Controller Assignment

A Report About Controllers And Their Implementation

Assignment

Sylvain Lott

October 2023

Assignment supervisors:

Reto E. Koenig
Morgane Kappeler

UNI FR
UNIVERSITÉ DE FRIBOURG
UNIVERSITÄT FREIBURG

Software Engineering Group
Department of Informatics
University of Fribourg (Switzerland)

Software Engineering Group

Abstract

The following report is about controllers, specifically BangBang and PID. It's main goal is to explain how those controllers works, their meaning, but also how they were implemented into an actual phone application, coded from the MIT App Inventor. It is an introduction to Process Controls.

Table of Contents

1 Introduction

This report is to be seen as an introduction to process controls, how a controller works and it's strength and weaknesses depending on the type of controller used. Here we'll focus on two type of controllers, the first one being the BangBang. It is a pretty basic and intuitive controller, but still efficient depending on the use case, although in this current application it is not optimal. Afterwards comes the PID controller, which is a more advanced version of the BangBang. The PID reprensents the main part of this report, as it is the one that needed to be correctly implemented and tuned. We'll explain and analyse it's implementation and mostly it's weaknesses, wether they are code related or values related.

In the end we're gonna have a final word about the project and discuss how it could be improved or optimized. We'll also open up possibilities to diver deeper into the subject and go further into it.

2
BangBang

2.1 Principle

The Bang-Bang controller works almost the same way as an ON/OFF switch, it is one of the simplest and most basic types of control systems used in engineering and industrial applications. It operates by toggling a system's control input between two states, based on a specified reference point.

It can be broken down into four main steps:

1. **Set error**: In this step, the controller compares the current variable value and the setpoint. The difference between the two is called the error.
2. **Action Control**: Here the controller checks wether the error acceptable or not. What is acceptable is defined manually. The range of acceptable values around the setpoint is known as the "deadband".
3. **Switch State**: If the error is not in the acceptable range, then the system should output something. In case the error is positiv, it'll output the maximum, if it is negative, it'll output the minimum.
4. **Cycling**: The controller will switch back from a state to the other until it is stabilized in the deadband.

2.2 Strength and weaknesses

The main advantage of the BangBang controller is it's simplicity, it is really easy to understand and implement: Furthermore it can be robuste enough for a lot of given situations though being this simple does lead to some weaknesses. The first one is that it will often lead to some oscillations, the system will never stop exactly on the target, but will often go further than necessary, then come back etc... This does lead to a decrease in accuracy overall.

If we want to go further, we could say that going from one extreme to the other this quick can lead to problems if applied on physical components.

3
PID

3.1 Principle

A PID controller is based on the same thing as a BangBang controller, the error (see 1). But a PID controller goes further than just two state depending on the polarity of the error. PID stands for Proportional, Integral, Derivative, meaning it takes into account not only the current error, but also the past errors and the error to come (Integral/Derivative). Let's break down those three components further:

1. **Proportional**: This is the closest part to the BangBang controller, it computes an output proportional to the error. The error is multiplied by a factor set manually. Here's how it's computed: $P = K_P \cdot E$
2. **Integral**: This part represents the "past". It is computed by multiplying the sum of the past error by a factor set manually. This helps to eliminate steady-state errors. Here's the whole formula: $I = K_I \cdot \int Edt$
3. **Derivative**: The last part helps us predict the errors to come by computing the speed of change of the current error. It is a multiplication of the derivative of the error by a factor set manually. Here's the whole formula: $D = K_D \cdot \frac{dE}{dt}$

After all those components are computed, the output is equal to the sum of them three: $U = P + I + D$. Here's a more detailed schema to further understand how it works:

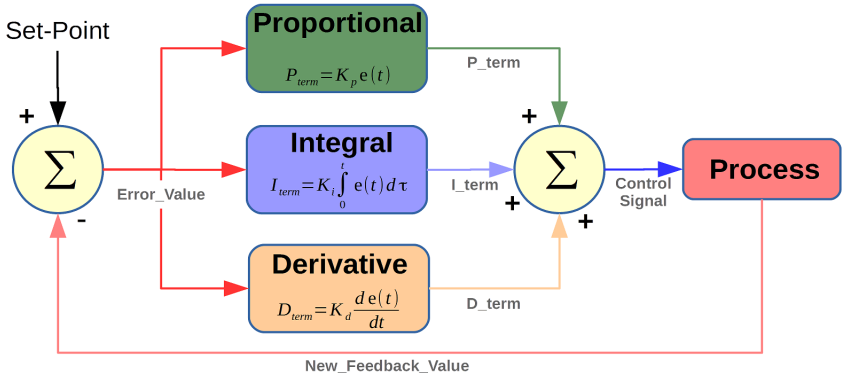

Fig. 3.1: Functioning of a PID controller.

3.2 Implementation

We'll now focus on how the implementation of the PID was done in the assignment. The "code" was done using the MIT App Inventor, which is a service that allows us to simply create phone applications without getting deep into the code. The implementation of the PID, as explained before, has several steps:

1. Compute the error
2. Compute the PID values
3. Add the PID values and return them

Here's how it was done:

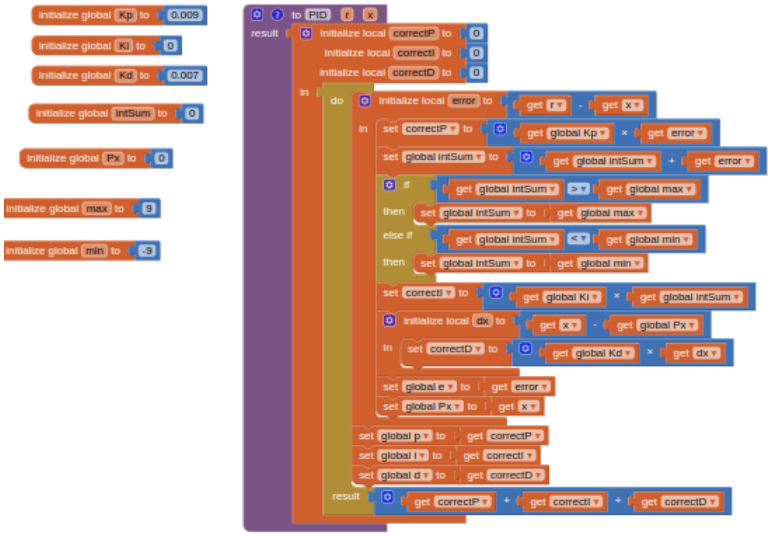

Fig. 3.2: The implementation of the PID controller.

We first declare and initialize the new PID values, they are set to 0 for now, and we compute the error $(r - x)$. Then we compute the first component, which is P. So here we set P to $K_P \cdot E$.

After comes the Intergral component, so we first need to add the current error to the sum of errors. This sum of past errors would stack infinitely if we didn't put any limit to it, that's why we need to declare an upper limit and a lower limit. If one of those is broken, we'll set the sum of the past errors to the limit it broke. Finally we can compute the Integral component this way: $I = K_I \cdot \sum E$

For the Derivative part, we first compute dx, which is the difference between the last x recorded and the actual x position. This gives us an idea of how the system is evolving at the moment. We then apply this formula: $D = K_D \cdot dx$

Lastly we update all the global values and return the added components.

3.3 Component tuning

The goal of this part is finding the right values for each of the K factors. The way to go for us is by starting with random values, see how it goes and analyse why it is (probably) not working. We then adjust the values and so on until we get a satisfying result. In this case the final chosen values were:

- K_P = 0.009
- K_I = 0
- K_D = 0.007

The first thing to notice is that those are very small values. That is because the plane wasn't suppose to move abruptly, it was supposed to always move slowly and smoothly so the passengers can stay still inside. In order to achieve this, we have to put small factors, otherwise the controller will output a very strong value and make the plane move quickly and abruptly. It is also what happens in the BangBang controller, it outputs either full in or full out, the result is a plane that is never still or flat.

The second odd thing here is that Integral is set to 0. That is because in this case, the plane had a lot of momentum and this lead to pretty big oscillations. One of the way to get rid of those oscillations is to lower Integral and make the Derivative greater. That is also why the Derivative is almost as high as the Proportional component.

Overall this values do work pretty well, though they are pretty low, leading to a plane that takes some time to adjust to a new target position. It does take some time for the plane to stabilize, but it eventually does. Those values will eventually allow the plane to land on the track safely and smoothly, although the "deadband" had to be increased as a 0.7 range was near to impossible to achieve.

4
Conclusion

In this report, we have talked about two types of controllers: the BangBang and PID controllers. These controllers play a vital role in the realm of process control, each having their strength and weaknesses.

The BangBang controller, while seemingly basic and intuitive, can be remarkably efficient in certain scenarios. Its simplicity allows for quick implementation and is suitable for on-off control applications. However, its limitations became evident in this case, as the plane could not be stabilized with this controller.

On the other hand, the PID (Proportional-Integral-Derivative) controller represents a more sophisticated approach to control systems. By continuously adjusting the output based on three components: Proportional, Integral, and Derivative, it provides a higher level of accuracy. The PID controller is a powerfull tool to have and is widly used in control systems.

As we conclude this report, it is essential to acknowledge that controller selection depends on the specific needs of the system at hand. While the BangBang controller can serve really well in simple applications, the PID controller shines when precision and dynamic control are required.

In closing, this report serves as an introduction to the wide world of controllers. It is a starting point for further study and experimentation, encouraging us to explore more advanced control strategies.