Contents

1 Altitude Control

1.1 Goals of the altitude controller

The altitude controller does exactly what the name implies—it controls the altitude of the balloon using the ability to vent gas from the valve and drop ballast. In general, it's goal is to keep the balloon's altitude within an allowed range while minimizing ballast use. Note there there is not necessarily "hard" range of altitudes, but you can think of it as two soft boundaries (say, 12.5km and 14km) that we wish the balloon to fly between, and staying around the middle is preferred. Also, its preferred to not rise and fall too fast (although you probably wont anyways if ur minimizing ballast use).

1.2 What makes it nontrivial (and thus fun)

I've noticed that after working on this project for so long, when I discuss the challenges of controlling ValBal's altitude with others, there is a disconnect between the what I focus on, and what they think about initially. This is because the things that make life hard are not always immediately apparent. The basic system sounds pretty simple: vent gas to go down, and drop ballast to go up. Try to stay around some reasonable altitude range. Sounds simple enough, right? If your a control's

engineer, you know things in real live are almost never that simple, and here are the factors that make it more than your textbook feedback control problem.

Noise and Uncertainty

This is what makes life hard but worth living. Noise turns a simple and easily solved problem into a much larger and complex beast. Ever single element of the valbal controls in coated in a thick layer of uncertainty and noise, it's what happens when you build the cheapest and scrappiest possible system. Every level of the controller must be robust to this, otherwise it's not really worth thinking about. Here are a few examples

- **The Atmosphere** is a harsh mistress. It's extremely noisy, and can be difficult to model. While the balloon dynamics are relatively straight forward, the disturbances on them caused by the atmosphere, are not. Turbulence causes winds to push the balloon up and down, and changes in albedo of the ground below can heat and cool the balloon in ways that we do not currently have sensors to measure. To make matters worse, these effects are not simple white noise, they are often highly structured in often difficult ways. Most notably, the atmosphere is like a the ocean, it's coated in waves. Next time you fly, take a look out the window while flying over a layer of flat clouds. You will often see rolling waves as if it were the surface of the ocean. The balloon floats atop these waves, frequently bobbing up and down. This makes life hard, as sometime the balloon will be riding a wave, and for minutes on end be actually falling or rising at velocities on the order of 1m/s when the balloon is actually perfectly neutrally buoyant. This means that the estimated velocity must be heavily filtered for the balloon, introducing tons of phase lag and a host of other issues.

- **Control Actions** are highly uncertain. Ballast typically does not have too much uncertainty, as we can measure how many BB's are dropped per second on the ground, but this can still change during flight due to electrical issues, wear and tear on mechanics, and temperature. However, the real issue it vent uncertianty. The rate at which gas vents from the balloon is entirely a function of balloon overpressure, which can range from being high pressure when the balloon is new and at high altitudes, to being 0, when the balloon is old and the latex decays. Measuring overpressure is expensive, and estimating it is hard, so this effectively means that at any point in flight, we really don't have a good idea of how much a vent is going to change the lift of the balloon.

- **All altitude measurements** come with much more baggage than just typical measurement noise. Barometric accurate is usually fairly smooth and accurate relative to itself. however, it comes with a large offset due macro scale pressure fluctuations in the atmosphere. GPS altitude is fairly good as an absolute

measurement, but it may jump around or can loose lock for long periods of time.

1.2.1   Nonlinear dynamics

Balloon dynamics are nonlinear due to drag on the balloon. However, the dynamics are simple enough that they can be decently well linearized, but it's still an additional thing to worry about.

1.2.2   Desired altitude is a poorly-defined range, not just a single altitude

The actual goal of the controller is not well defined on the outset. It's not as if we are aiming for some target altitude. If we are only trying to maximize endurance, we really don't care what altitude we are at as long as we are above 12.5km and not too high (the upper bound is not well defined, call it 15km or so). This poses a problem if trying to use classical linear feedback control.

1.2.3   Being "Optimal" actually matters

Any good engineer knows that 95% of the time there is nothing better than "good enough". There's always a more fancy, theoretically-better way to do something, and it's usually not worth the time and effort and added complexity if you want to actually get things done. For many feedback control problems, a simple tuned PID controller will work just fine and it would be a waste to do anything more. However, for ValBal, this is not quite the case. Don't get me wrong, I use approximations and hacky things that are dumb but work all over the place on ValBal, but a simplistic PID controller or similar is not sufficient for altitude control. The control effort, dropping ballast and venting gas, is a very finite resource, and is usually the limiting factor on ValBal flights. Endurance is the main way that we measure performance on a ValBal flight, and the controller directly impacts this. Combined with the above point that we are not just trying to hit a specific target, but rather a range of altitudes, this means that any good altitude controller for ValBal will need to depart from a classical linear feedback control system.

1.2.4   Everything must run on a low power embedded processor

This is not an issue for classical control methods for the most part, but it a major hurdle for an optimization-based method. Not a big deal, but it is something to think about.

1.3   Modeling valbal

In order to control the balloon, we need to understand how it works. We aim to come up with the simplest possible model that accurately descibes the balloon's

behavoir, and thankfully it isn't too difficult to do so. [TBC]

1.4 Layman's summary

I've found that writing out the math nicely using notations from controls to be more confusing than necessary for an average person, so this section will explain things in English as much as possible. It's probably longer than the mathy sections as it's meant to be more intuitive, but less concise and precise.

The altitude controller uses 2 nested control loops, the outer one taking in the altitude target (middle of the desired altitude range) and outputting a commanded velocity, and the inner one takes in a commanded velocity and outputs the commanded change of lift of the balloon per unit time (I call this $\dot{l}$ in latex and on slack I call it dldt). A continuous-valued $\dot{l}$ is approximated by dropping small amounts of ballast or venting for a small interval of time with a variable time interval between, much like PWM to approximate an analog signal. The key to the whole controller are two elements, the velocity estimator and the deadband.

1.4.1 Velocity estimator

To get an estimate for velocity[1] for the inner loop to control off of, I heavily filter the derivative of position as it's noisy as hell, and also add in an estimate of how previous controller actions (like ballast/vent) will effect the the future velocity of the balloon, as it wont be seen right away in the heavily filtered velocity.

1.4.2 Deadband

To make it so the controller tries to stay with in two boundaries rather than just always shoot for the middle, I add a non-linearity in the for form of a deadband. I calculate this deadband such that the controller is perfectly happy to fly at a constant velocity and offset from the set point by some number of meters (corresponding to where the bound is). That is, there is an allowably steady state error. However, if the balloon is rising/falling away from the middle setpoint but not yet past the boundary, the controller will act such that it slowly levels off and converges to the boundary
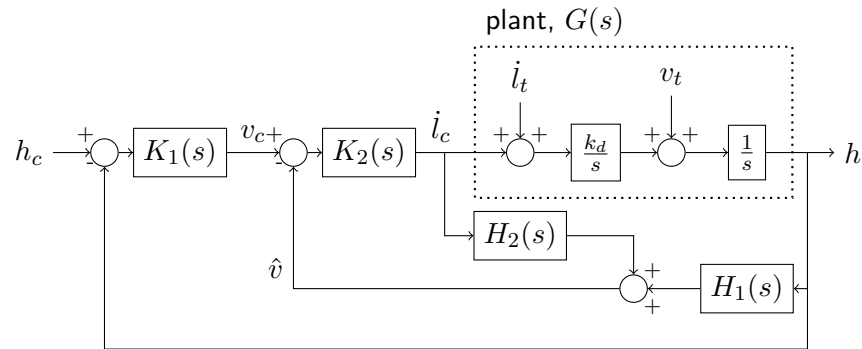
1.4.3 Compensator

The control comspensators themselves are actually dead simple, just proportional control with the gains between the two tuned to a large time constant and critical damping. this is because I put all the "work" of the controller in the estimator rather than the compensator (note about linear controls is that estimators and compensator are in fact two sides of the same coin)

---

[1]I used the terms velocity and ascent rate interchangeably

1.5   More mathy version

Lasanga consists of two nested feedback loops, an outer loop that tracks an altitude and commands a velocity, and an inner loop that tracks a velocity and commands a change in lift. A block diagram of the idealized system is shown below[2]

$$
\begin{array}{ll}
K_1(s) = K_h & \text{proportional altitude loop compensator} \\
K_2(s) = K_v & \text{proportional velocity loop compensator} \\
H_1(s) & \text{veloctiy estimator, lowpass filter on derivative of position} \\
H_2(s) & \text{estimator of effect of changes in lift}
\end{array}
$$

$$
\begin{array}{ll}
h_c & \text{commanded altitude (set by Flight Controller)} \\
v_c & \text{commanded velocity (output of position loop)} \\
\dot{l}_c & \text{commanded change in lift per unit time (output of velocity loop)} \\
\dot{l}_t & \text{atmospheric disturbances that change balloon lift (heating/cooling)} \\
v_t & \text{atmospheric disturbances that change balloon velocity (turbulence)} \\
\hat{v} & \text{estimate of velocity} \\
h & \text{actual balloon altitude}
\end{array}
$$

Figure 1: Idealized system block diagram

"Plant" or $G(s)$ refers to the *system* (atmosphere + balloon) dynamics, whereas everything else represents the control feedback structure that is computed in the ValBal flight code.

---

[2]For those without much control background: block diagrams represents the structure of a control system. Each block represents some kind of system with inputs and outputs. Each solid line with an arrow on it represent a value and show how outputs of some blocks are connected to inputs of others. Circles represent adding or subtracting, (the +/- sign where the arrow enters the circle indicates which) while blocks represent *transfer functions*. Transfer functions are linear systems represented in the Leplace domain. Without going into too much detail, a simple Transfer function $K(s) = 5$ (s is the variable in the Laplace domain) is simply scalar multiplication by 5. $K(s) = 1/s$ represents integration of a value, and $K(s) = s$ represents the derivative of a value.

1.6   Constants Definitions for Code

Contstant	Default	Description
freq	20	frequency that the controller is called at (Hz)
k_v	1e-3	velocity Gain
k_h	1.5e-3	altitude Gain
b_dldt	6e-4	lift rate of ballast actions (kg/s)
v_dldt	3e-3	lift rate of valve actions (kg/s)
b_tmin	5	minimum ballast interval (s)
v_tmin	5	minimum valve interval (s)
h_cmd	13000	commanded altitude (m)
kfuse	7	atmosphere gain for velocity estimator
kfuse_val	0.5	velocity estimator gain modifier for valve
ss_error_thresh	750	error tollerance for deadband (m)

2   Trajectory Control