

 Topics

 My Posts

 More

 Categories

 Lunar Scout (LS)

Guidelines

ChatbotDocs

 All categories

 Tags

 task-0

 task-1

 task-2

 other

 task-4

 All tags

Skip to main content



# [PRACTICAL] Implementing control systems on Hardware!

Krutarth  e-Yantra Staff

Jan 30

## Aim:

Implementing control algorithms like PID or LQR on a simulator like MATLAB, Simulink, CoppeliaSim, Gazebo etc. is too much of an ideal scenario. A person with average theoretical understanding can manage to achieve expected results with relative ease.

But implementing the same on hardware, dealing with non ideal conditions, with Varying power sources, messy electronics, on discrete time systems like microcontrollers, noisy sensors, and non-ideal actuators(some of them aren't even rated & don't have a proper datasheet). ... and most importantly **our lack of understanding in the art of "Observing the physics acting on a real hardware robot"**.

**In summary, we don't know how to DESIGN, DEBUG & TUNE "hardware control systems" !!!**

This material aims to help LEARNING some of the most Underrated skills that are essential for a control systems engineer. *Specifically for someone working on Self-Balancing robots !*

## 0. Miscellaneous : Tips before you start ahead

### 1. Empirical (Observation based) PID TUNING

Note: the gain values might be of different magnitude and order for your physical system. This is an analogy to demonstrate the basic approach.

Do not hesitate to take a look at the full playlist to understand better.

## And ...the observations might look like the following

(Note: this is for a single PID - which can control only the tilt angle and it's velocity, but not linear or angular displacement. It will require another one for that.)

## 2. Advanced factors that affect PID implementation on a discrete system (microcontroller)

### Continuous vs Discrete time control - what is the difference?

There are many advanced factors that help the PID implementation on a microcontroller get better. Following BLOG is a good start for these:

<http://brettbeauregard.com/blog/2011/04/improving-the-beginners-pid-introduction/>

### How to relate with LQR now?

LQR is like a PD(PID without the "I") controller algorithm. So the way of observing the system to know what should be tuned... is the same. Difference is that it tries to find optimal gain combinations, where one needs to tune the weights of "Q - matrix" diagonal elements.

- Increasing the magnitude of diagonal **element of Q matrix corresponding to tilt angle state - acts like adding proportional effect.**
- Increasing the magnitude of diagonal **element of Q matrix corresponding to tilt angular velocity state - acts like adding derivative effect.**

Same for yaw angle & angular velocity states. Which makes it like two PD controllers acting.

An important thing you should notice, is that ***even when you change one weight of Q-matrix, almost all the gains in final K-matrix will change!***

That is because the LQR is based on the

[Skip to main content](#)

dynamics of the system that we modeled mathematically, and it knows that increasing one state's gain might also affect all the other states. Specifically in systems with coupled state dynamics.

Also, LQR gives optimal gains based on priority sequence and the relative difference in magnitude of weights of Q-matrix matters very much.

- If *tilt angle control is fundamental for balancing, then the states corresponding to that need highest priority.*
- Velocity weights for an angle(tilt/yaw) decides **how fast/slow you want to allow the robot to change it's angular velocity.**

More the weight of a state, more will it try to "Regulate" system around the desired equilibrium point values of that corresponding state(angular position or velocity).  
From the name : **Linear Quadratic "REGULATOR"**

## Multiple PIDs

These are necessary for complete control (including yaw) of the system.

Now two common variations can be used:

### 1. **Parallel:**

This case is more intuitive to tune and implement as it's just summing up both PID outcomes.

### 2. **Serial/Cascaded:**

This needs a little thought on how the system is coupled & which state depends on the another & how.

- It requires you to choose which is the primary state and which is the secondary state. In other words dependent & independent state.

Specific to the Lunar Scout Bike, choose among yaw and tilt angle for this.

[Skip to main content](#)

Refer back to **“block diagram representations”** & **“signal flow graphs”** basics to draw and understand better how the resulting gains of system are calculated from the variation used in the closed loop system.

The **WIKIPEDIA** page of **“PID”** is a must read for anyone who is implementing PID control. There are tuning methods, effect of each terms, variations, modifications, limitations and how to overcome them.

Even when you are wondering how your self balancing bike behaves weirdly 🤔 ...Asking questions, and asking the right questions is the key to understand! 😂




- [🔗 \[URGENT\] New items on the FORUM](#)
- [🔗 \[Announcement\] Open Door Session](#)

Closed on Jan 30



Listed on Jan 30

Pinned on Feb 1

Related Topics

Topic	Replies	Views	Activity
 <a href="#">Lunar Scout: Task-4</a>	1	970	<a href="#">Nov 2023</a>
 <a href="#">[PRACTICAL] Recorded Session</a>	2	159	<a href="#">Feb 10</a>
 <a href="#">Lunar Scout: Task 1</a>	3	9.6k	<a href="#">Sep 2023</a>
<a href="#">Doubt Regarding control loop</a>	1	43	<a href="#">Feb 14</a>

[Skip to main content](#)

Topic	Replies	Views	Activity
  <b>Lunar Scout: Task 2</b>	<b>3</b>	<b>3.8k</b>	<b>Sep 2023</b>