**Project Update #3**
**Jared Porter, Ethan Thompson, & Mary Thompson**
Draft of final report

Closed loop approach:

## Closed Loop

### Interactive PID

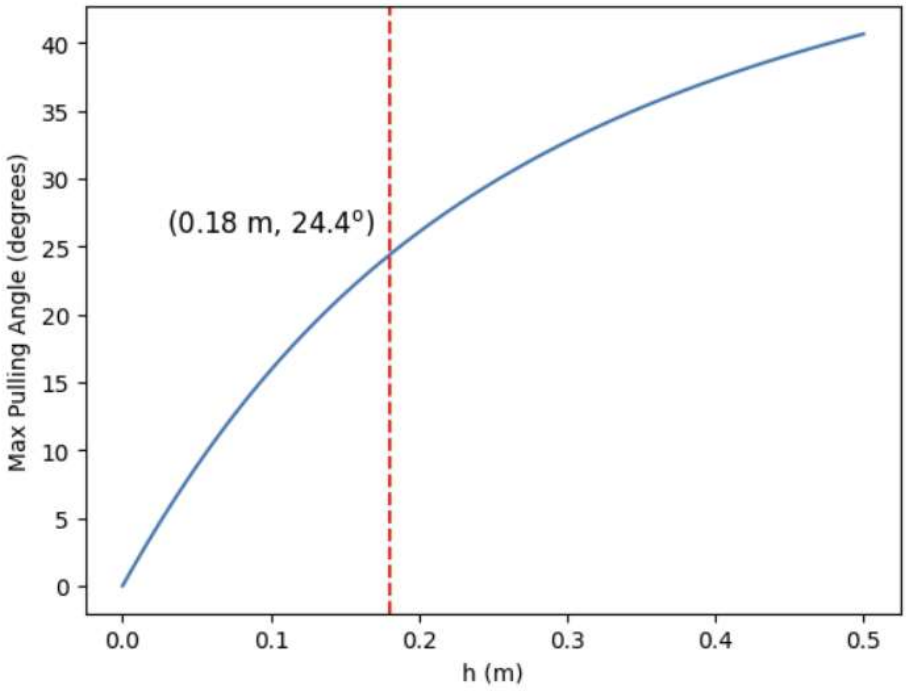
[ ]:
```
r_lever = 21/100 # m
motorSpeed = 120 # rpm
thetaMax = 30    # degrees

# Calculate max pulling angle
def maxPullAngle(h):
    return 90 - thetaMax - np.arctan(r_lever/(h + r_lever*np.tan(thetaMax*np.pi/180)))*180/np.pi

hs = np.linspace(0, 0.5, 100)
plt.plot(hs, maxPullAngle(hs))
plt.xlabel('h (m)')
plt.ylabel('Max Pulling Angle (degrees)')
h_actual = 0.18
angleMaxActual = fsolve(lambda ang: maxPullAngle(h_actual) - ang, 10)[0]
plt.text(0.95*h_actual, 1.05*angleMaxActual, rf'({h_actual:.3g} m, {angleMaxActual:.3g}$^{{\text{{o}}}}$)', fontsize=12,
         ha = 'right', va = 'bottom')
plt.axvline(h_actual, color='r', linestyle='--')

# Calculate Max dTheta
spoolD = 15/1000   # m
spoolSpeed = motorSpeed / 60 * np.pi * spoolD # m/s
maxDTheta = spoolSpeed/(r_lever*2*np.pi)*360 # degrees/s
print(f"max dTheta = {maxDTheta:.2g} deg/s")
```

```
max dTheta = 25 deg/s
```




In [13]:

```
# Process Parameters
maxTime = 100
maxTheta = 25     # deg
maxDTheta = 25    # deg/s
nPts = 1000
m = 5/1000 # kg
g = 9.81 # m/s^2
b = 1.5e-3 # Ns/m (damping coefficient)
```

```
# Process Variable Arrays
times = np.linspace(0, maxTime, nPts)
dTheta = np.zeros(len(times)) # deg/s
theta = np.zeros(len(times))  # deg
pos = np.zeros(len(times))
vel = np.zeros(len(times))
SP = np.zeros(len(times))
Ps = np.zeros(len(times))
Is = np.zeros(len(times))
Ds = np.zeros(len(times))

# Setpoint changes
SP[50:] = 10/100
SP[300:] = -10/100
SP[600:] = 0/100

# Lever differential equation
def lever(state, t):
    thetaVal, dThetaVal = state
    return dThetaVal, 0

# Ball differential equation
def ball(state, t, thetaVal):
    x, dx = state
    return dx, m*g*np.sin(thetaVal) - b*dx/m

def simulate(Kc, KI, KD):
    # Configure PID controller
    pid = pc.PID(Kc, KI, KD, dt=times[1]-times[0], outputLimits=[-maxDTheta, maxDTheta])

    # Reset process variables
    theta[:] = 0
    pos[:] = 0
    vel[:] = 0

    # Simulate
    for i in range(len(times)-1):
        tArr = [times[i], times[i+1]]
        # Set the setpoint and get PID output
        pid.setpoint = SP[i]
        if theta[i] > maxTheta or theta[i] < -maxTheta:
```

```
    plt.plot(times, SP*100, label='Setpoint (cm)')
    plt.ylabel('Position (cm)')
    plt.legend()
    plt.subplot(4, 1, 4)
    plt.plot(times, Ps, label='P')
    plt.plot(times, Is, label='I')
    plt.plot(times, Ds, label='D')
    plt.ylabel('PID Terms')
    plt.xlabel('Time (s)')
    plt.legend()
    plt.tight_layout()
    plt.show();

# Ziegler-Nichols Rules
Ku = 220/0.6 # Ultimate gain. Point where system oscillates with KI = KD = 0
Tu = 10  # s, oscillation period
Kc = 0.6*Ku
KI = 2*Kc/Tu
KD = Kc*Tu
print(Kc, KI, KD)

# Create interactive sliders
interact(simulate, Kc=widgets.FloatSlider(value=96, min=0.0, max=1000.0, step=0.1),
         KI=widgets.FloatSlider(value=0.0, min=0.0, max=1000.0, step=0.1),
         KD=widgets.FloatSlider(value=580, min=0.0, max=5000.0, step=0.01))
# Manually optimized: 400, 200, 3000
# Optimized values:   96,  0.0, 580
```

```
220.0 44.0 2200.0
interactive(children=(FloatSlider(value=96.0, description='Kc', max=1000.0), FloatSlider(value=0.0, descriptio…
  <function __main__.simulate(Kc, KI, KD)>
```

# Abstract:

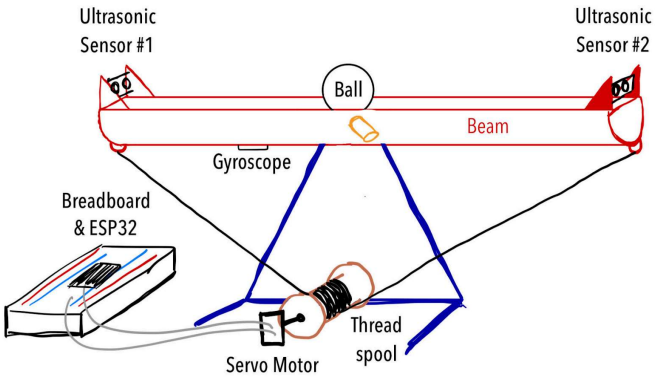
The objective of this report is to stabilize a ball along a beam using data from an ultrasonic sensor and a PID controller. The set point along the beam should be adjustable and the controller should restabilize when exposed to disturbances like flicking the ball. The actuator will be the servo motor controlling the string connected to the edges of the beam. The measurement will be the position of the ball on the beam. The controller is the PID controller created by the code

# Introduction:

**Literature Review**

The ball-and-beam system is a foundational example in control systems, illustrating feedback principles through the challenge of maintaining the ball's position on a pivoted beam. Studies, such as Găşpăresc (2014), have demonstrated the effectiveness of Parallax Ping ultrasonic sensors in this application. PID-based ball-and-beam controllers are well-documented (e.g., Saad, 2017), the novelty of this project, however, lies in the use of spools and thread, driven by a servo motor, to control the beam's torque, offering a unique mechanical approach to actuation.

# Theory:




Primary goal: maintain the ball at a desired position, stabilizing the system despite disturbances

The electrical circuit consists of

- Esp32
- MG90s micro servo motor
- Ultrasonic sensors
- gyroscope

Mechanical parts

- 3D printed trough (42 cm)
- 3D printed base
- 3D printed dowel connecting base and trough
- Thread & spool

A circuit diagram is available upon request (hehe)

Key variables:

There are several variables at play in this simple ball and beam case study, and they can largely be divided into three categories: constants, the process variable, and the controlled variable. The constants include things like the length of the beam and the mass and size of the ball. The measured process variable is the position of the ball, which can be used to calculate the velocity of the ball if taken in two instances. Finally, the controller output is functionally the angle of the beam, which is indirectly controlled through direct control of the angular velocity and run time of a servo motor.

Limitations

- Due to the matching frequency of the opposing ultrasonic sensors, we can only run one ultrasonic sensor at the same time, therefore they alternate taking readings every 1/20 of a second
- There is a max angle the beam can reach when correcting, additionally we set an allowed max angle of 30 degrees.
- There is noise in the ultrasonic sensor due to surroundings and measurement errors therefore the recorded location of the ball might not always be fully accurate
- Latency at high speeds, will the controller be able to respond quickly enough?
- Will the servo motor be able to create high enough torque
- The curvature of the ball prevents all ultrasonic readings from returning to the sensor, decreasing accuracy

Challenges:

- Overshoot
- Settling time
- delay

# Discussion/Recommendations:

Stability analysis:
Model parameters:
controller/optimizer tuning with justification for best performance:

# Conclusion:

Consistent with objective in the introduction

Conclusions should include the main results of the study, recommendations, and any future work that could be pursued. The conclusion should address the objective

“A mathematical model of the ball and beam system was developed using physical and electrical laws. A simplified mathematical model was derived through system parameters. The controller parameters values (Kp, Ki and Kd) were obtained by using manual tuning method from practical model so as to perform best system response. From experimental results, it is found that the best controller parameters which gave the best response of the system are: Kp=4, Ki=1 and Kd=2. The accuracy of the system is tested by adjusting the position of the ball at three different points and it found that the accuracy doesn’t affected by changing the set point.”
https://www.academia.edu/33080054/Design_and_Implementation_of_Ball_and_Beam_System_using_PID_Controller