Design of a discrete open loop control algorithm

To begin, consider the four inputs to the system: throttle, pitch, roll and yaw, which are pulse position modulated (PPM) signals coming out of a receiver. Using Arduino, time the rise and fall of each respective channel using the *mircos()* command. Every control stick has an upper and lower pulse position, determined by the length measured of the pulse, and they are all the same. The measurement is stored as an integer variable.

Say denote the four inputs, with “1” denoting that this is the first state of these variables. State 1 is classified by the minimum and maximum . Generalizing this notation, for any state , and are the maximum and minimum values respectively.

Therefore, . For PPM values that worked for every axis, it has been shown empirically a good selection is . The higher that is and lower that is, the less resolution the input will have, since values outside of will be constrained to these limits.

However, the resolution of State 1 is not fully defined by the above equation. The issue is that the PPM is encoded at discrete intervals of 4. It is desired to transform State 1 into some State 2 where each corresponds to a proper discrete interval. Just be careful that the selection of in the program is a multiple of 4.

Resolution of State 2 is defined as: , which means that there are unique discrete values. Control of the drone does necessitate that there be an odd number of states, so that there can be an equal resolution split down the neutral position. This will be implemented later, but it does impose a requirement: is even, which requires to be both even or both odd, but not mixed.

State 2 has a downside in that the state is shifted forward by its lower bound, rather than its lower bound being 0. For example, transforming for the empirical data results in . Instead, some State 3 could be produced with the same resolution as State 2 but rooted at 0.

Now, all the variables will range from . State 3 is the last state where the input values will all be in the same bounds. From here, the throttle should continue to be a one-sided input, but the angular commands should be a two-sided input. This is to establish that there are “positive” and “negative” directions for rotation.

To establish what these directions mean in real life, consider the quadcopter as a “X” with its four motors. Motor 1 is placed on the upper-left corner of the X, with Motors 2, 3 and 4 following placement on the corners in the counterclockwise direction. It is established that pitch is rotation about the Motor 2-4 line, with positive direction as upwards from Motor 1, and negative downwards from Motor 3. Similarly, roll is defined as rotation about the Motor 1-3 line, with positive direction as upwards from Motor 2. Yaw is defined as positive counterclockwise motion about the plane which all four motors and the frame rest upon.

With this nomenclature it is required that the pitch, roll and yaw are “split up” into two-sided inputs in some State 4. It is first necessary to define the “middle” point of State 3, call it . Since is even, say that . It is sufficient to say that so it is going to be substituted by that and largely left as such. Classify State 4 below:

Just to recap, it is possible to go back all the way to State 1 and classify State 4 on that basis:

State 4 is the last realized state of input, before it is used in some transfer mechanism to the output. It can be seen here that the throttle has one less than twice the one-sided definition as the rotations (“zero” is shared). For the sake of fluidity throughout the document, a vector can be established relating all quantities back to the one-sided definition. The resolutions can also be used as maxima for the actual input values by subtracting by 1.

For the example, .

Now for the outputs. The outputs to the motors are rigidly defined due to the use of the Servo library to control them. They are given variables for Motor 1 through 4, . Say they are bounded by , which in reality is [20,180]. At 20, the motors are at a stop, and at 180, they are full power. The resolution can be described as , which for all practical purposes .

Perhaps it would make sense to remove the offset, so a more well-understood state can be used:

At first glance it may seem strange that any given input has less resolution than any output. However, it should be understood that a single variable of the output is defined by multiple variables of the input. How to relate those variables is a challenge with many systems of equations and inequalities.

A preliminary starting question would be, “*how many input variables are required to resolve the output?”* It turns out that there are three required at any given time: throttle, roll or pitch (depending on which motor can actuate that command) and yaw. There is a solution in the continuous case of the problem which goes like this:

The idea of the challenge is instead of using floats in a continuous case, to be able the model the transfer of the inputs to outputs in a discrete sense.

From here on in the document:

The example case for a generalized strategy for all motors will be the governing equation for Motor 1. Herein:

Without knowing about how works specifically, it should be noted that the input arguments are on a different fixed-point scale than the output. There must be a way to match the resolution of the input with that of the output. Establish an axiom for this purpose:

This axiom could direct one towards two different paths. A possibility is to make , which would put every element of . In a continuous system, that would be a perfectly fine solution. However, a discrete system does not exist accurately without the use of floating point values. For speed it is desired to stick to integers.

In this way, it is necessary to establish that the function must have discrete outputs that are in the series of the output, . Some transformation of the inputs must occur to match.

To start, consider that all three inputs are additive in some manner. Therefore, the function can be separable into its three components:

Consider that each variable, regardless of the function imposed upon it, maintains its native one and two-sided resolutions. For all practical purposes of flight control, the one-sided resolutions will be used. With the assumption that the sum of the input resolutions is greater than the resolution of the output, the requirement for the function is as follows:

This imposes two problems: (1) that must be within the bounds of , and that any values of must be an element of . This requires a higher level of constraint on what the input values can be, and may also vary the interval on which they are passed.

Consider the continuous form for pitch: . Say in the case of positive rotation, there is a step up in motor speeds for and a step down in speeds for . Say the step width is and that it is the same for both motors. This was triggered by a single positive step in the input . Consider the initial case where . However the pitch is not going to solely determine the speed, so can be removed from the term so far. Therefore it can be rewritten: . It is known though that . is in an entirely different sequence, but it is reasonable to expect some *maximum width* between the initial and the new value on one motor, which is what the pitch is actuating. This will be denoted by , which can relate back to the original incremental step change:

Now it becomes rather tedious to continue defining some arbitrary step distance in motor speeds, and it really unnecessary. The best one to use that will maintain the highest mapping definition will be . Thus, which will fix the amount of points there are in the sequence. The question then becomes, how many points will there be? Better yet, how many points *can* there be, and how can these be correlated back to the input?

Consider the upper boundary condition: is desired for each motor, and that is desired at some position which means:

There can be an obvious problem though if one decides to assign . It was given that *must be an integer* and the selection of it as such (remember, this is a given rather than an unknown) means that the right side of the equality forces two conditions outside of just the equation:

The easiest way to satisfy Condition 2, if it is not naturally satisfied, is to subtract by 1. Now that are given, then it is possible to resolve . Another two conditions now arises due to the denominator:

Running through some possible selections for the sample case: are all of the good values. It is no surprise that the recognized *actuating* step size is . An example of this being implemented is as follows. Consider so evened out, , and it is selected that It follows that . Therefore, Then the input series and the output series for some and would be:

What should be noticed is that by using integer division, the values which would be greater in floating point are floored. For example, but , and in reality it would be .

Also notice that and such if were negative, then the values would flip signs in the output.

The same set of equations for pitch are the same but for roll. This will be addressed further.

The derivation for yaw is similar in some ways, but it affects motors in different ways. Consider the equation and that each motors is stepped in its canonical direction by 1. Say also that the maximum difference in individual motor speeds is defined by , and there is some corrected with the new condition . The governing equation now becomes again with and the constraints on . For the case used in this project: . Without deriving the mapping, one can see that it is in the same way as the pitch and roll.

One might be concerned that, with a very high-resolution input, that some limit be imposed such that . The motivation is that one does not artificially create a output value higher than .

Recall again All that is left is to find a satisfactory so that the output is fully mapped to the input. Denote which is the maximum possible output attributed to roll and pitch. The below condition must be satisfied:

Since it is expected for each step for the speed to increment some , then the function can be defined as: . From here on it just makes sense to continue with the real example derivation. Recall that so that a value for can be solved:

From here it becomes clear that it is fruitless to make a to be stored as an integer. Instead, the analytical representation of it will be implemented into the total function. Instead, there should be a variable defined as

The adjustment will now be made for adding the to the equation:

There is one problem with this notation that is an artifact of programming: It is possible to have inputs such that the speed is negative, specifically not having enough throttle when the pitch and yaw are negative. Motors can only go negative when there is pitch/roll and/or yaw being imposed. Therefore, there exists a throttle cutoff value which will impose these conditions:

This condition is imposed on Motors 1 through 4, so that they are all spinning at the same speed prior to being greater than . The value is derived from satisfying this inequality:

This would make a non-integer, so to satisfy the inequality it must round up: