IMU positioning system

We knew from the very beginning of the season that we needed to fix an issue common to both Ask Academy teams last year. This issue being a lack of a stable autonomous. Autonomous is often very difficult to program for due to having to tinker with movement values. We set out to solve this by being able to read in Velocity and Position of the robot during autonomous -and operator control to. An idea was developed that involved using an IMU(Inertial Measurement Unit) to grab axial acceleration, then use integral calculus to compute the velocity. From this velocity we get position using integration too.

As hinted at before, the IMU has the capability to track acceleration on multiple axes. We planned on using this to find axial velocities using tabular integration, and assumed initial velocities of 0. We then planned to use tabular integration on these velocities to find position values using starting position settings for the robot. The math for this works because $\int_a^b A(t)\ dt = V(a) - V(b)$. If we assume $V(A) = 0$, then $V(B) - V(A) = V(B)$. We then got position because $\int_a^b V(t)\ dt = P(b) - P(a)$. If we say $P(a) = C$, C being a constant, then $P(B) - P(A) = P(B) - C$. However, we can't just use these plain integrals, we have data points not formulas to work with in our code. Note, we did have to also divide the acceleration by gravity in ft/sec^2 since it was measured in earth gravitational units.

In order to work with aforementioned data points in integrals we use a method of tabular integration, namely, the trapezoidal riemann sum integral approximation. The formula for that approximation is: $\Delta X/2 * (f(x_0) + 2f(x_1) + ... + 2f(x_{n-1}) + f(x_n))$. We implemented this using a code efficient method. For example, for velocity we declared a global float called V_x, we would then add $(A_x + PA_x) * dt/2$ to it every clock cycle(every 20 ms). In this case pax was previous ax, and we would start with it at 0, and change it to ax at the end of the code. This ensured the first/last parts of an integral were only represented once, but that the middle parts were represented twice(needed because of the $2f(x)$). Note, we also used formulas found in the code to get overall velocity/acceleration for controller readings during operator control.

We ultimately had to end up abandon the IMU tracker due to feasibility. After looking at more forum questions about doing the same thing we were doing, we saw that quite a few people had tried it, but to no avail. Though the concept works theoretically, turns out the slow refresh rate of acceleration makes the velocity and especially position(due to error magnification) not remotely accurate within just a couple of seconds. However, shortly after we came up with an easier design that dodges this flaw.