Quadcopter Project

- **Goal:** Write a MATLAB code to control a quadcopter

- **Approach:**
  - Determine desired position \((x^*, y^*, z^*)\) (can be any function of time you like)
  - Determine actual position \((x, y, z)\)
  - Use a feedback controller to determine values for motor thrust, roll, pitch, and yaw rate that will correct any difference between \((x^*, y^*, z^*)\) and \((x, y, z)\)
Design Process

1. Complete some hand calculations to understand how feedback control works

2. Write a MATLAB code that predicts motion of quadcopter with feedback control

3. Use MATLAB code to calculate values for proportional, derivative and integral gain

4. Write a MATLAB script that will use feedback control to calculate roll, pitch, yaw and thrust values to send to quadcopter

5. Test controller and program quadcopter to fly a prescribed path (you can try to think of some path that would be interesting!)
Design calculation #1: EOM for proportional vertical control

Quadcopter, mass \( m \).

We send a signal \( \tau \) (a number) to quadcopter.

Motors produce thrust \( \beta \tau \).

Calculate \( \tau \) using proportional control \( \tau = K_p (z^* - z) \).

Derive and solve EOM:

\[
\begin{align*}
F &= ma \\
-T &= ma \\
m \frac{d^2 z}{dt^2} &= T - mg \\
mg &= T = \beta K_p (z^* - z)
\end{align*}
\]

Rearrange EOM:

\[
\frac{m}{\beta K_p} \frac{d^2 z}{dt^2} + z = z^* - mg
\]

Case I: Vibration:

\[
\frac{1}{\omega_n^2} \rightarrow c
\]
SOLUTION 1:

The equation

\[ \frac{1}{\omega_n^2} \frac{d^2 x}{dt^2} + x = C \]

with initial conditions

\[ x = x_0 \quad \frac{dx}{dt} = v_0 \quad t = 0 \]

has solution

\[ x = C + X_0 \sin(\omega_n t + \phi) \]

\[ X_0 = \sqrt{(x_0 - C)^2 + \frac{v_0^2}{\omega_n^2}} \quad \phi = \tan^{-1}\left( \frac{(x_0 - C)\omega_n}{v_0} \right) \]

or, equivalently

\[ x(t) = C + (x_0 - C)\cos \omega_n t + \frac{v_0}{\omega_n} \sin \omega_n t \]

\[ z(t) = \left( \frac{x}{2} - \frac{mg}{\beta k} \right) \left( 1 - \cos \omega_n t \right) \]

Lousy controller!
Design calculation #2: EOM for proportional - derivative control

Quadcopter, mass \( m \).
We send a signal \( \tau \) (a number) to quadcopter
Motors produce thrust \( \beta \tau \)
Calculate \( \tau \) using proportional - derivative control
\( \tau = K_{PZ} (z^* - z) - K_{DZ} \frac{dz}{dt} \)
Derive EOM, find formula for \( K_{DZ} \) to get to steady state in shortest time

New EOM:
\[
\frac{1}{\omega_n^2} \left\{ \frac{m}{\beta K_{PZ}} \frac{d^2 z}{dt^2} + \frac{K_{DZ}}{K_{PZ}} \frac{dz}{dt} + z \right\} = z^* - mg \frac{8 K_p}{\beta K_p}
\]

\[
\omega_n = \sqrt{\frac{\beta K_{PZ}}{m}} \quad \frac{25}{\omega_n \sqrt{K_{PZ}}} \Rightarrow z = \frac{K_{DZ}}{2} \sqrt{\frac{\beta K_{PZ}}{m}} - \frac{K_{DZ}}{2} \frac{18}{\sqrt{K_{PZ} m}}
\]
Expected behavior \((5 < 1)\)

To remove offset add missing thrust

\[
\tau = \tau_0 + k_{pt} (z^* - z) + k_{dz} \frac{dz}{dt}
\]

\[
\tau_0 = \frac{mg}{\varepsilon} - \text{number}
\]
Choosing $k_{p2}$ and $k_{d2}$

1. Choose period of 4 s (allows Kinect to keep up with motion)

$$\omega_n = \frac{2\pi}{4} \rightarrow \text{Solve for } k_{p2}$$

2. Choose $k_{d2}$ to give close to critical damping or $\zeta = 0.8$
Design calculation #4: 3D motion

Quadcopter, mass $m$.

Control signals: thrust $\tau$, plus roll, pitch $\theta, \phi$ (degrees) and yaw rate $\lambda$

Thrust is now:

$$
\begin{bmatrix}
I_x \\
I_y \\
I_z \\
\end{bmatrix} = 
\begin{bmatrix}
\sin \theta \sin \psi + \cos \theta \sin \phi \cos \psi \\
\cos \theta \sin \phi \sin \psi - \sin \theta \cos \psi \\
\cos \theta \cos \phi
\end{bmatrix} \beta \tau
$$

Derive EOM

In MATLAB solve for $[x, y, z, v_x, v_y, v_z, \psi] = \mathbf{w}$

Need formula for $\frac{d\mathbf{w}}{dt}$ for ode45

$$
\begin{align*}
V_x &= \frac{dx}{dt} \\
V_y &= \frac{dy}{dt} \\
&\text{etc}
\end{align*}
$$
\[ F = ma \quad \Rightarrow \quad m \frac{dV_x}{dt} = T_x \quad \text{and} \quad m \frac{dV_z}{dt} = T_z - mg \]

\[ \frac{dy}{dt} = \lambda \quad \text{Set by controller} \]

\[
\begin{bmatrix}
\dot{x} \\
\dot{y} \\
\dot{z} \\
\dot{V}_x \\
\dot{V}_y \\
\dot{V}_z \\
\dot{\psi}
\end{bmatrix} = \begin{bmatrix}
V_x \\
V_y \\
V_z \\
\frac{T_x}{m} \\
\frac{T_y}{m} \\
\frac{T_z}{m} - g \\
\lambda
\end{bmatrix}
\]

Controller must specify \([\tau, \Theta, \phi, \lambda]\) and \(k_{xy}\).

Get \(k_{p_3}, k_{D_3}\) using from hand calcs or Matlab.

Get \(k_{I_2}\) from Matlab.
Design calculation #5: 3D P-D controller

\[
\tau = \tau_0 + K_{pZ}(z^* - z) - K_{dZ}v_z \\
\theta = -C_y\cos{\psi} + C_y\sin{\psi} \\
\phi = +C_x\cos{\psi} + C_y\sin{\psi} \\
C_x = K_{pXT}(x^* - x) - K_{dXT}v_x \\
C_y = K_{pYT}(y^* - y) - K_{dYT}v_y \\
\lambda = K_{p\psi}(\psi^* - \psi)
\]

Implement in MATLAB, tune \( K_p \) and \( K_z \) for horizontal/vertical motion

Follow instructions on handout

Optimal: Add integral control
To remove offset $e_z = \int_0^t (z^* - z) \, dt$

Change controller to

$$C = C_0 + K_p (z^* - z) - K_d \frac{dz}{dt} + K_i \int_0^t e_z \, dt$$

PID controller

Corrects for offset
Design calculation #6: Integral control

Need to correct systematic offset from desired position

Modified controller:

\[
\begin{align*}
\dot{e}_x &= \int_0^t (x^* - x) \, dt \\
\dot{e}_y &= \int_0^t (y^* - y) \, dt \\
\dot{e}_z &= \int_0^t (z^* - z) \, dt \\
\tau &= \tau_0 + K_{pZ} (z^* - z) - K_{DZ} \dot{z} + K_{iz} e_z \\
\theta &= -C_x \cos \psi + C_x \sin \psi \\
\phi &= +C_x \cos \psi + C_y \sin \psi \\
C_x &= K_{pXY} (x^* - x) - K_{DXY} \dot{v} + K_{ixy} e_x \\
C_y &= K_{pXY} (y^* - y) - K_{DXY} \dot{v} + K_{iy} e_y \\
\lambda &= K_{p\psi} (\psi^* - \psi)
\end{align*}
\]

Add this to MATLAB

Add \( e_x, e_y, e_z \) to unknowns in ODE 45

\[
\frac{d\dot{e}_z}{dt} = z^* - z \\
\text{etc}
\]