Team Assignment 4
CDF Analysis

Denver Hoggatt
Spencer Clegg
Colton Ottley
Rohith Prasad

After finishing our decide function as specified by the requirements given to us in assignment one, we were asked to create a moderately complex input for our decide function, call it 100,000 times, recording the time for each execution, and then store the result in a CDF plot. We created several X and Y inputs, randomly distributed amongst the capable double precision floating point values, and used these as inputs to our decide function. We then used the clock_gettime() functionality in the Beaglebone Black, combined with the CLOCK_MONOTONIC input arguments, to produce fairly accurate timing details for each of the decide function's run-times. We then printed out the results of the total amount of run-time each function call took, and computed the following CDF in MATLAB:

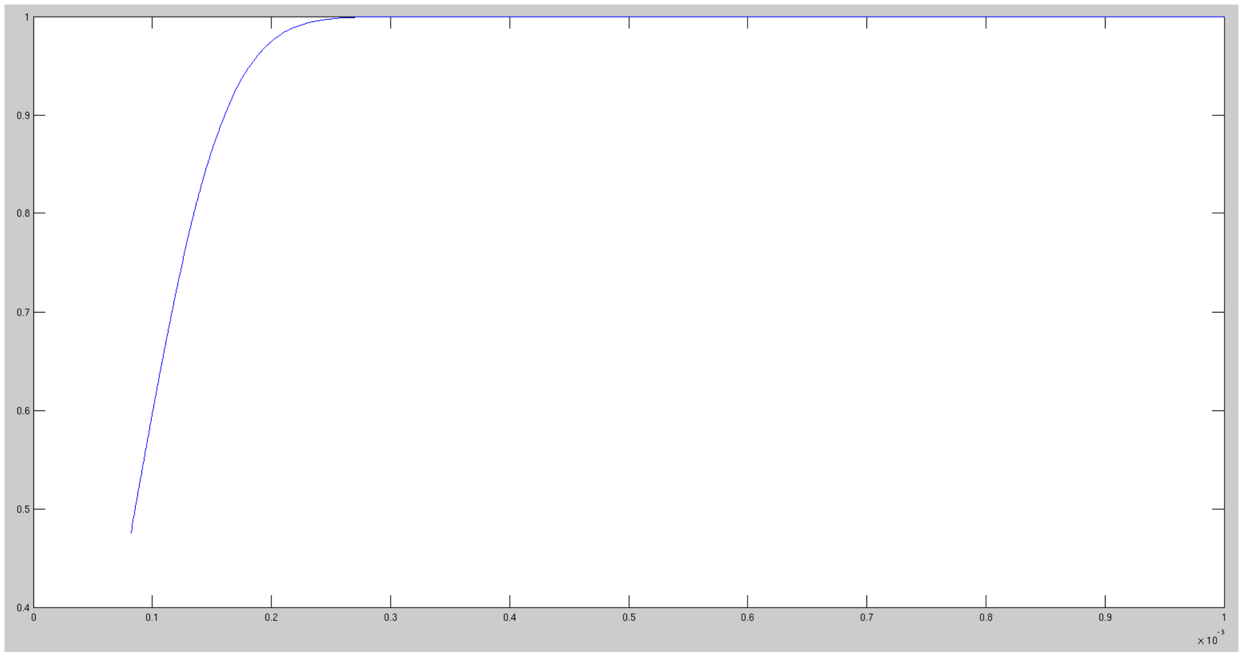

Figure 1: Cumulative Distribution Function of the Time for DECIDE to Execute

The X-axis in the figure is the observed execution time, and the Y-axis ranges from 0 to 1 (as with most standard CDFs). As can be seen, there isn't a lot of variance in the time it takes to execute. This is due to the fact that there aren't many extra tasks that the processor is handling, such as interrupt handling, and the processor can dedicate most of it's alloted time to finishing the decide function call. However, this is not perfect, and there are still other items that the processor needs to handle, which is why even under a single program run, there is still some stochasticity.

After running this test, and creating this graph, we installed the Iperf network testing tool on the BeagleBone Black, and ran the test again. However, this time, we ran the test alongside the iperf tool, as this would allow us to determine how much of an impact extraneous programs and interrupts have on the decide call. The results are as follows:

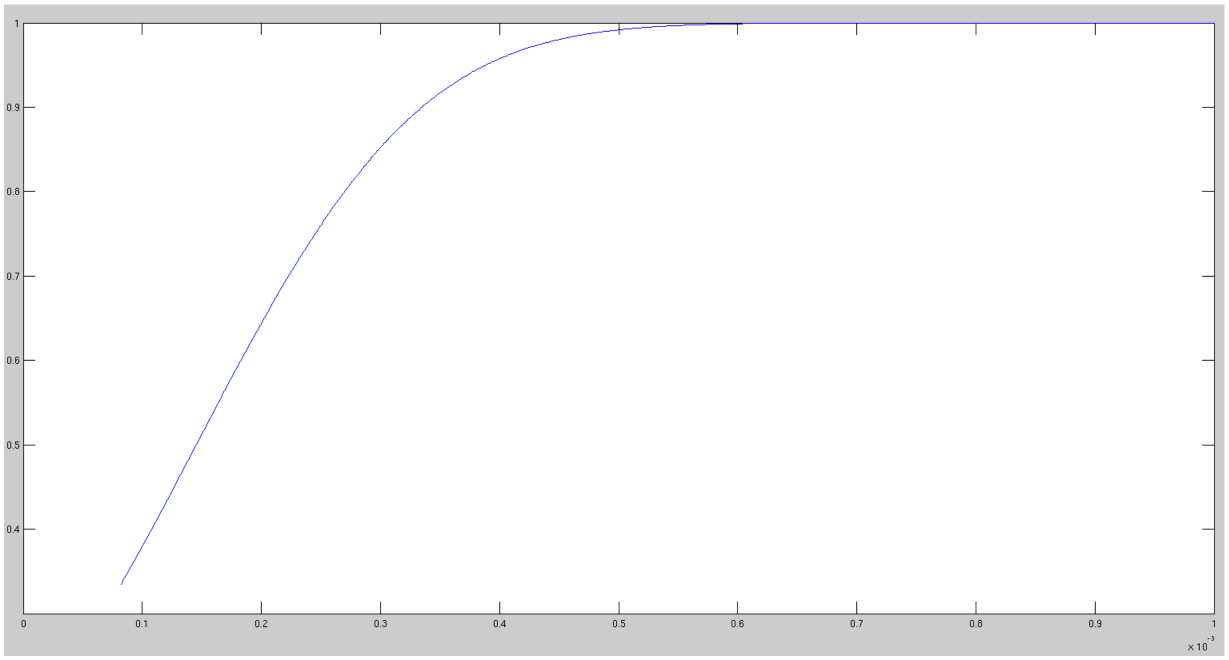

Figure 2: Cumulative Distribution Function of the Time for DECIDE to Execute While Running Iperf

The X and Y axis are the same as the previous graph. As was expected, there is a lot more variance in how long the program takes to execute. While many of the executions still ran in a similar amount of time, there are a lot more outliers that took much longer to execute. This gives us that long and rounded curve on the CDF as it increases.

Lastly, we need a way to fix this problem should our program come under a heavy network load. The first, and easiest way, is to simply disconnect the network. If timing for the decide program is that critical, then it should be worth considering whether or not we need any network communication in the first place. The second way, which is more difficult on a linux machine such as the BeagleBone Black, is to somehow get the program to run at a higher priority than the interfering interrupts. This could be done by disabling the interrupts, or by somehow allowing the program to run on a higher priority interrupt handler. Lastly, we could simply offload the program to run on a separate processor, which makes it so that the interrupts on the BeagleBone's main ARM processor won't interfere with the running of the code.