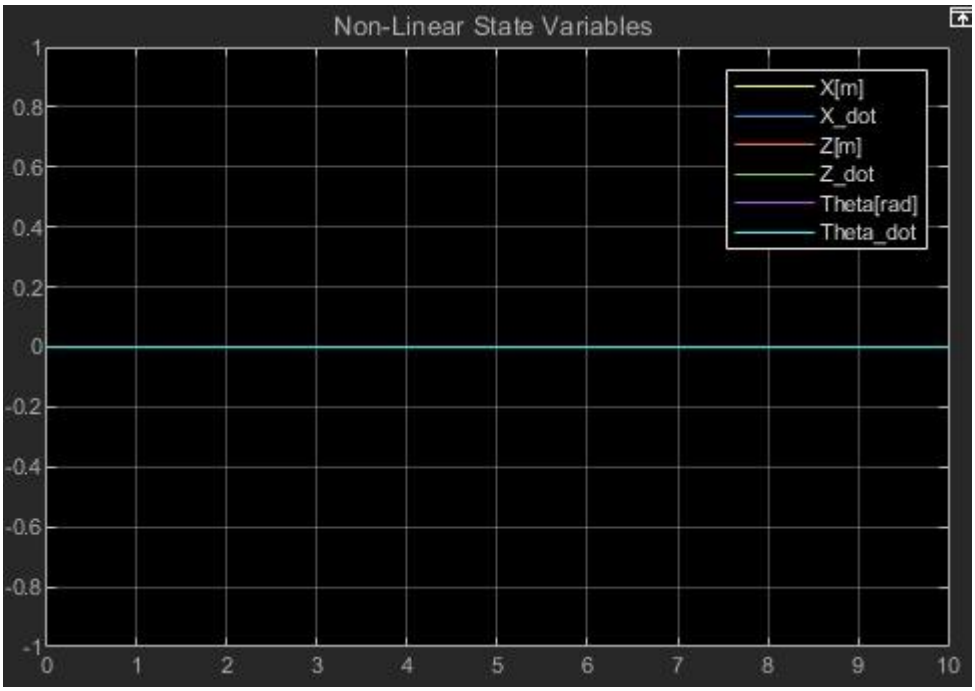
5.6

הערה: לפי הנתונים והניסוח אנו מבינים שתנאי ההתחלה מבטאים את הערך ההתחלתי של המיקום, ולא את הסטייה מנקודת שיווי המשקל. בשל כך, אנו מניחים שכוונת הסעיף היתה לייצב את הרחפן סביב נקודת שיווי משקל שהיא בווקטור ה-0 (אחרת הנש"מ מהסעיף הקודם רחוקה מהראשית, הנתונה למשל כתנאי התחלה בסעיף א', ולא נצליח לייצב את המערכת). בכל אופן כמובן שההתנהגות תהיה דומה אם הערכים הנתונים הם הסטיות מהנש"מ שבסעיפים הקודמים, עד כדי קבועים שנוספים לגרפים.

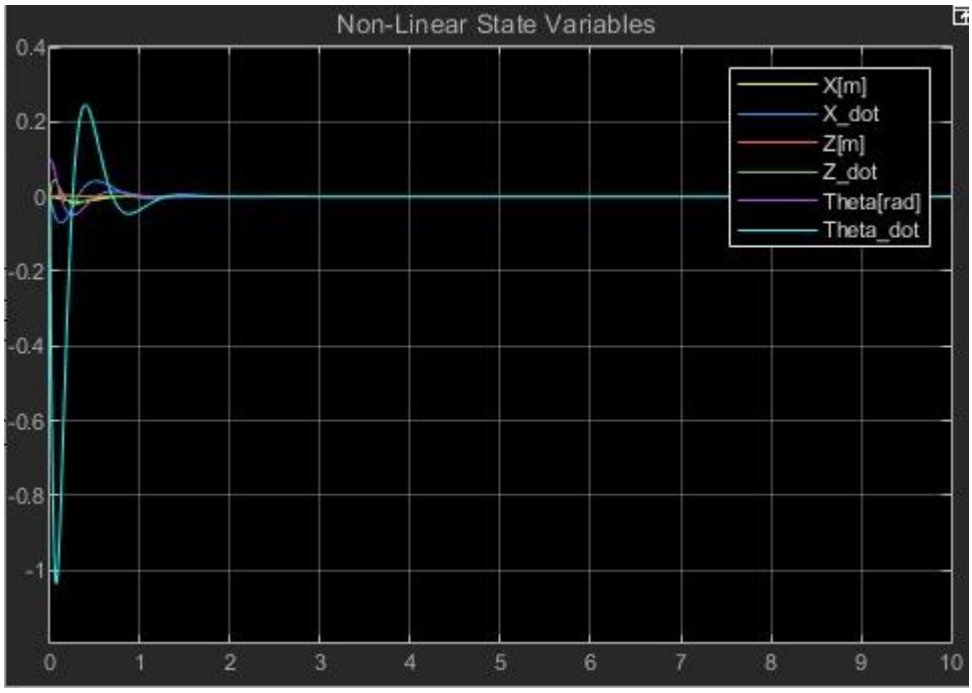


שרטוט המערכת:

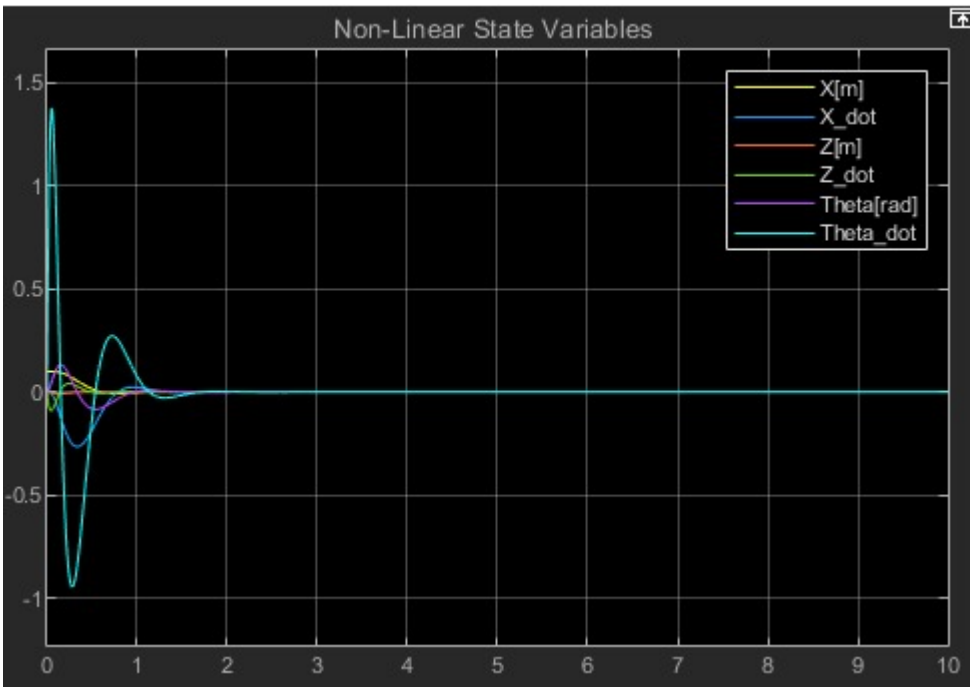
(1) התחלנו בתנאי התחלה 0 לכל המשתנים, ובגלל שאנחנו בנקודת שיווי המשקל, כמובן שנישאר בה.



(2) התחלנו בנטייה קלה של הזווית, קרובים מספיק לנקודת שיווי המשקל ולכן ניתן לראות שאנחנו מתכנסים לנקודת שיווי המשקל.

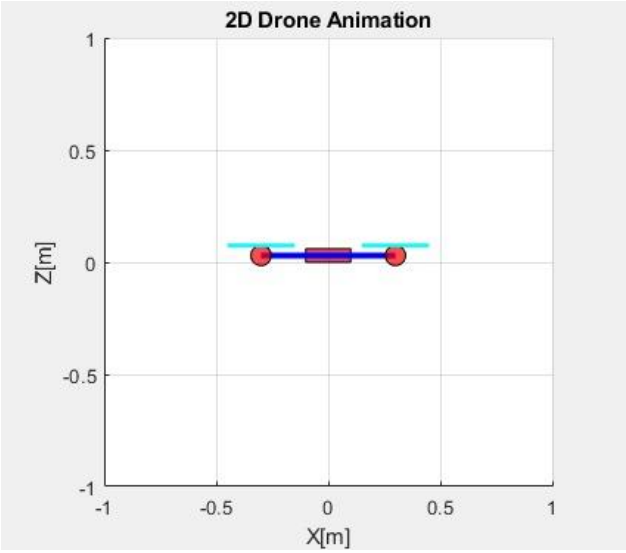


(3) כאן אנחנו מתחילים בגובה וזווית ששווה לנקודת שיווי המשקל, אך במיקום אופקי שונה במעט. בשל קרבתו לנקודת שיווי המשקל, גם כאן אנחנו מתכנסים אליה.



אנו רואים שכאשר אנחנו קרובים מספיק לנקודת שיווי המשקל אנחנו מצליחים לייצב את המערכת האמיתית (שאינה מקורבת), על אף שאנחנו עושים שימוש בווקטור הגברי משוב המצב שחישבנו תחת הנחת לינאריות המערכת.
אף על פי כן, ברור שאם אנחנו רחוקים מהנש"מ לא נוכל להבטיח התייצבות סביבה, ואנחנו עלולים לקבל התנהגות לא צפויה.

(5.7) עבור תנאי ההתחלה הראשון מהסעיף הקודם הרחפן כאמור לא יזוז ממקומו במערכת המקורית, וגם במערכת הלינארית. לכן מתקבל:



ליתר ביטחון: עבור תנאי ההתחלה הראשון מסעיף 5.4, אנחנו מקבלים: