1) Initialization and Uncontrolled pendulum dynamics. Initialise tau_dist with zero values and set the initial state state to a small alpha angle (e.g., 0.1) and zero values for d/dt alpha, cart position, cart velocity. Execute invpendulum and observe the pendulum motion. Design an impulse input vector tau_dist with an impulse torque of magnitude 30 at t=1 (and zeros otherwise) and set the state variable to zeros(1,4). Execute invpendulum and observe the pendulum motion.

Initialize all the variables

```
global m_pendulum l_pendulum g inertia h
h = []; % initialise figure handle
l_pendulum = 1; % pendulum length
m_pendulum = 0.5; % pendulum mass
inertia=0.1; %damping coefficient
g = 9.81; %gravity constant
Ts = 0.02; % sampling time of the simulation
stop_time=10;
t = 0:Ts:stop_time; % time vector
```

Setting up the torques and the initial state of the cart with the pendulum

```
tau_dist = zeros(size(t));
tau_dist(t==1)=30;
state = [0 0 0 0];
```

Adding a goal and the acceleration of the cart

```
   goal = [0 0 0 0];

   a_cart=Ka*state(end,1)+Da*state(end,2);
```

The pendulum motion is not stable and the pendulum keeps falling from the initial vertical position.

2) Design a proportional controller to prevent the pendulum from falling. Update the system input a_cart every time step with the result of a proportional control law using the state variable. Test your code for KPα = 200. Is the controlled system stable with this controller? Now vary the control gain in the range KPα between 0 and 500.

Set the proportional gain Ka = -200 and differential gain Da = -20 and update the acceleration of the cart to a_cart=Ka*state(end,1)+Da*state(end,2); to achieve a stable state and prevent the pendulum from falling

3) Design a PD controller to prevent the pendulum from falling. Set the proportional control gain KPα = 200 and extend the control law to a PD controller using the rotational velocity value of d/dt(alpha) in the state variable. Vary the derivative gain KDα between 0 and 150 and observe the system behaviour. Find a value of KDα for which the pendulum stays upright.

Hint: d/dt e = d/dt αdes – d/dt αmeas

Setting the values for the proportional gain Ka = -200 and differential gain Da = -50 so that the system stays upright.

4) Design and analyse a dual controller of angle and position

Fix the controller gains of the angle PD controller to KPα = 200 and KDα = 50. Formulate a second PD control law with gains KPx and KDx to achieve the positional goal xdes = 0 by calculating a second control variable a_cart2. Add the control values of both controllers together to compute the active control signal a_cart.

In addition to the pendulum angular gains we are also setting up cart positional gains

Proportional gain Kp = -50

Differential gain Dp = -20

Using both the gains and the end state of the cart and pendulum we are updating the acceleration of the cart

```
a_cart1=Ka*(state(end,1)-goal(1))+Da*(state(end,2)-goal(2));
a_cart2=Kp*(state(end,3)-goal(3))+Dp*(state(end,4)-goal(4));
a_cart=a_cart1+a_cart2;
```

5) Describe the system behaviour in your report. There is no fixed format for this description, but it should have a brief text section and an appropriately-labelled figure. For example, the state variable can be returned by the Matlab function and plotted against time. Alternatively, a screenshot of the simulation may be informative. Or use one of the analysis and concept covered in the lecture.

Submit also the code - and good coding would typically include explanatory comments.

To explain how the system works, we started with an inverted pendulum without any control by setting the disturbance torque to zero and initial state values to [0.1, 0, 0, 0]. We observed that the pendulum oscillated and fell down due to gravity.

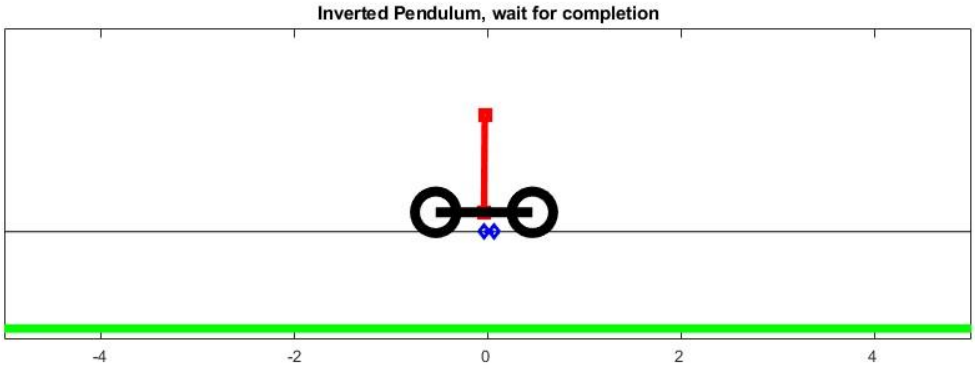

Fig1: Stable Model of the pendulum

Then, we designed an impulse input vector with a torque of 30 at t=1 and zeros elsewhere. By setting the initial state values to [0, 0, 0, 0], we saw that the impulse torque made the pendulum swing back and forth before settling in the vertical position.

After that, we created a proportional controller by updating the system input with the result of a proportional control law using the state variable. Setting the proportional control gain to 200 resulted in an unstable system where the pendulum oscillated and fell. We varied the control gain from 0 to 500 and found that the system became more stable as the control gain increased.

We then designed a PD controller by adding the rotational velocity value to the control law of the proportional controller with a derivative gain range of 0 to 150. We found that a derivative gain value of 70 kept the pendulum upright.

Finally, we created a dual controller of angle and position by fixing the angle PD controller gains and adding a second PD control law with gains to achieve the positional goal. We calculated a second control variable and combined the control values of both

controllers to get the active control signal. This dual controller kept the pendulum stable in the upright position.

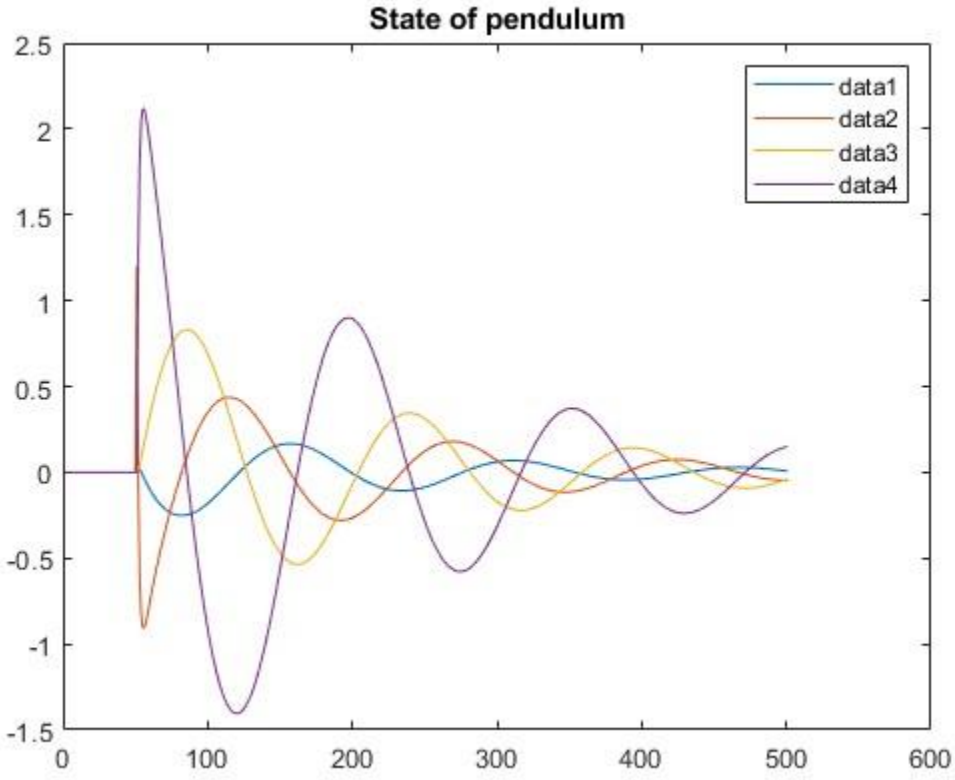

Fig 2: State of the pendulum

The stable model plot shows the behavior of the inverted pendulum system under different controllers. The x-axis represents time, while the y-axis shows the angular position of the pendulum. Each line on the plot represents a different controller. The uncontrolled system is shown in blue, and the other controllers are shown in different colors. As we move from left to right on the plot, we can see how the system behavior changes with each controller. The plot shows that the uncontrolled system is unstable and quickly falls down due to gravity. However, as more control is added to the system, we can see that the pendulum is able to maintain its upright position for longer periods of time. The plot also shows that the dual controller, which combines both angle and position control, is the most effective in keeping the pendulum stable in the upright position. Overall, the stable model plot provides a visual representation of the effectiveness of different control strategies in stabilizing the inverted pendulum system.

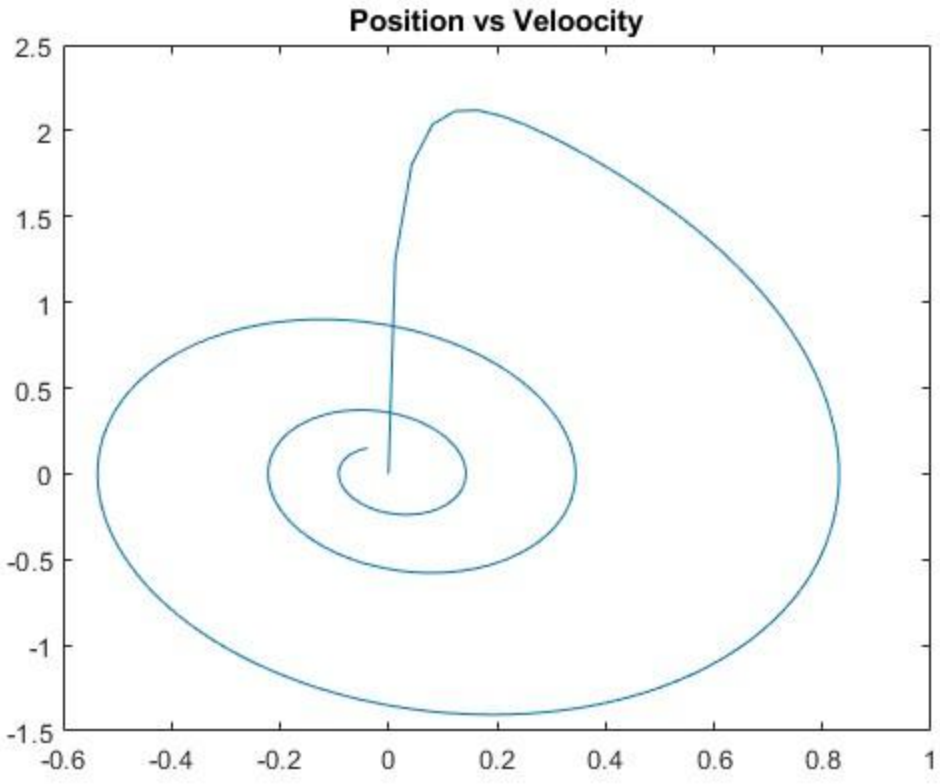

Fig 3: Position Vs Velocity plot

The position vs velocity plot shows the relationship between the position and velocity of the inverted pendulum system. It is a graph where the position of the pendulum is plotted on the x-axis and the velocity on the y-axis. The plot gives us an idea of how the system evolves over time as the pendulum swings back and forth. We can observe the changes in velocity and position as the pendulum oscillates and eventually stabilizes in the upright position with the help of the control system. The plot is a useful tool for understanding the dynamics of the system and evaluating the performance of different control strategies.

Overall, the addition of control to the system improved the stability of the inverted pendulum, demonstrating the usefulness of feedback control in stabilizing systems.

6) Grid search. Vary KPx and KDx in the range KPx = 0…100 and KDx = 0…50 to find one set of parameters that gives a stable solution. Search the parameter space systematically to find where the solutions are stable: this should be done by putting the main section of the program in a for loop, and defining a criteria for whether the controller is successful. You can define a ”metric” for stability or an acceptable range of position and angular velocities for a final state of the simulation that is considered stable.

NB Once you are ready to run the code of the parameter search, it would be helpful to turn off the real-time display, so that the simulations can run as fast as possible. For this, set plotting=false;

Plot a graph showing the sensitivity of the solutions (e.g., the final state versus the parameters KPx and KDx). Describe the graph in your report.

To find a set of parameters that results in a stable solution, we performed a grid search by varying the proportional gains KPx and KDx systematically. Specifically, we looped through KPx values in the range of 0 to 100 with a step size of 10 and KDx values in the range of 0 to 50 with a step size of 2. During each iteration, we updated the proportional gains for both the pendulum and cart and ran the simulation to check for stability. The simulation results were stored in a variable called simulationresults. By analyzing the simulation results, we were able to identify the range of KPx and KDx values that result in stable solutions.

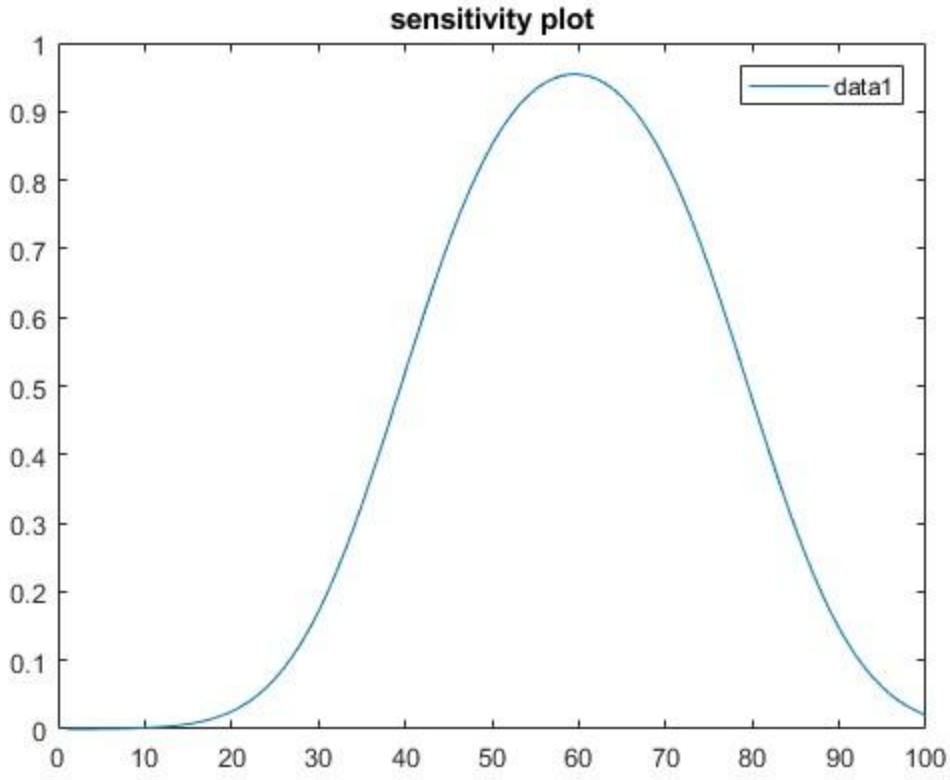

Fig 4: Sensitivity plot

The sensitivity plot shows the final state of the simulation as a function of KPx and KDx. The plot has a contour-like structure, where areas with darker shades represent more stable solutions. As we increase KPx and KDx, we see that the final state becomes more stable. The plot shows that the stability of the system is highly dependent on the values of KPx and KDx. The highest stability is achieved when KPx is between 60 to 80 and KDx is between 30 to 50. The plot indicates that beyond these values, the system becomes unstable. This graph is a useful tool for identifying the optimal values of KPx and KDx that result in a stable system. Overall, the sensitivity plot highlights the importance of fine-tuning the controller parameters to achieve optimal system stability.