LRT Simulation for At Grade Traffic Flow

Tristan Ford

Systems (Signal) Engineering Co-op

2018-04-30/2018-08-31

SNC · LAVALIN

Table of Contents

SNC·LAVALIN

Background

The LRT Traffic Simulator is a project that was started with a specific goal in mind. We wanted to study the effects that imposing signaling methods onto level crossings would have on the Eglinton Line in Toronto. The Eglinton Line is an LRT with an at grade portion that is controlled solely by the driver of the train. By adding additional safety measures to the operation of the LRT we would be slowing throughput of trains and increasing cost of the project, so identifying the exact impact to the project is paramount. The reason for doing so is to make operation of the LRT safer and less prone to failures due to the driver or other factors.

A long-term goal for this project is to become a self-sustained simulation tool to be used for any railway. By creating this product in house, we can tailor the functionality to best suit the needs of SNC Lavalin.

Technical Background

To predict the effect on throughput of trains, two variables are important to keep track of: position and time. Knowing the position in time of each train on the railway will allow us to properly analyze how the line is evolving. Realistic models of train movement, intersections and stations are also needed so that our simulation provides useful information. These aspects are dealt with in various subsections in the simulation that will be explained in this document.

I have used MATLAB Simulink to create the simulation due to the block diagram toolkit. This program provides a modular approach to problems and allows quick iteration and testing. There is also the potential for large scalability.

Train Block

The core of the simulation is the part that simulates the trains. This block is designed to take as an input the distance to the nearest obstacle and output its speed and position. Figure 1 shows all the internal subsystems in the Train block.

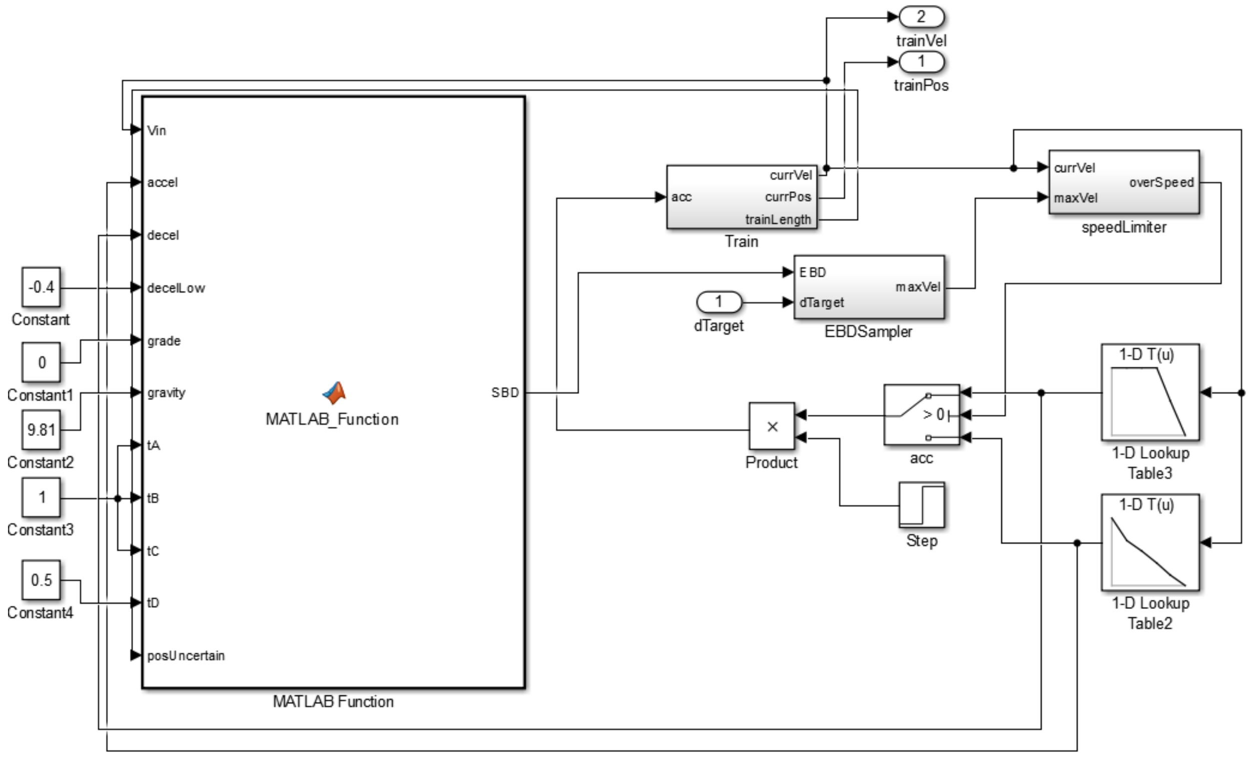

Figure 1

We will go through each subsection individually and explain their functions. We will go from left to right, starting with the block labelled “MATLAB_Function.”

SBD Calculator

This subsection runs a short bit of code that determines the necessary distance for the train to come to a stop under worst case conditions. The various inputs relate to the standard way of calculating a safe brake distance. The more obvious ones, like gravity, Vin, accel, decel, are for what you would expect. Something to note is the 0 input to grade. This simulation in its current state assumes a completely flat railway.

Train

The train block is almost exclusively an integrator as we can see from figure 2. This subsection takes in the current acceleration of the train to determine its speed and position over time. The more interesting part is the reset protocol. The block that compares the position output with 14500 is there to determine if the train has run the full length of the track. After reaching the end of the track, we reset the position to 0.

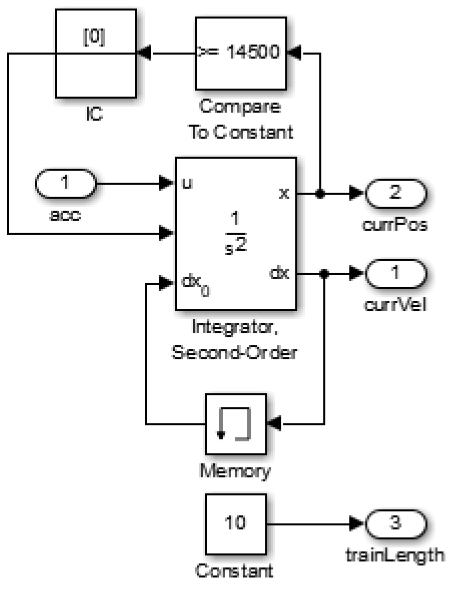

Figure 2

EBD Sampler/Speed Limiter

The EBD Sampler compares the safe brake distance and the distance to nearest obstacle to determine at what speed the train can continue moving. In conjunction with the Speed Limiter, this then compares the maximum speed with the trains current speed to decide if we need to accelerate or decelerate. This true/false signal is sent to the switch labelled “acc” to pick an output from either the acceleration or deceleration lookup tables. These lookup tables output a realistic acceleration for the current speed of the train based off real data.

Miscellaneous

The step function multiplied with the output from the switch is there for starting trains at different times. The step is 0 for a certain amount of time which keeps the train stationary and jumps to 1 when we want the train to start moving.

Nearest Obstacle

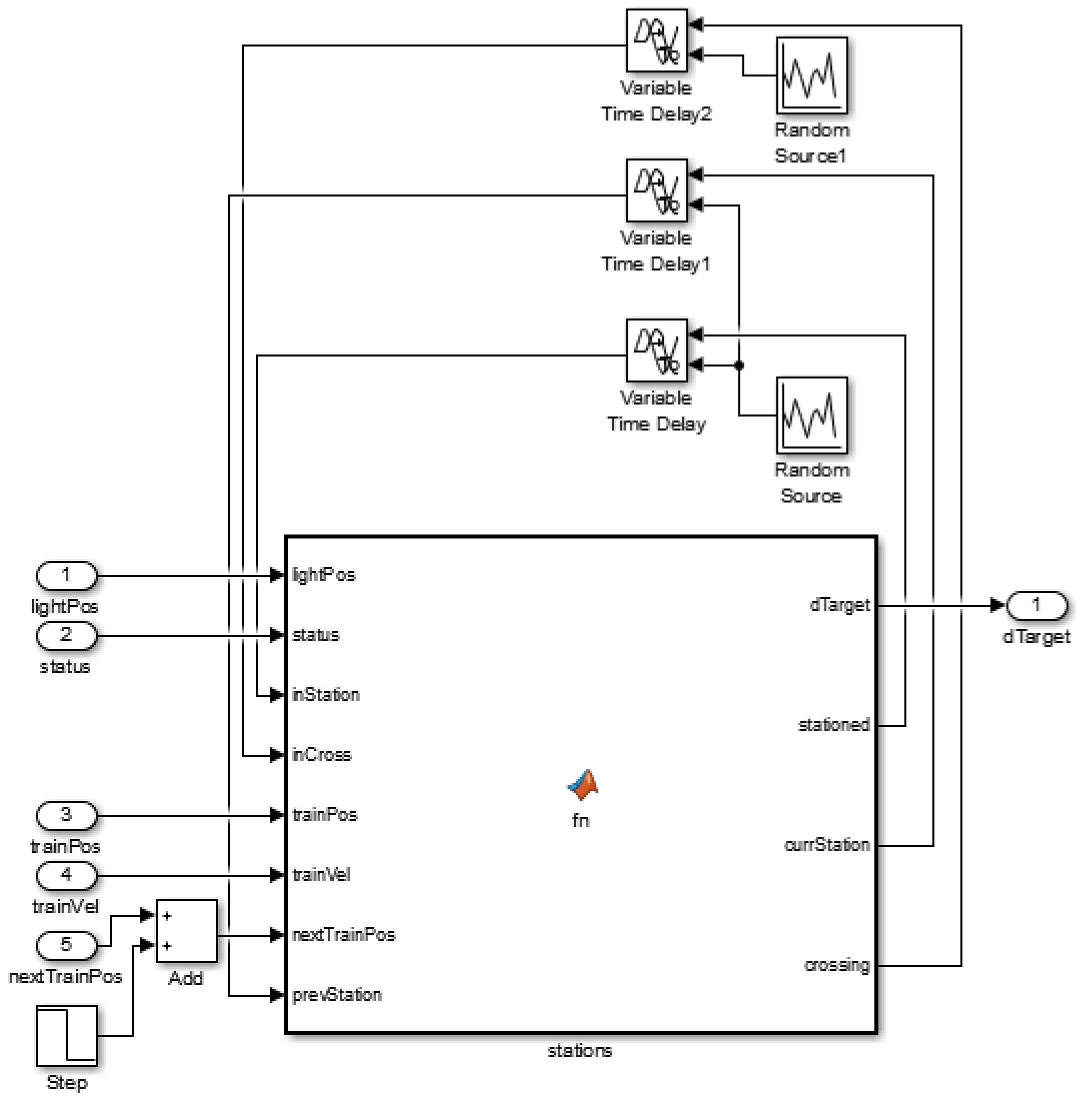

Figure 3

Figure 3 shows the subsection that determines the distance to the nearest obstacle for any given train. In the simulation there are three possible obstacles: lights, stations and other trains. In an, albeit large, bit of code this block compares the distances between the train in question and all other obstacles to output the distance to the nearest one.

The time delays are there to force the train to stay stationary for a certain amount of time at stations. Once a train has stopped at a station, that station is ignored for a certain amount of time to let the train pass through that obstacle. The time delay associated with the “crossing” output is for the ends of the tracks. Trains are supposed to stop at the terminus stations for longer periods of time, so they have their own time delay.

Overall Program

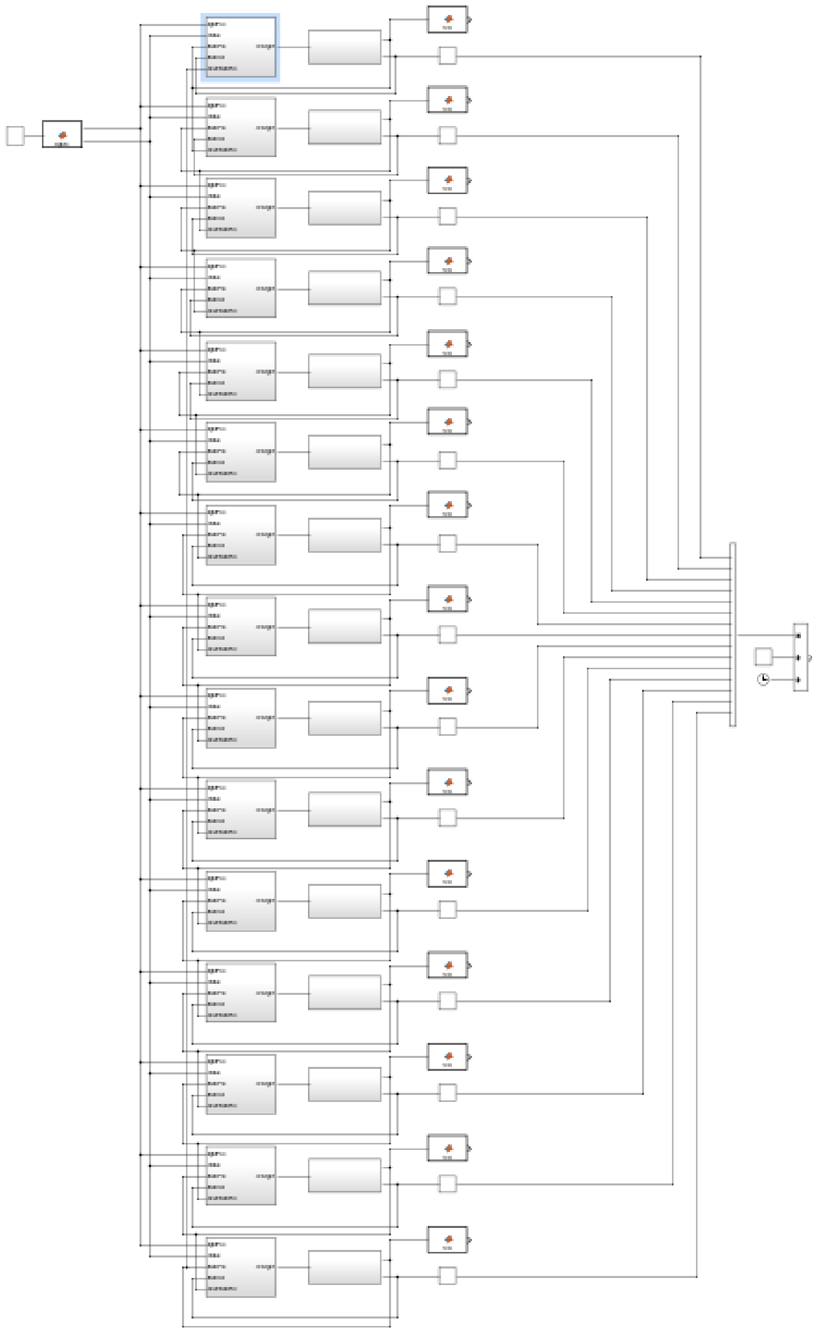

Figure 4

Figure 4 displays the setup for monitoring 15 trains running on the railway. On the left we have a small subsystem simulating the on and off status of lights which is fed into the 15 Nearest Obstacle blocks. These blocks are each associated with a specific train. The position and speed of each train is recorded and displayed on a graph.

Results

In figures 5 and 6 we see a snapshot of one run through the simulation. Something we want to see in the simulation is a reduction of bunching. We do not want trains running right next to each other. As you can see in figure 5, I have started the trains very close together and over time they separate. If we let the simulation run for long enough an equilibrium is reached where the trains will naturally avoid bunching.

Another important part of the simulation is randomness. We want each run of the simulation to be different so that we can examine every test case. This is achieved through random stop times at stations as well as randomizing train speed and acceleration rates. This is something to be improved upon in the future.

Looking at the average speed of the trains, we can estimate the throughput of trains and time to travel the length of the railway. These are both important numbers for cities building LRTs. This also provides insight towards deciding a reasonable line speed.

In conclusion, this simulation has provided a large amount of knowledge towards implementing signaling logic onto the Eglinton LRT. There is much room for improvement in the future and hopefully this simulation can grow into a real tool with a user-friendly interface.

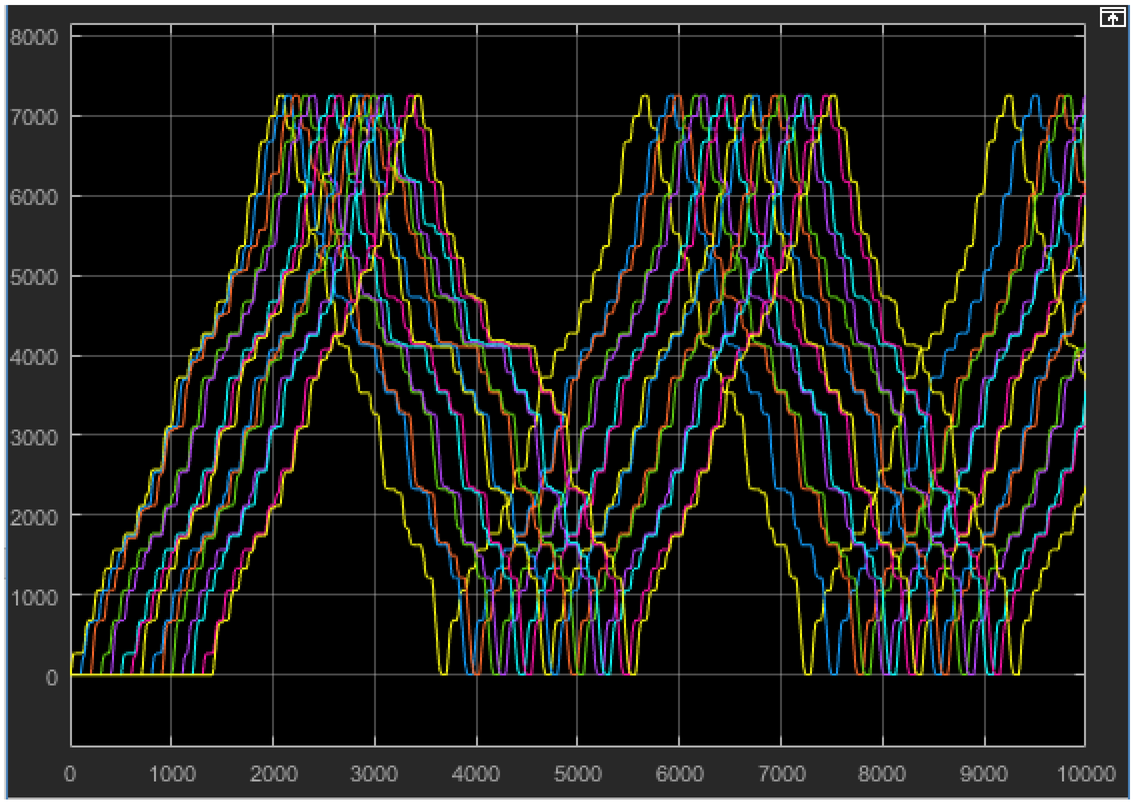

Figure 5, Position vs. Time

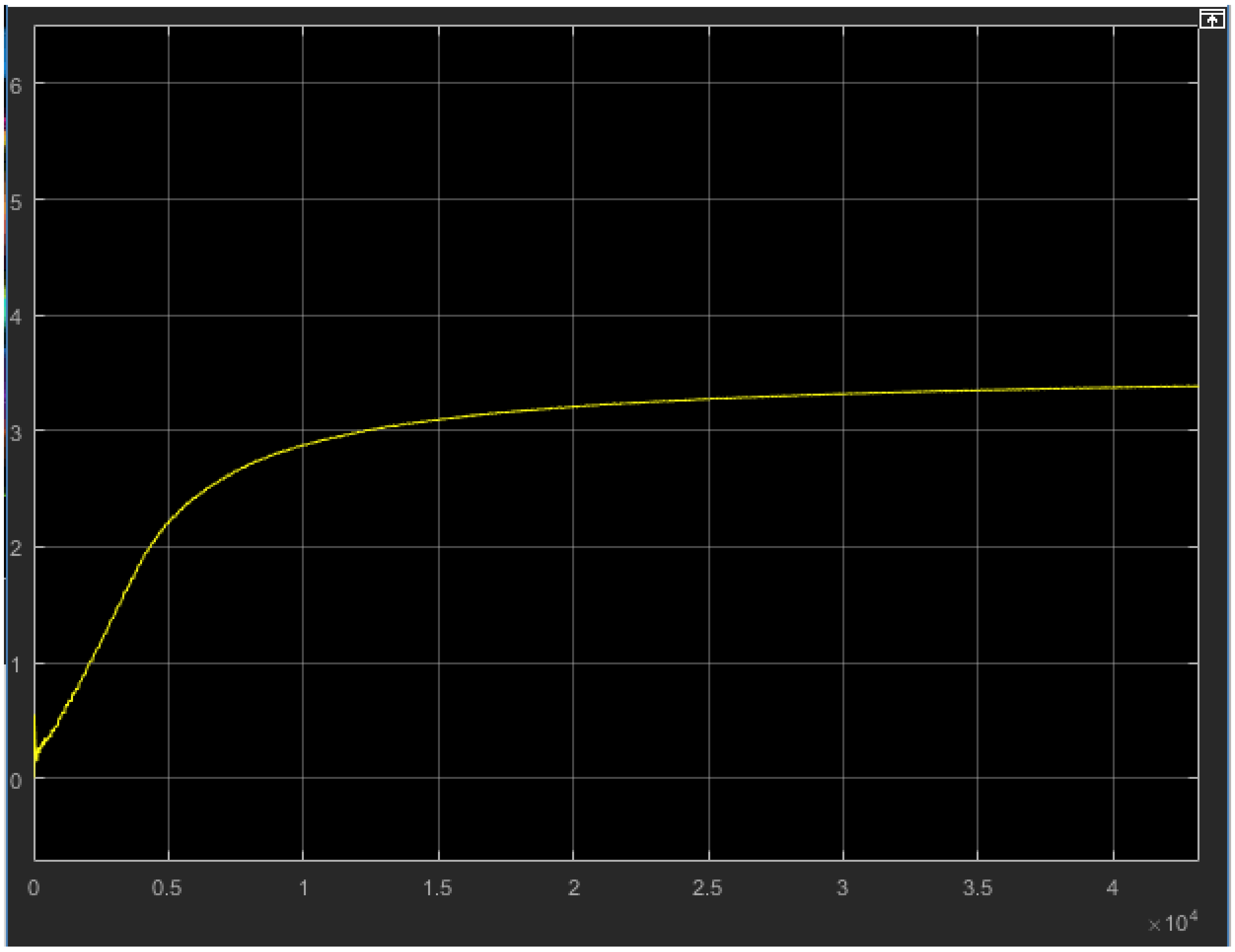

Figure 6, Average Speed vs. Time