Stale Data Analysis in Intelligent Transportation Platooning Models

Cavender Holt and Jon C. Calhoun
Holcombe Department of Electrical and Computing Engineering - *Clemson University*, Clemson, USA
Email: {cavendh, jonccal}@clemson.edu

***Abstract*—As autonomous systems become more integrated into human lives, it is essential to ensure that they maintain high levels of safety and reliability to prevent accidents or injuries from occurring. One rapidly growing area is that of autonomous vehicles. One common scenario for autonomous vehicles is traveling down a roadway in a platoon or convoy. The vehicles need to communicate data between them to maintain their positions. However, unreliability in the network can lead to missing or late arriving data, which forces the cars to use stale data from earlier communications. This paper investigates the effects of stale data on autonomous platooning models. In particular, we investigate which parameters may need to be protected by performing sensitivity analysis and analyzing how the platooning model behaves in the presence of stale data. By increasing the understanding of how the model behaves and which input parameters are influential, algorithms can then be developed to specifically target the characteristics of the vulnerable systems.**

***Index Terms*—stale data, sensitivity analysis, intelligent transportation systems, fault tolerance**

I. Introduction

In intelligent transportation, communication is utilized between multiple vehicles to maintain a fleet or platooning structure. As autonomous vehicle systems grow more prevalent, safety and reliability concerns become increasingly important. Communication sensors, however, are vulnerable to intermittent failure, especially in unstructured environments such as rain, cold, snow, etc. [1]. In addition to sensor failure, the vehicle system may also be subject to communication attacks [2]. Whenever these effects occur on communication, stale data may be injected into the models contained in the vehicles. Stale data is data from a sensor that is not up-to-date. When stale data is present, a model is forced to compute on the last known correct values that it received, as communication has failed. It is therefore important to understand the effects imposed on transportation systems when stale data is introduced. To study these effects, we desire that our analysis technique is model agnostic, such that we do not have to understand the in-depth details of each model. We select two primary techniques to study the models.

The first technique that we employ is statistical fault injection, which allows the modeling of stale data injections into a model. Statistical fault injection is a classic technique in determining the robustness of systems perturbing certain components/data in the system to emulate a failure [3]–[5]. After injecting failures into a test model, the qualitative effects are measured, such as collisions of vehicles or deviations from the ideal route in the domain. The error can also be measured quantitatively by comparing the deviation of the simulation with the injection to the nominal simulation. This error can inform algorithm developers to design more robust solutions.

The second type of analysis that is performed is global sensitivity analysis. Global sensitivity analysis allows for the exploration of N-dimensional parameter space for a given model [6]. Through the exploration of the parameter space, the output behavior of the model is attributed to the inputs. In particular, a ranking is determined for which inputs are very influential to the sensitivity of the output of the model. This analysis helps determine which parameters need extra protection to increase the robustness and reliability of these models in the future. By combining these analysis techniques, it is possible to determine the impacts and magnitude of the effects of stale data, as well as determine a plan to improve the model safety in the future. This paper makes the following contributions.

- Develops an injection methodology to emulate stale data affecting a vehicle model in the Simulink environment.
- Performs analysis on the effects of injecting stale data faults into the model simulation.
- Generates a framework to automatically fit empirical data to probability distributions utilized in sensitivity studies.
- Evaluates the sensitivity analysis methodology on a vehicle test model to determine parameters that may need protection from stale data.

The rest of this paper is organized as follows. Section II presents background on fault injection and Sobol sensitivity analysis. We present and detail our sensitivity study, probability fitting pipeline, and injection methodology in Section III. Section IV evaluates and discusses the model behavior by injecting stale data and performing a sensitivity analysis. Section V discusses work related to the area of our study. Finally, we conclude in Section VI.

II. Background

A. Distribution Fitting

In order to encapsulate the characteristics of empirical data, it is often important to fit a probability distribution to the data. The probability distributions allow for the sampling of an input

Distribution A: Approved for public release; distribution unlimited. OPSEC #6861.

space that can be used for applications such as sensitivity analysis. This process is not straightforward, as the probability distribution must capture the details of the empirical distribution so that it provides an accurate model of the data. If the fitted distribution does not reflect the characteristics of the experimental data, the results of the analysis could lead to invalid conclusions. In order to determine if a probability distribution fits a set of empirical data, a method named the Kolmogorov–Smirnov test (KS-Test) can be utilized. The KS-Test utilizes the empirical cumulative distribution function (ECDF) of the empirical data, which is the total number of observations at or before the current point in the distribution divided by the total number of observations. The ECDF is compared to the cumulative distribution function(CDF) of the fitted distribution by the Kolmogorov-Smirnov statistic (KS-Stat). The KS-Test has two hypotheses, the null hypothesis is that the data follows a specified distribution, and the alternative hypothesis is that the data does not follow that distribution. The test statistic is defined by Equation 1 where $F_0(x)$ is the number of observations observed at or before the point x divided by the total number of observations and $F_r(X)$ is the accumulated probability of the distribution we are testing for fitting [7]. A smaller value of D indicates a better fit between the ECDF and the fitted CDF. The D value is then compared to the entry in the critical value table with the correct significance level and the number of observations in the empirical data. If the KS-Stat is less than the value in the table, then the null hypothesis is accepted that the specified distribution is a good fit for the data. If the KS-Stat is greater than the critical value, the null hypothesis is rejected, and the alternative hypothesis is accepted that the distribution does not fit. The KS-Test allows for many distributions to be tested for a set of data to determine a probability distribution that is well suited to the data.

$$D = max(F_0(X) - F_r(X)) \tag{1}$$

B. Statistical Fault Injection

Statistical fault injection is classically utilized to determine the dependability of an application or system when soft or hard errors such as bit-flips or crashes occur in the system [3]–[5]. When a fault occurs in a system, the nominal behavior of the system is perturbed. It is essential to have the ability to quantify not only the amount of error introduced by the fault, but also understand the qualitative effects of the fault. A fault injection campaign is performed by simulating the numerous occurrences of fault injection trials, where for each trial a unique fault is injected. Note that the behavior of a simulated fault should try to fully capture the qualities of a true fault for accurate analysis. Different types of faults may be modeled, such as a bit-flip occurring when a cosmic particle contacts computing hardware [3], [5] or when communication fails, and a model must continue to utilize the last known data. The reference behavior of the system is then compared to the behavior observed while the fault injection campaign is performed.

To quantitatively measure the effects of fault injection, we primarily observe the absolute and relative error. These absolute error gives the difference between the nominal simulation and the perturbed simulation and provides information about how much deviation has been introduced. The absolute error through time also indicates behavior such as convergence or divergence with respect to the error. It is important to understand whether the system can recover and converge back to the expected behavior, or if the system will diverge and become unstable when an error occurs. The relative error provides similar information as the absolute error but normalizes the scale such that it is easier to understand the magnitude of the error across the different scales of variables in systems. The qualitative effects are often seen by creating visualizations of the simulations with injected error. The qualitative effects are often closely tied to the physical phenomena that are represented by the application, such as a collision in a car model due to error.

C. Sobol Sensitivity Analysis

Sensitivity analysis defines the decomposition of uncertainty in the output of a model to the uncertainty in the inputs of a model [8]. By decomposing the uncertainty of the model and apportioning it to the inputs, a better understanding can be obtained of how the model transfers changes in the input to the output of the model. Ideal sensitivity analysis methods can provide quantitative measures of uncertainty to the inputs, such as variance [6]. The methods are also global and model-free, which allows the model to be treated as a black box and avoids assumptions such as model linearity and additive models. One downside of obtaining a quantitative measure is the cost of the analysis. Qualitative sensitivity analysis such as the Morris method [9] converges to a result much faster than variance-based sensitivity analysis such as the Sobol method [8]. The Sobol method often requires a large number of samples and computation resources to reach a result but provides quantitative rankings, which the Morris method cannot provide [6], [10].

Sobol sensitivity analysis is a technique within global sensitivity analysis that is variance based. The overall output variance of a model is attributed to the individual inputs and their interactions together within the model. The calculation is performed by varying individual variables to determine the first-order effects, and then combinations of variables to determine the higher-order effects. The first-order indices and N-order indices are the metrics that Sobol sensitivity defines to rank the importance of input factors [11], [12]. The equations for the first-order and N-order indices are described by Equation 3 and Equation 4. The D_t term is the total variance of the model, D_i and $D_i..n$ are the variance of an individual factor or combinations of factors, and S_i is the sensitivity index. The total order index provides a method to calculate the contribution of a factor i and all of its higher-order contributions in one metric to reduce computational complexity [11], [12]. In Equation 5, D_t is the total variance, $E_{X_{\sim i}}(D_i(Y|X_{\sim i}))$ is the expected value of the variance of all other variables and combinations in the model excluding the variable we are evaluating. By subtracting from one, we are left with the variance contribution of the ith variable. This metric attributes a fraction of the total variance to a given input or combination of inputs. The contributions of each input are estimated by the decomposition of the model given in Equation 2 where each summation term describes the contribution of individual variables, two variables combined, and so forth until the contribution of varying all variables is calculated. By attributing the variance from input to output, we obtain more understanding of the model's behavior.

$$D(f) = \sum_i D_i + \sum_{i<j} D_{ij} + \sum_{i<j<k} D_{ijk} + ... + D_{1...n} \tag{2}$$

$$S_i = \frac{D_i}{D_t} \tag{3}$$

$$S_{i...n} = \frac{D_{i...n}}{D_t} \tag{4}$$

$$S_{Ti} = 1 - \frac{E_{X_{\sim i}}(D_i(Y|X_{\sim i}))}{D_t} \tag{5}$$

III. Methods

The model analysis framework that we build consists of two primary studies which provide information about the behavior of the current model under test. The first analysis of statistical fault injection provides information about the effects that occur when stale data is injected into a simulation of the model. The second analysis that is performed is global sensitivity analysis, which is utilized to encapsulate information regarding the importance of a given input to the output of the model. In order to support the first analysis, an injection system is built into the Simulink environment. The supporting systems for global sensitivity analysis require the model simulation, the generation of probability distributions to model the inputs, and the actual global sensitivity calculations. The block diagram detailing the workflow is shown in Figure 1 and Figure 2. Figure 1 describes the entire pipeline for calculating sensitivity coefficients by generating the empirical data, creating probability distributions from the data, utilizing the distributions to sample the parameter space, and performing sensitivity analysis on those samples. Figure 2 shows a more abstracted view of this process, as we simply create the inputs and treat the sensitivity analysis as a black box to produce the coefficients. These two methods of analysis help identify which parts of the model may need to be protected from effects such as stale data.

A. Statistical Fault Injection Methods

In order to perform the fault injection into the models, we must implement a process to emulate stale data and a method to perform of stale data injections at various time and locations. Since many vehicle models are simulated in the Matlab/Simulink environment, we implement a custom Simulink block written in Matlab. The Simulink block is placed on the input signals to a model in order to mock stale data injections. The block enables injections to be performed at any time-step in the simulation, which allows us to understand how a stale data injection affects a model simulation. The injection functionality also enables the stale data injection to be repeated multiple times, which helps indicate if the error continues to accumulate or if the error diminishes and converges. The length of the injection is configurable such that, once again, the convergence of the error can be determined. The injection consists of forcing the model to continuously compute the last known valid value from an input signal. This coincides with the definition of stale data such that the model no longer has access to the most updated information, whether that is due to sensor failure or an attack on a communication system.

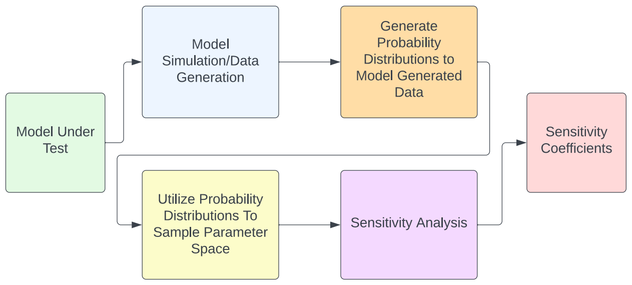

Model Under Test
Model Simulation/Data Generation
Generate Probability Distributions to Model Generated Data
Utilize Probability Distributions To Sample Parameter Space
Sensitivity Analysis
Sensitivty Coefficients

Fig. 1: Overall global sensitivity workflow.

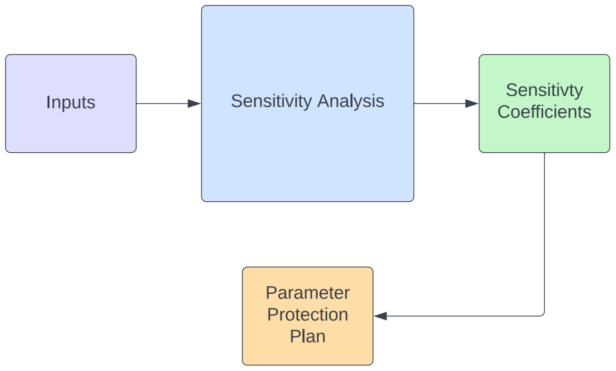


Fig. 2: Black box view to calculate sensitivity coefficients.

B. Global Sensitivity Input Modeling

In order to perform the global sensitivity analysis on the model that we are studying, the inputs for the sensitivity analysis must be probability distributions. In order to generate probability distributions that are fed into the sensitivity analysis framework, we utilize the data collected from the nominal simulation of the model. We then employ automatic probability distribution fitting to avoid performing analysis on

TABLE I: Supported Distributions in Probability Fitting Framework

Supported Distributions	
Exponential	ExtremeValue
Gamma	GeneralizedExtremeValue
GeneralizedPareto	HalfNormal
InverseGaussian	Logistic
Loglogistic	Lognormal
Nakagami	NegativeBinomial
Normal	Poisson
Rayleigh	Rician
Stable	tLocationScale
Weibull	Uniform

each individual input. We support the distributions that are shown in Table I.

In order to determine whether a distribution is well suited to the empirical data collected in the simulation, we first generate the empirical CDF, which describes the exact probability of the data collected as we accumulate samples from the empirical data across the input range of the variable. We then check through all of our supported distributions in Table I to determine which distribution best captures the characteristics of the empirical data. The determination for which distribution fits the best is performed with the Kolmogorov–Smirnov test (KS-Test). The KS-Test finds the maximum distance between the empirical distribution and the fitted distribution. We attempt to minimize the KS-Test statistic, which is the maximal distance between the fitted and the empirical distributions across the set of distributions.

C. Global Sensitivity Sobol Sensitivity Analysis

In order to perform global sensitivity analysis, we select the Sobol method, which allows the total variance in the model to be decomposed such that it is attributed to the inputs. In our work, we utilize the global sensitivity Matlab toolbox named GSAT [11], [13]. The toolbox supports two different sensitivity methodologies, Sobol sensitivity analysis, which we utilize in this study, and Fourier Amplitude Sensitivity Testing (FAST). We select the Sobol method due to the less restrictive assumptions about the model, as FAST imposes extra conditions (e.g. model smoothness). Moreover, the Sobol method is a black box methodology, as seen in Figure 3. This allows us to analyze and determine better methods to protect models without having to know and understand the intricacies of each individual model.

The Sobol method ingests the probability distributions for each model input. The inputs are then sampled utilizing Monte Carlo sampling based on the probabilities associated with the variable obtaining a particular value. This allows us to better capture the behavior of the model by ensuring that each input is likely to take on values that occur frequently during model simulations. Two sets of samples are generated, which allows a base set to be utilized to capture the total variance of the model. The capture of the baseline variance or total variance of the model allows us to obtain a concise ranking of the sensitive inputs at the end of the analysis. The variance of each individual input is divided by the total variance, which means every input gets a sensitivity ranking between zero and one. Once the baseline variance is established with the initial sample set, the decomposed variance calculations begin. In particular, there are two types of calculations performed in this section. The first type is a first-order calculation that computes the variance of a single input variable by itself. This is important as it provides an understanding of how the model reacts if you change a single input value. The second type of calculation that is performed encapsulates the sensitivity of the interactions between multiple variables. This provides an interpretation of how the model behaves when multiple input values change simultaneously. All the higher-order effects are calculated from just varying two inputs simultaneously to varying all the inputs simultaneously. Once the parameter space exploration is completed, the analysis outputs the ranking of which variables are most sensitive to a given output of the model. The Sobol method in the toolbox supports models that have multiple output variables, which allows a wider range of models to be tested. This case is handled by running multiple sensitivity analyses in parallel. The results from the global sensitivity simulation can then be further utilized in the future to understand which input parameters may need further protection from effects such as stale data that we explored in the fault injection studies.

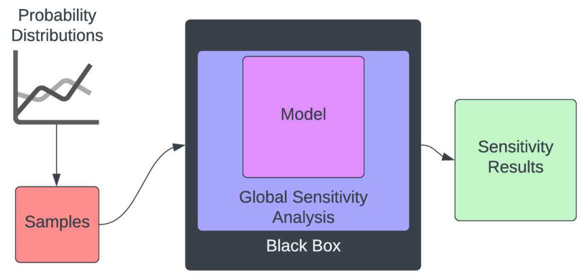
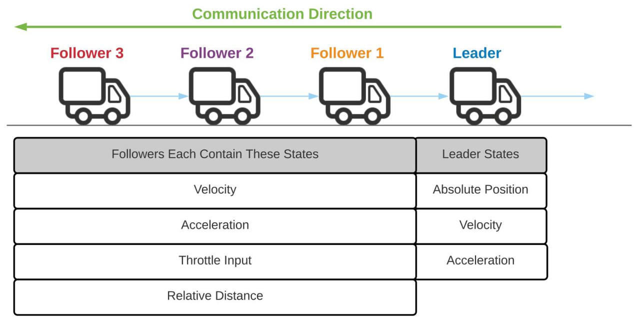


Fig. 3: Black box methodology encapsulated within the Sobol sensitivity analysis.

IV. Experimental Results

A. Model Definition

For our experiments, we investigate a one-dimensional platooning model that has a parameterizable number of vehicles in the software environment. We run all experiments using Simulink version 10.2 and Matlab 9.9 R2020b. In our examples, we utilize four vehicles for the stale data injections and perform the sensitivity analysis with two vehicles. A visualization of the model under test is shown in Figure 4. The model has a leader vehicle that sets the pacing of the platoon. In particular, the leader vehicle has an absolute position, velocity, and acceleration as its internal parameters. The vehicles that follow the leader all contain the same set of parameters and are essentially copies of the other following vehicles. The following vehicles contain velocity, acceleration,



Followers Each Contain These States	Leader States
Velocity	Absolute Position
Acceleration	Velocity
Throttle Input	Acceleration
Relative Distance	

Fig. 4: 1D platooning test model parameters for each vehicle type and the communication pattern.

throttle input, and relative distance as their internal states. The following vehicles attempt to maintain a specified relative distance between themselves and the vehicle directly in front of them. The controls systems regulate this distance by adjusting the velocity, acceleration, and throttle input parameters. The target relative distance in our experiments is set to the value of 20 meters. The other two notable parameters for this framework are the velocity ramps of the leader. The velocity ramps are inputs to the model that ramp up the velocity of the leader vehicle to set the velocity for the platoon formation. The number of outputs for the model is a function of the number of vehicles in the simulation and is described by the following formula, $Outputs = 3 + 4 * (NumVehicles - 1)$ since the leader vehicle has 3 parameters and each following vehicle has four parameters. Each individual output variable is a linear model that performs a sum of products from the inputs, where each input is multiplied by a coefficient set by the control system. We run each simulation for 100 time-steps.

B. Statistical Fault Injection

We perform a statistical fault injection to further understand the qualitative and quantitative effects of stale data imposed on our model under test. Figure 5 shows an injection performed into the velocity variable of the first following vehicle in a model simulation. A 10-second-long injection is performed at time-step 50 that forces the model to compute with the last known valid value in the simulation. Once the stale data injection is completed, we immediately see the model attempt to perform a correction as the velocity signal swings rapidly to adjust to the error that was just injected. One other effect we see in Figure 5 is that the error eventually converges back toward the nominal simulation in which no error occurs. This is expected since the control system should damp out the error eventually, but it is important to understand the effects that occur quickly after the initial error injection. Figure 6a and Figure6b show variables from the same simulation in which stale data is injected into the velocity variable. We see, the error propagates to the other vehicles states in the model as the system tries to correct for the incorrect and erroneous behavior of the vehicle suffering the stale data injection. However, the

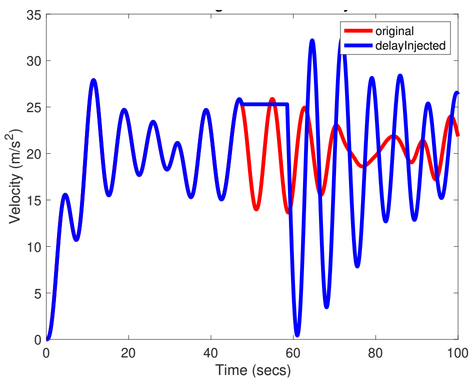
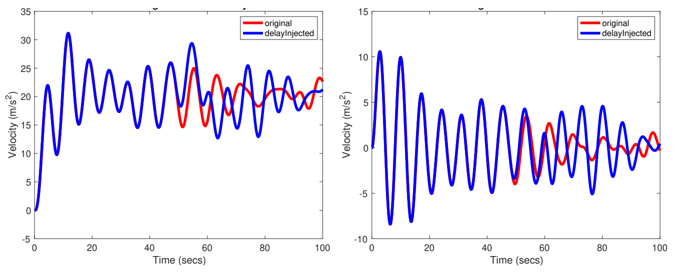


Fig. 5: Deviation of the velocity after a stale data injection.



(a) Velocity.

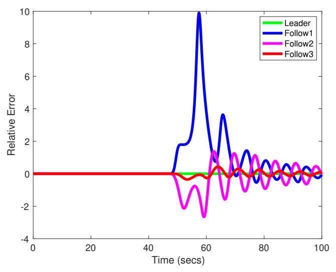
(b) Acceleration.

Fig. 6: Propagation and deviation of stale data between two variables. Fault injection occurs in Velocity.

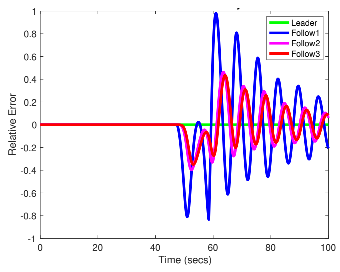
magnitude of the error decreases as the number of vehicles between the one suffering the stale data injection increases. Once again, we see the control system eventually converging and reducing the error in the states. Figure 7a and Figure 7b show the relative error across the vehicles once the stale data injection is performed in the first vehicle at time-step 50. Once again, we note the converging error and the reduction in error propagated backward through the vehicles. In Figure 7a, we see a very large impact on the first vehicle's relative position, reaching a relative error of 10 or an error ten times the measurement of the nominal simulation value. This is a very large error deviation and provides a substantial effect on the shape and safety of the platoon. The velocity variable receives less overall relative error and only reaches a value of one for the relative error of the first following vehicles (where the error is injected). The largest effect is seen in the position of the vehicles when the stale data injection is performed.

C. Distribution Fitting

In order to properly model the inputs for global sensitivity analysis, we fit probability distributions to our empirical simulation data by utilizing the KS-test. Figures 8a-8f show the fit CDF distributions to the empirical CDF distributions and the corresponding PDF function. Based on all the CDF figures, we see that the empirical data's characteristics are well encapsulated. The empirical data cumulative distributions have some slight variations compared to the fitted distributions, but the deviations are small enough that they pass the goodness



(a) Relative error: Position



(b) Relative error: Velocity

Fig. 7: Impact of stale data on accuracy of each car's Position and Velocity. The platooning model naturally damps error with time.

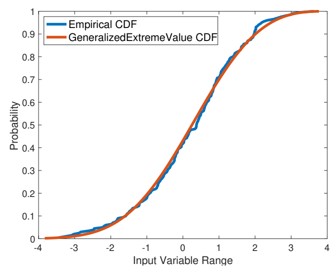
Variable Name	p-value ($p < 0.01$)	KS-Stat
Leader Acceleration	0.707	0.0487
Following Vehicle 2 Throttle Input	0.842	0.0426
Following Vehicle 3 Relative Distance	0.390	0.0627

TABLE II: Probability Distribution p-value and KS-Stat

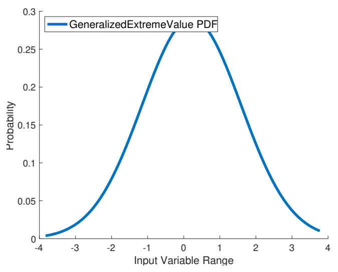
of fit tests. Table II shows the KS-Stat and p-values for the distributions plotted in Figures 8a-8f. We see in Table II that the following vehicle 2's throttle input variable fits the best out of these variables, with a p-value of 0.842 and KS-Stat of 0.0426. The KS-Stat is the smallest in the table and indicates the minimum deviation from the logistic distribution that is fit to the variable. The following vehicle 3's relative distance variable is the worst fit, with a p-value of 0.390 and a KS-Stat of 0.0627. We see in Figure 8e at the 20–30 input value range, there is a deviation that the fitted stable distribution does not entirely capture. However, as a whole, it captures enough of the empirical information relatively well.

D. Sensitivity Results

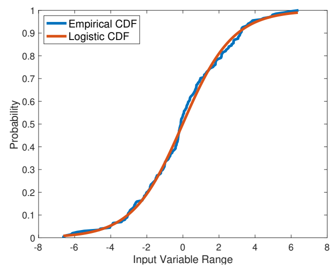
The first-order results of the Sobol sensitivity analysis are shown in Figure 9a. Figure 9b shows the coefficients relating the inputs of the model to a given output. For Figure 9a, the rows of the heat map show inputs and the columns are outputs. A higher value in a cell indicates that the variable on that row contributed substantially to the output in the corresponding column. We note that many of the variables are not related at all, which is indicated by a zero in a given heat map cell. We note the strong diagonal, where each variable is highly related to its previous value. This is due to the control system's feedback loop utilized within the model. We, therefore, expect variables to be quite sensitive to changes in their own value, which is shown by the strong diagonal pattern in the first-order indices in Figure 9a. Another important feature is the sensitivity of the linear velocity ramps. We once again see that the coefficients on the linear model indicate that the values of the velocity ramps are important to the value of throttle input for following vehicle one and the acceleration of the leader vehicle. This is again captured by the sensitivity analysis, as the velocity ramps obtain first-order sensitivity indices of 0.45 for the leader acceleration and 0.34 or 0.33



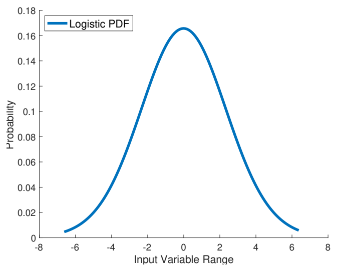
(a) Following Vehicle 1 Acceleration CDF.



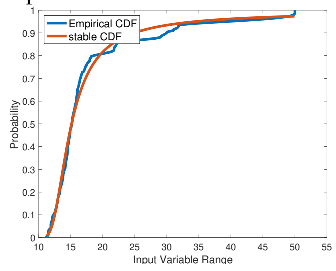
(b) Following Vehicle 1 Acceleration PDF.



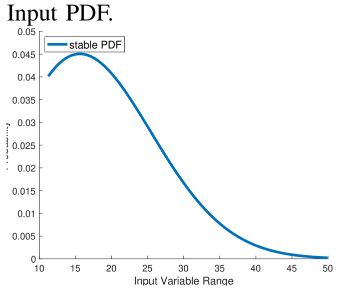
(c) Following Vehicle 2 Throttle Input CDF.



(d) Following Vehicle 2 Throttle Input PDF.



(e) Following Vehicle 3 Relative Distance CDF



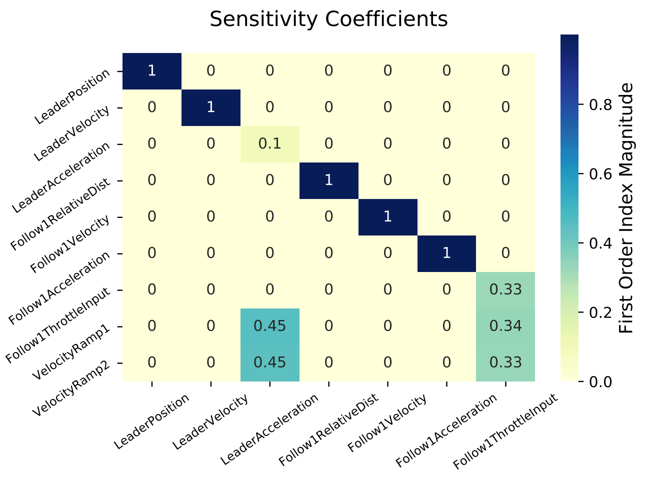
(f) Following Vehicle 3 Relative Distance PDF.

Fig. 8: Probability distributions fit to empirical simulation data.

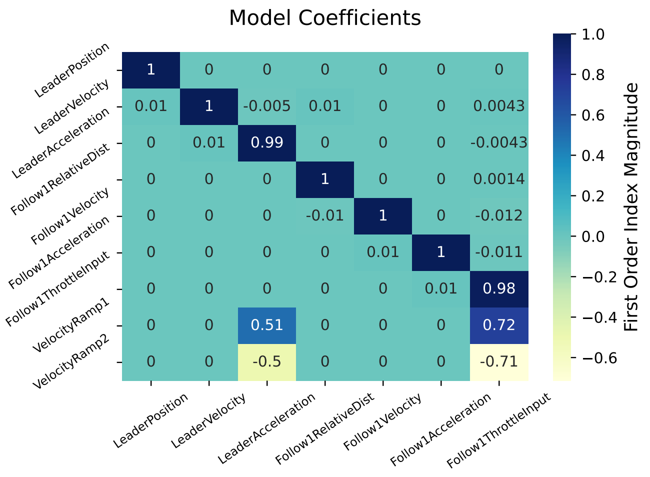
for the throttle input of the first following vehicle. While this method provides valuable information about the importance of parameters in the model, the Sobol method is computationally expensive in terms of memory and time. As the number of parameters in the model grows, the number of possible inputs increases exponentially, which requires a large amount of storage. This means we need an increasing number of samples to sufficiently explore the parameter space, and we also have to vary an exponentially increasing number of combinations. In our experiments, the full model contains 21 parameters. We estimate the compute time to run the Sobol sensitivity analysis to take weeks to converge sufficiently. To make execution time, tractable, we prune the parameter space selecting 9 parameters, which reduces runtime to a few several hours. Selecting fewer parameters does reduce the power of the Sobol analysis as not all sensitivity indices have fully converged, but as the number of parameters approaches the maximum, results fully converge.

V. Related Works

Le et al. [14] applies sensitivity analysis to autonomous vehicle models in order to determine the effects on three metrics they define. In particular, they investigate throughput,



(a) First Order Sensitivity Indices



(b) Linear Model Coefficients

Fig. 9: Sensitivity of inputs/states (y-axis) to outputs (x-axis).

passenger safety, and comfort but do not investigate platooning models and how communication may impact those systems.

Qiao et al. [15] apply global sensitivity analysis in order to determine which parameters are important to traffic simulation models in the domain of intelligent transportation. In particular, they determine which parameters must be selected carefully and which parameters may be discarded. The authors stick to traffic topics specifically, such as lane change distance and stop distance, within their study.

Monteil et al. [16] apply global sensitivity analysis to car-following traffic models. The authors utilize global sensitivity techniques to disregard insensitive parameters, but employ other statistical techniques to perform parameter estimations. While we both utilize global sensitivity analysis on a leader-follower based model, the end goal of the analysis diverges into different domains in intelligent transportation. The authors apply their results to traffic modeling, while we apply our results to determine which parameters may need to be protected in models from stale data.

Vepsäläinen et al. [17], and Fiori et al. [18] both utilize Sobol sensitivity analysis techniques to understand which parameters are highly important for energy consumption in intelligent transportation. Vepsäläinen et al. [17], in particular, investigate models concerning the energy utilization of buses in the city. While we also apply Sobol sensitivity analysis to vehicle models, we focus on the controls models for platooning instead of energy modeling in our work.

Gallo et al. [19] investigate virtual coupling for variable train composition by utilizing sensitivity analysis to determine the effects of transport demand, passengers behavior, and boarding trains. The authors seek to determine which parameters should be modeled to higher fidelity. This work focuses on which parameters need better protection from external effects.

Jha et al. [20] utilize a machine learning Bayesian fault injector which seeks to find a safe situation that becomes unsafe after a fault is injected. The authors must utilize causal and counterfactual reasoning to estimate the behavior of the autonomous driving system under the effects of a fault. They inject faults into parameters such as the throttle and observe how the autonomous driving system responds. We also inject faults and observe whether an unsafe condition occurs. We target platooning models in our analysis instead of individual autonomous driving systems. Additionally, our faults are modeled after the effects of stale data by continuing with the last known valid value, while the authors of this work directly modify the value of a parameter to a new value to simulate a fault.

Pisu et al. [21] model faults in a platoon using cruise control to maintain the distance between the vehicles. They implement an observer to attempt to mitigate faults from the sensors. The observer detects faults using the residual. While these authors focus on detecting faults in sensors, we focus on injecting faults and determining the magnitude of the effects. These techniques may be helpful in the future to protect parameters that we identify as sensitive.

VI. Conclusion

Overall, we can successfully implement our two analysis methods in a method that allows intelligent transportation models to be analyzed seamlessly. With the model that we tested, we found that stale data can cause drastic negative effects such as collisions with other vehicles through statistical fault injection. Additionally, we saw the convergence of the introduced error, but an initial deviation spike in some parameters was ten times the value of the nominal simulation. The global sensitivity analysis was also successfully able to reflect the expected sensitivity of our simple linear model. We additionally note that the Leader Acceleration and Following Vehicle 1Throttle Input parameters should have the velocity ramps protected, as they are sensitive in particular to those inputs. The rest of the values should primarily have their feedback value protected, as they are most sensitive to that value. The global sensitivity analysis provides valuable information about models, but it is a resource-intensive method both in terms of memory and computation time. Therefore, this technique is best suited for evaluating models with smaller sets of input parameters as with each increasing input, the dimensions of the search space increase.

Acknowledgments

Clemson University is acknowledged for generous allotment of compute time on the Palmetto cluster. This material is based upon work supported by the National Science Foundation under Grant No. SHF-1910197 and SHF-1943114. Acknowledgment: This work was supported by the Simulation Based Reliability and Safety Program for modeling and simulation of military ground vehicle systems, under the technical services contract No. W56HZV-17-C-0095 with the U.S. Army DEVCOM Ground Vehicle Systems Center (GVSC).

References

[1] N. O'Mahony, S. Campbell, A. Carvalho, S. Harapanahalli, G. A. Velasco-Hernández, D. Riordan, and J. Walsh, "Adaptive multimodal localisation techniques for mobile robots in unstructured environments : A review," in *2019 IEEE 5th World Forum on Internet of Things (WF-IoT)*, 2019, pp. 799–804.

[2] M. Dibaei, X. Zheng, K. Jiang, R. Abbas, S. Liu, Y. Zhang, Y. Xiang, and S. Yu, "Attacks and defences on intelligent connected vehicles: a survey," *Digital Communications and Networks*, vol. 6, no. 4, pp. 399–421, 2020. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S235286481930197X

[3] J. Calhoun, L. Olson, and M. Snir, "FlipIt: An LLVM Based Fault Injector for HPC," in *Euro-Par 2014: Parallel Processing Workshops*, L. Lopes, J. Žilinskas, A. Costan, R. G. Cascella, G. Kecskemeti, E. Jeannot, M. Cannataro, L. Ricci, S. Benkner, S. Petit, V. Scarano, J. Gracia, S. Hunold, S. L. Scott, S. Lankes, C. Lengauer, J. Carretero, J. Breitbart, and M. Alexander, Eds. Cham: Springer International Publishing, 2014, pp. 547–558.

[4] P. Ramachandran, P. Kudva, J. Kellington, J. Schumann, and P. Sanda, "Statistical fault injection," in *2008 IEEE International Conference on Dependable Systems and Networks With FTCS and DCC (DSN)*, 2008, pp. 122–127.

[5] R. Leveugle, A. Calvez, P. Maistri, and P. Vanhauwaert, "Statistical fault injection: Quantified error and confidence," in *2009 Design, Automation Test in Europe Conference Exhibition*, 2009, pp. 502–506.

[6] A. Saltelli, Ed., *Sensitivity analysis in practice: a guide to assessing scientific models*. Hoboken, NJ: Wiley, 2004.

[7] "Kolmogorov–Smirnov Test," in *The Concise Encyclopedia of Statistics*. New York, NY: Springer New York, 2008, pp. 283–287. [Online]. Available: https://doi.org/10.1007/978-0-387-32833-1_214

[8] I. M. Sobol, "On sensitivity estimation for nonlinear mathematical models," *Matem. Mod.*, vol. 2, pp. 112–118, 1990.

[9] M. D. Morris, "Factorial sampling plans for preliminary computational experiments," *Technometrics*, vol. 33, no. 2, pp. 161–174, 1991. [Online]. Available: http://www.jstor.org/stable/1269043

[10] A. Saltelli, M. Ratto, and T. Andres, *Global sensitivity analysis: The Primer*. Wiley, 2008.

[11] J. Nossent, P. Elsen, and W. Bauwens, "Sobol' sensitivity analysis of a complex environmental model," *Environmental Modelling & Software*, vol. 26, no. 12, pp. 1515–1525, 2011. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1364815211001939

[12] D. Kumar, A. Singh, P. Kumar, R. K. Jha, S. K. Sahoo, and V. Jha, "Sobol sensitivity analysis for risk assessment of uranium in groundwater," *Environmental Geochemistry and Health*, vol. 42, no. 6, pp. 1789–1801, Jun. 2020. [Online]. Available: http://link.springer.com/10.1007/s10653-020-00522-5

[13] flax, "Global sensitivity analysis toolbox," https://www.mathworks.com/matlabcentral/fileexchange /40759-global-sensitivity-analysis-toolbox. MATLAB Central File Exchange. Retrieved March 20, 2022., 2022.

[14] M. L. Maitre and A. Prorok, "Effects of controller heterogeneity on autonomous vehicle traffic," in *2020 IEEE 23rd International Conference on Intelligent Transportation Systems (ITSC)*, 2020, pp. 1–6.

[15] Q. Ge, B. Ciuffo, and M. Menendez, "An exploratory study of two efficient approaches for the sensitivity analysis of computationally expensive traffic simulation models," *IEEE Transactions on Intelligent Transportation Systems*, vol. 15, no. 3, pp. 1288–1297, 2014.

[16] J. Monteil and M. Bouroche, "Robust parameter estimation of car-following models considering practical non-identifiability," in *2016 IEEE 19th International Conference on Intelligent Transportation Systems (ITSC)*, 2016, pp. 581–588.

[17] J. Vepsäläinen, K. Otto, A. Lajunen, and K. Tammi, "Computationally efficient model for energy demand prediction of electric city bus in varying operating conditions," *Energy*, vol. 169, pp. 433–443, 2019. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0360544218324307

[18] C. Fiori, V. Marzano, V. Punzo, and M. Montanino, "Energy consumption modeling in presence of uncertainty," *IEEE Transactions on Intelligent Transportation Systems*, vol. 22, no. 10, pp. 6330–6341, 2021.

[19] F. Gallo, A. Di Febbraro, D. Giglio, and N. Sacco, "Global sensitivity analysis for the evaluation of the effects of uncertainty of transport demand and passenger behavior on planning railway services with variable train composition," in *2021 IEEE International Intelligent Transportation Systems Conference (ITSC)*, 2021, pp. 2298–2305.

[20] S. Jha, S. Banerjee, T. Tsai, S. K. S. Hari, M. B. Sullivan, Z. T. Kalbarczyk, S. W. Keckler, and R. K. Iyer, "Ml-based fault injection for autonomous vehicles: A case for bayesian fault injection," in *2019 49th Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN)*, 2019, pp. 112–124.

[21] Z. A. Biron and P. Pisu, "Distributed Fault Detection and Estimation for Cooperative Adaptive Cruise Control System in a Platoon," *Annual Conference of the PHM Society*, vol. 7, no. 1, 2015, number: 1. [Online]. Available: https://papers.phmsociety.org/index.php/phmconf/article/view/2545