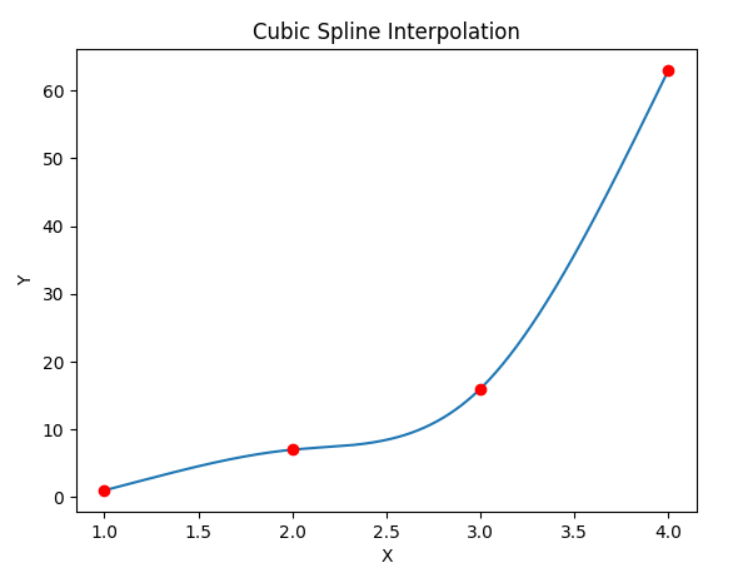
**PPC MODULE**

* **Aryan Kayande**

Checkpoint 1: [Birth\_Rate\_India\_Nepal\_Bhutan.ipynb](https://colab.research.google.com/drive/1IRe0-pJjeN6uT6wpmZuVa1fUVrS6S-RQ?usp=sharing)

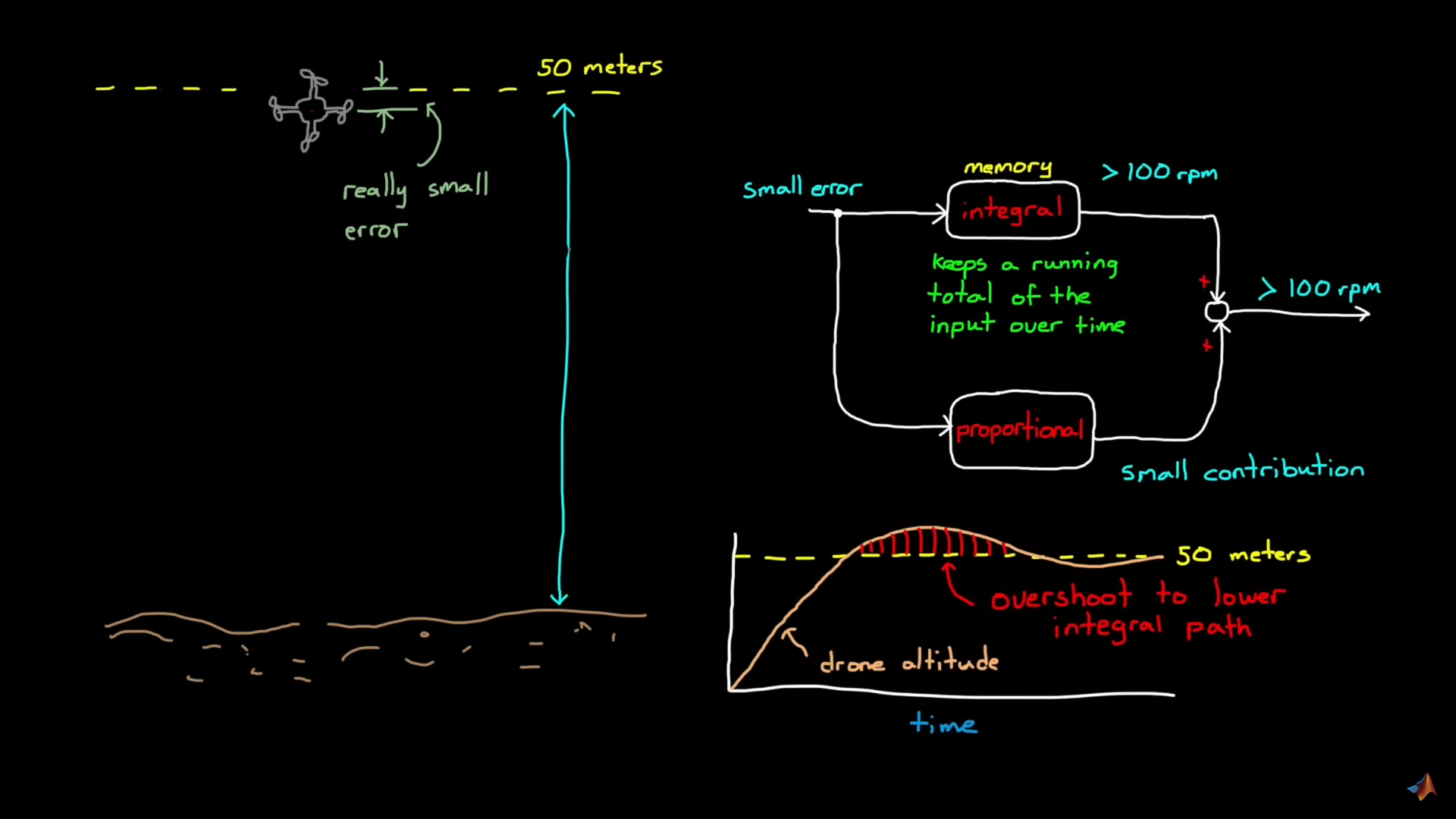
Checkpoint 2: [Spline Interpolation.ipynb](https://colab.research.google.com/drive/1foIYt7T2lu4ySCKvJNqepxB5OV0bxM_9?usp=sharing)



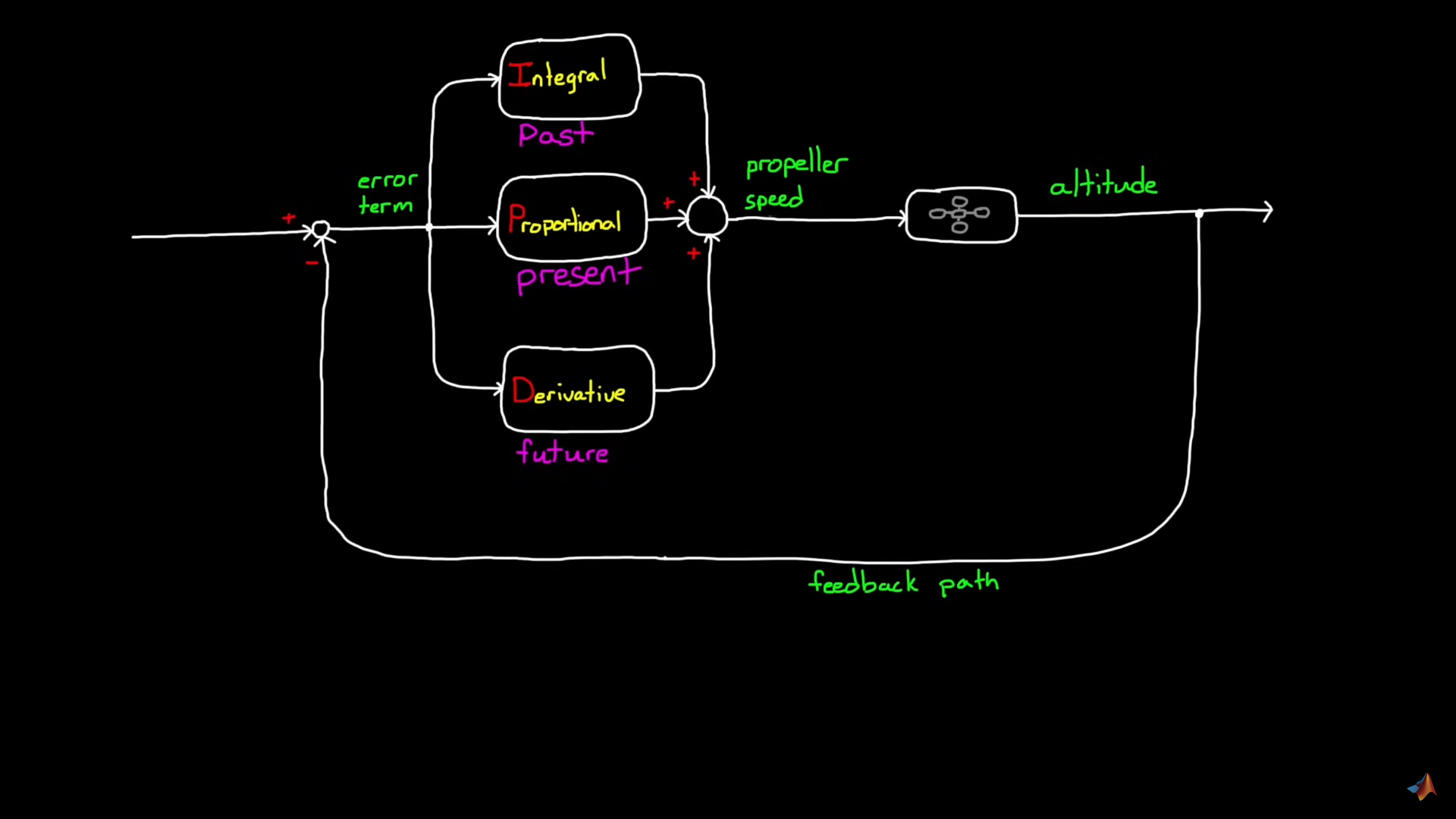
Knew about interpolation:) Watched the videos and implemented them in the assignment.

Checkpoint 3: [PID.ipynb](https://colab.research.google.com/drive/1pwbHqaWzEdEdtwG58u7uI0V6gd06oSE0?usp=sharing)

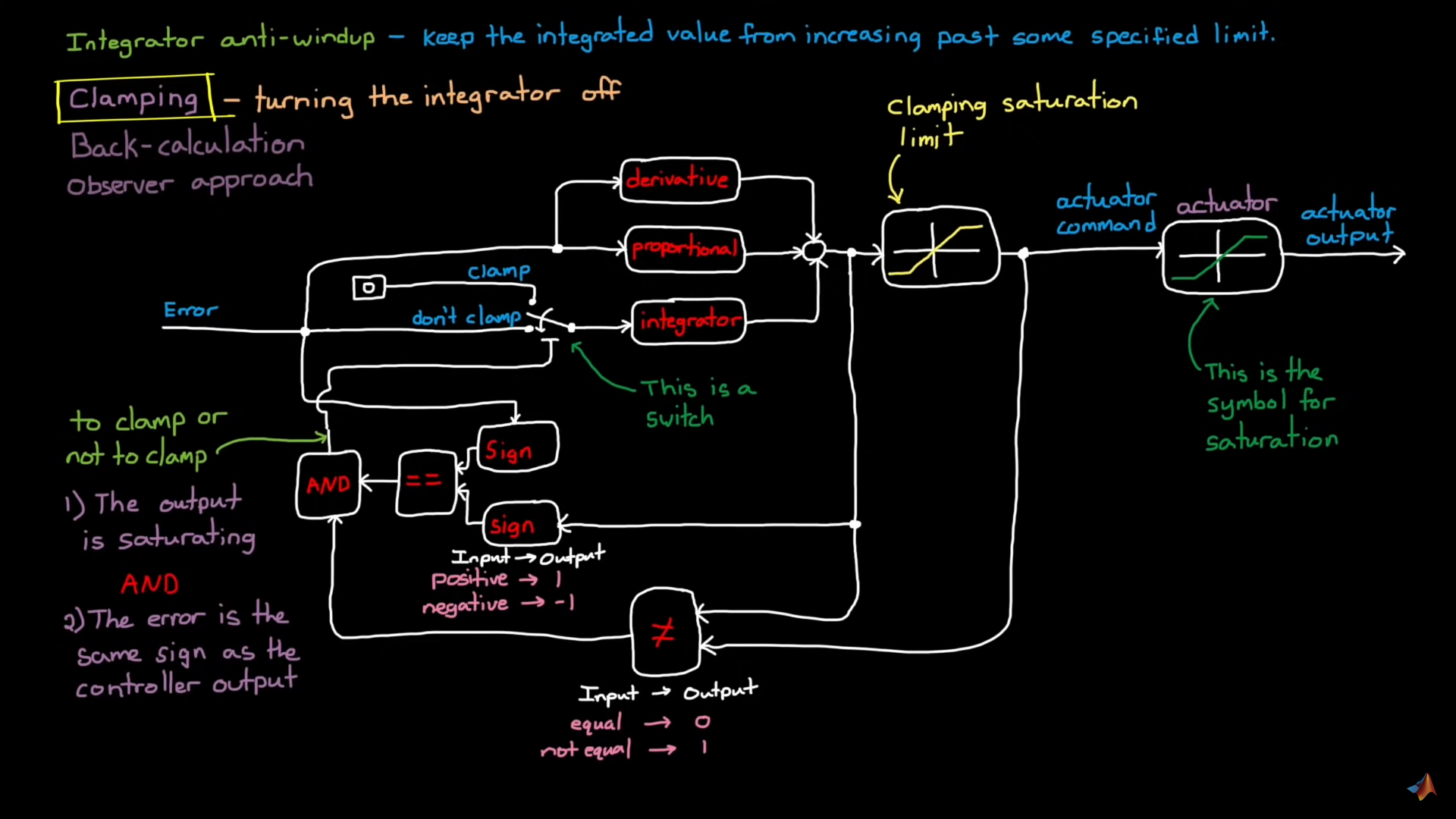
PID stands for Proportional Integral Derivative. It is a control feedback mechanism which provides precise control over a system’s behaviour. Suppose we want to control a particular attribute, we first set a target value, then based on how far we are currently from the target value, we increment/decrement the parameter accordingly. Now, there are several ways in which the change can be related to the error. The first thing that we do is change = K \* error, where K is a constant. In this case,we get an oscillating graph. Thus, to rectify this, we add an integral term, which keeps track of the total input over time. This helps in reaching the equilibrium stage, but the path taken might not be ideal, as it suffers from overshooting in case of high Ki. so, we add one more term called the derivative term.



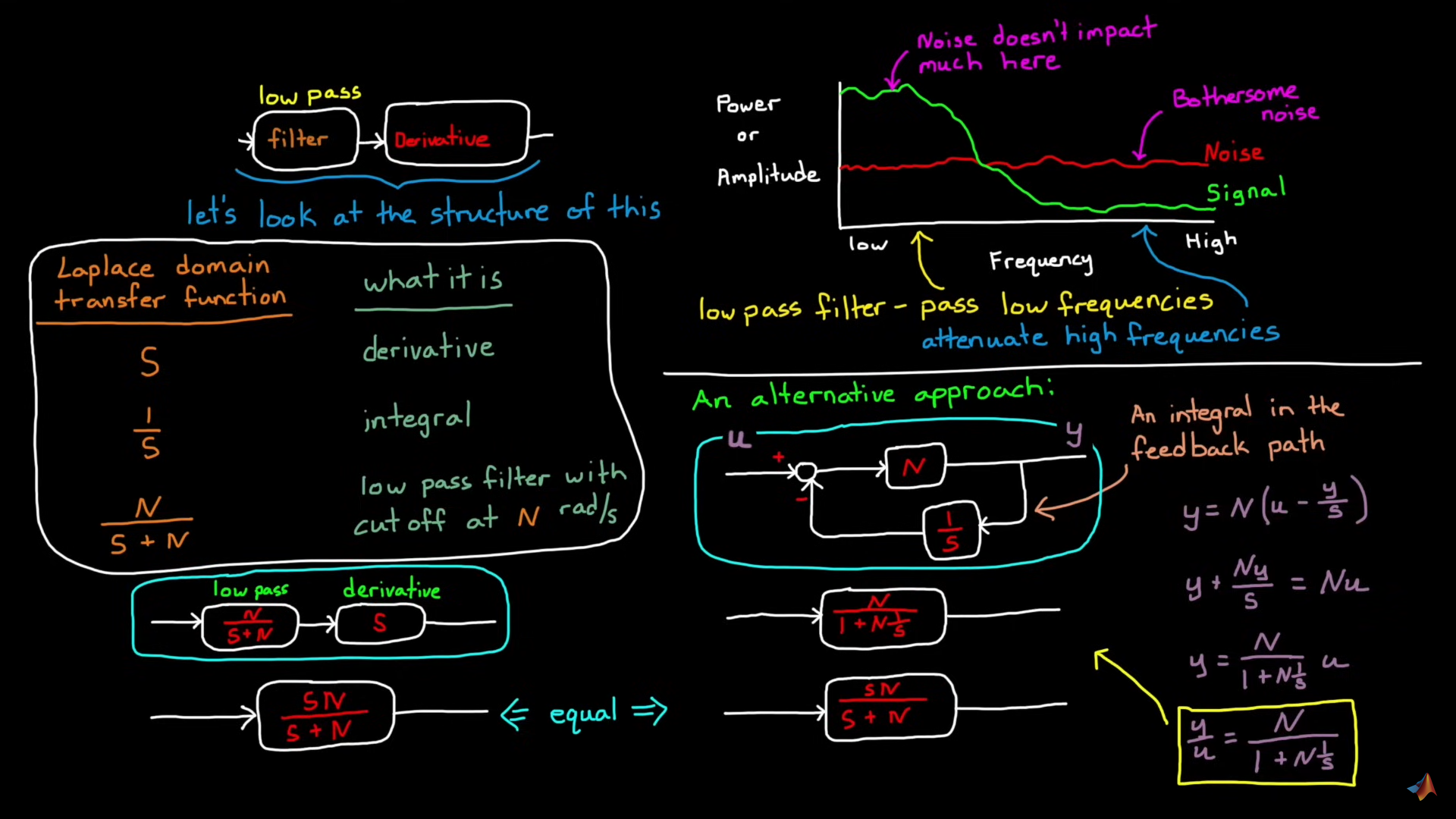
This derivative term adds damping in the system by anticipating future changes in the error. Thus, when the signal changes rapidly, the derivative term applies a damping term proportional to the error which prevents overshooting. This is how the entire PID system works:



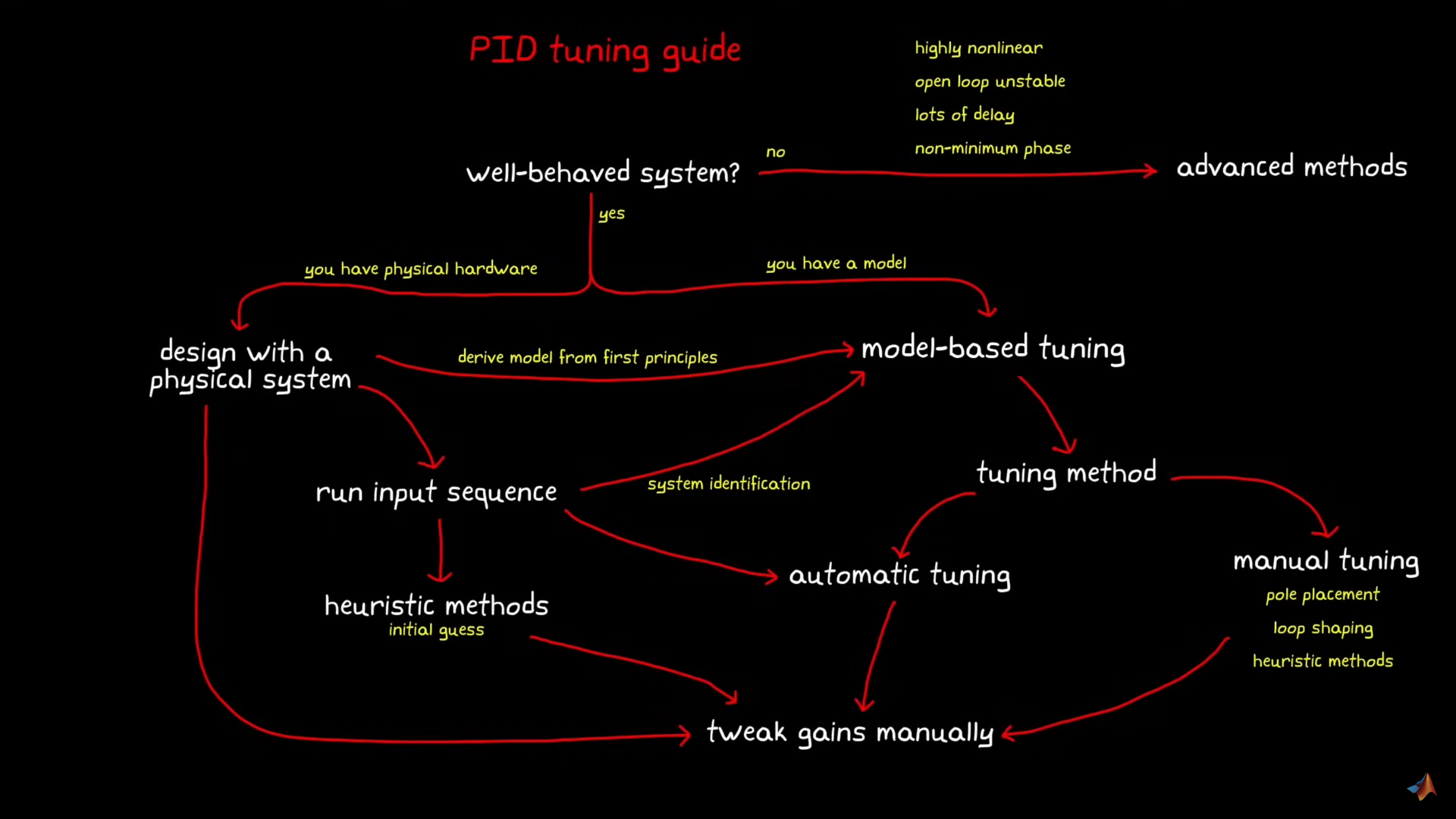
Anti - Windup: PIDs are used in real life applications like actuators, which have physical limitations. For example, a particular propeller of a drone can rotate as fast as 50 rpm only. Thus, our PID system suggests that the propeller should rotate at a faster rate than its limit, which cannot happen in real life. This is called windup. Several anti windup methods are used in PID controllers like Clamping, Back-calculation, observer approach. In Clamping, if the change term leads to the signal beyond the limit, then we turn off the integrator. This is the basic functioning of clamping:



Another problem that occurs commonly in PID controllers is noise. The sensors sometimes give noise, which makes it difficult to control it. For this issue, we apply some filters before passing it onto the derivative term. One of the common filters is the low pass filter, which passes low frequencies by filtering out the high frequency noise. By doing this, it prevents the derivative term from reacting excessively to sudden fluctuations.



One of the major things in PID is tuning, which is basically the process of adjusting the Ki, Kp and Kd to achieve desired performance. In the assignments given to us, we did manual tuning, that is we changed the parameters manually and observed the results every time and adjusted the parameters accordingly. Another common method is the Ziegler-Nichols method. In this method, we first set Ki, Kp, Kd to 0 and then keep on observing the system's response as we change these parameters. First, we increase Kp, until the system begins to oscillate steadily and note the critical value and period of oscillation. Then there are some methods like classic or the overshoot method, by which we calculate the values of other parameters using predefined formulas. In all the methods, finally we tune the values manually to check the corresponding results as follows:



**Final Assignment:** -

**Optimisation:** Basically choosing inputs that will give best possible outputs. For example, finding the value of x for which f(x) is maximum. Several of the terms used in optimisation algorithms are: -

**Objective Function**: The value we want to optimise. Example: if we want to maximise the area of a rectangle, the objective function is area.

**Decision variables**: the parameters optimiser can change. In the above example, length and breadth of the rectangle are the decision variables (also called as manipulated variables). Note, as the number of decision variables increase, it gets more and more difficult to optimise.

**Gradients**: describe the slope of a function. In 2D: df/dx.

**Constraints**: describes where the optimisers can not go. This is particularly important in real-life scenarios like using actuators or propellers, where the objective function can not go beyond a limit.

**Derivatives:** Finds the direction of the steepest descent. Also known as gradient in multiple dimensions. We then determine the step size which is the amount by which we proceed in the gradient direction.

**Gradient Free Algorithms:** These algorithms don’t use derivatives for optimisation. Examples of this are: exhaustive search, genetic algorithm, particle swarm etc.

Assignment Link: [Spline Interpolation.ipynb](https://colab.research.google.com/drive/1foIYt7T2lu4ySCKvJNqepxB5OV0bxM_9?usp=sharing)

Note : - Link same as Spline Interpolation, though I added a few things at the end.