Exercise 1

Risto Rushford

February 3, 2020

Contents

1 Object in Freefall and with Parachute

1.1 Strategy and Expectations

For this first exercise I built a model to represent an object in freefall. Then as I went on to develop the second model, I realized that the graphs remind me of the output from an amateur rocket simulator program called "Open Rocket", and chose to make adjustments based on what I've seen of a rocket deploying its parachute at a set height after "apogee" (when the rocket begins falling back to earth after flying up away from it).

At $t = 0$, the "rocket" is at it's maximum altitude. My first run with the simple model had it set at an unreasonable height (for an amateur rocket), but I adjusted it to begin falling at 2000 meters for the second, more complex model. I was not able to model equilibrium or goal-seeking behavior into this model, yet, but I hope to achieve that with a future exercise. This exercise has challenged my understanding and comprehension of system dynamics and the use of the tool Vensim, and I look forward to having more opportunities to push myself further.

1.2 Model Development

To build this model I needed to make some simplifying assumptions:

- Time is measured in seconds, with timesteps of .25 seconds for a total of 50 seconds.
- Beginning height is arbitrary - although once I adjusted the model to reflect amateur rockets, I set the height within reasonable bounds.
- Air density can be approximated as "1" (this was to avoid floating point overflow errors).
- Gravity can be reasonably set to **10** (again, to avoid calculation errors).
- I can ignore terminal velocity (for the time being)

1.3 Model 1: Freefall

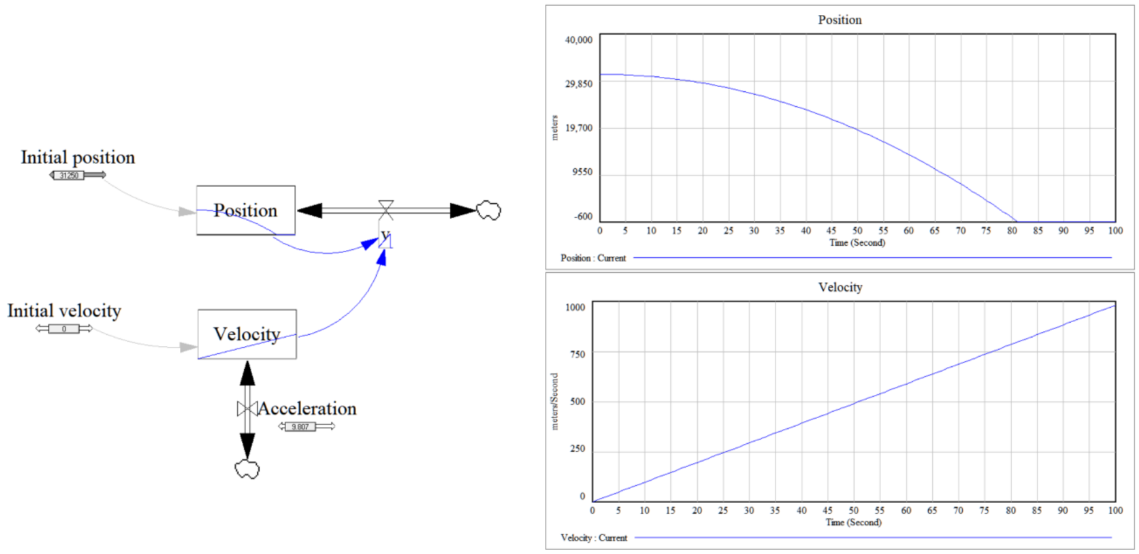

Figure 1: A run from the simple free fall model with graphs

This model acted just as I expected that it would, with velocity increasing at a constant rate due to acceleration from gravity, and position decreasing at an exponential rate. The equations I used for velocity and position were:

$$v = a \cdot t + v_0 \tag{1}$$
$$v_0 = 0 \tag{2}$$
$$p = p_0 + p\ dt \tag{3}$$
$$p_0 = 30,000 \tag{4}$$

With v = velocity, p = position and a = acceleration. Note: I used SyntheSim when I took the above screenshot.

I also used an IF THEN ELSE function to cease position change when it reached a value of zero. Because I did not include terminal velocity into the equation, the velocity continued to climb throughout the enitre freefall. This is something I did not think to factor into the model at first, and intend to include in future work. This would be a "goal seeking behaviour" for the velocity function.

1.4 Model 2: Parachute

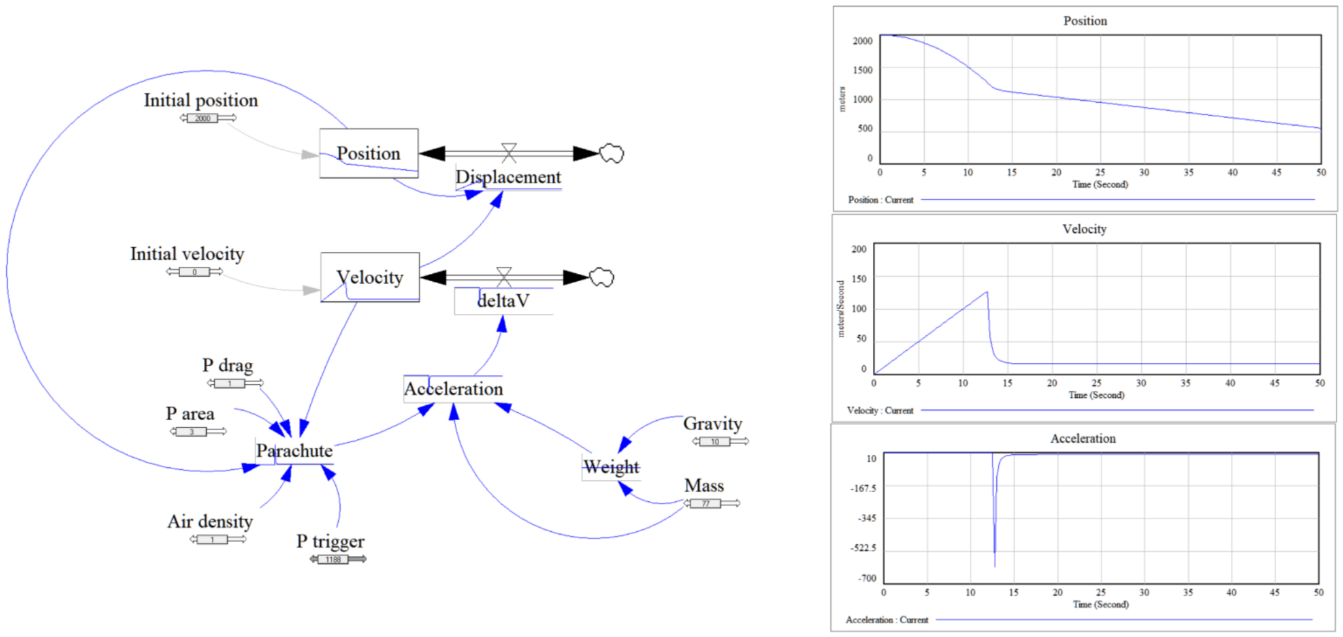

Figure 2: A run from the parachute model with graphs

This second model was much more challenging, and I had to do a fair amount of research about the variables involved to develop it to the state displayed above. I took the initial freefall model that I wrote about in the previous section, and added parameters of weight (mass times the acceleration of gravity) and the drag created from deploying a parachute.

I had several challenges arise during the parachute model that took me a while to figure out. First was figuring out how and when to trigger the parachute. It seemed obvious that an IF THEN ELSE statement would again be useful, but I initially was thinking that I should set it based on velocity. When I did this, it quickly became apparent that this was not the correct method as the model would fall into a pattern of triggering and “un-triggering” the parachute as the velocity oscillated around a certain range.

$$w = m \cdot g \tag{5}$$

$$a = \frac{w - d}{m} \tag{6}$$

$$r = -air \cdot C_d \cdot area^2\, v^2 \tag{7}$$

The additional equations used in the *Parachute* model, with w = weight, m = mass, g = gravity, r = parachute resistance, air = air density, C_d = parachute drag coefficient, and $area$ = area of the parachute.

I corrected the strange model behavior by setting the parachute to be triggered at a certain altitude (1000 meters by default). It was at this point that it became necessary to adjust the initial position to a smaller range, as the parachute drag calculations were causing floating point overflow errors. I tried several means of simplifying the parachute drag variables (reducing them to integer values, eliminating the drag coefficient and air density) but the error still kept popping up.

Eventually I learned how to trace the error and determined that the problem was simply that the velocity function was climbing to too high of a value and then the parachute function was taking its square, so the model broke down every time the parachute would deploy. This was solved by reducing the starting position to only 2000 meters. The graphs above show the expected relationship between position, velocity and acceleration, providing an element of validation to my model (this is where the model began reminding me strongly of the amateur rocket flight profile simulator, **OpenRocket**).

I did begin working on a rocket version of the model, but have not had a chance to meaningfully implement the changes before this assignment became due.

1.5 Conclusions

While I know that these models are far from complete, I believe that they do a decent job of representing real behavior of objects in freefall, first in a vacuum but subjected to constant acceleration (model 1), then with an item falling from a specified height and deploying a parachute (model 2).

1.6 What I learned

I learned a lot about building my own model in Vensim through this exercise (as opposed to following along with the slidecasts in D2L). Despite weeks of reading through the text and working through the examples in Vensim, there really is no replacement for performing my own model formulations and developing them myself. I've also learned important lessons on how to verify and debug my model, and should be able to complete the next exercise in less time (or conversely, accomplish more in the same amount of time).

1.7 Potential Improvements

Even the parachute model is a relatively simple one. I would like to expand this model to emulate an amateur rocket from launch, to apogee, to parachute deployment and landing. I could also take it a step further and attempt to emulate some control functions such as propulsive landings and thrust vector controls in windy conditions.

See below for a graph from one of my OpenRocket simulations.

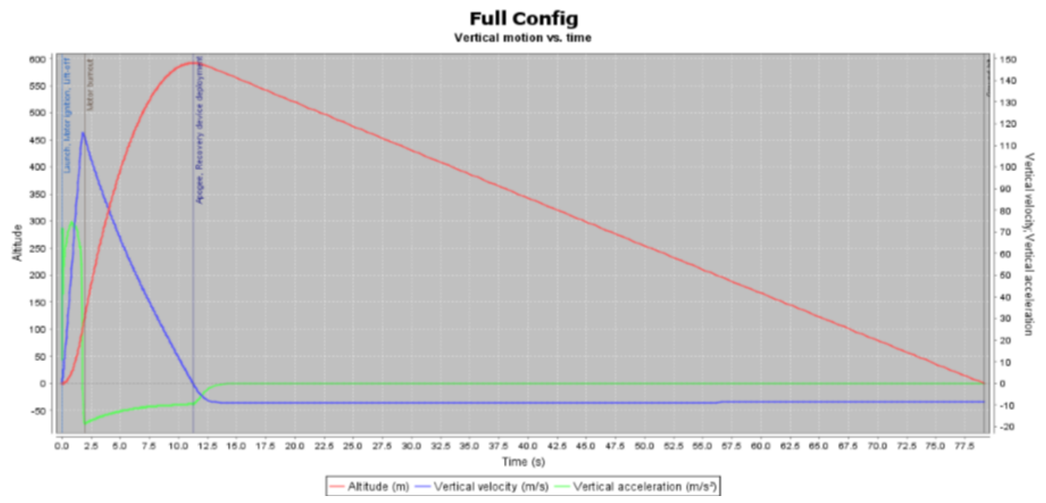

Figure 3: A pre-flight simulation from just before I got my **L1** amateur rocketry certification.