2021-01-0896 Published 06 Apr 2021

Pycrash: An Open-Source Tool for Accident Reconstruction

Joe Cormier and James Funk Biocore LLC

Gray Beauchamp and David Pentecost Kineticorp LLC

Citation: Cormier, J., Funk, J., Beauchamp, G., and Pentecost, D., "Pycrash: An Open-Source Tool for Accident Reconstruction," SAE Technical Paper 2021-01-0896, 2021, doi:10.4271/2021-01-0896.

Abstract

Accident reconstructionists routinely rely on computer software to perform analyses. While there are a variety of software packages available to accident reconstructionists, many rely on custom spreadsheet-based applications for their analyses. Purchased packages provide an improved interface and the ability to produce sophisticated animations of vehicle motion but can be cost prohibitive. Pycrash is a free, open-source Python-based software package that, in its current state, can perform basic accident reconstruction calculations, automate data analyses, simulate single vehicle motion and, perform impulse-momentum based analyses of vehicle collisions. In this paper, the current capabilities of Pycrash are illustrated and its accuracy is assessed using matching PC-Crash simulations performed using PC-Crash. The results indicate that Pycrash is well-equipped to perform fundamental accident reconstruction analyses, including impact related effects, but its simplified suspension model is a limitation. It is hoped that others within the accident reconstruction community will use Pycrash, make suggestions for its improvement and develop additional capabilities.

Introduction

Accident reconstructionists rely on computer software to perform calculations and analyses related to their work. Whether it is a spreadsheet-based customized solution or a purchased software package, accident reconstruction analyses are founded in fundamental physics and peer-reviewed accident reconstruction techniques. While purchased packages provide turn-key solutions for accident reconstruction analyses and visualization, they do not provide freedom to adapt the calculations or output, are limited to the functionality provided by that piece of software, and can be cost prohibitive.

Reconstruction software packages like VCRware, Virtual Crash, PC-Crash and HVE determine planar vehicle motion based on inputs such as braking and steering and the resulting tire reaction forces. The tire model is a critical component in predicting vehicle motion and there are several formulations that determine the forces applied by the tire as a function of slip angle, tire properties, and suspension characteristics [8, 30, 34, 9, 14]. The application of purchased software packages like PC-Crash has been well-documented and extensively evaluated under various conditions [25] which is a benefit of purchasing a reconstruction software package.

While lacking a graphical interface, mathematical models of vehicle motion consisting of varying degrees of complexity have been developed and their results published. However these custom-built solutions are not made available to the public and, therefore, only provide utility for the owner [24, 36].

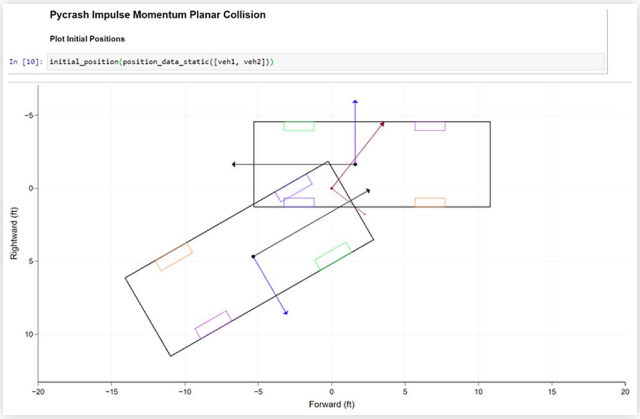
While accident reconstruction is not a novel science, the mathematical methods are evolving, and data sources continue to expand. An open source tool allows users to change the software functionality to fit a specific application and utilize new sources of data as they become available. Additionally, the details of the calculations can be explored and visualized to the desired level of detail when the source code is accessible. Finally, open source is free and can be improved upon by anyone. The purpose of this paper is to introduce the Pycrash software package and illustrate its current capabilities.

Methods

Pycrash was written in Python (Python Software Foundation, www.python.org), an open source, object-oriented language. There is a wide range of open source, scientific packages written in Python that provide a means to process, visualize and analyze data well beyond the typical spreadsheet or accident reconstruction software. Pycrash makes extensive use of the Pandas [21], Numpy [18], Scipy [35] and Plotly [33] packages for processing data, calculations and visualization.

Pycrash provides the user with callable functions that create vehicle objects to be analyzed, perform calculations and create visualizations. In addition to running scripts from a basic Integrated Development Environment (IDE), Python

FIGURE 1 Interacting with Pycrash using Jupyter Notebook and plot of vehicle initial positions for impulse-momentum simulation indicating location of impact point and impact plane orientation



can be run within a Jupyter Notebook which provides an interactive document that can contain text, code and visualizations (Figure 1, Appendix 1) [19]. The analyses presented here were written in Jupyter Notebooks to serve as a tutorial. The data produced by Pycrash is stored in a dataframe, so that it can be easily accessed by the user outside of Pycrash. All the data produced by Pycrash can be readily saved in the common .csv or other spreadsheet formats.

Pycrash is composed of modules that perform isolated tasks within the overall package. This facilitates collaboration, so that different methods for analyses can be built as separate modules and then called upon by the main code. There are basically three main categories of applications within Pycrash: 1. Fundamental equations and methods for general reconstruction, 2. Planar vehicle motion and, 3. Planar impact mechanics. The goal of this paper is to illustrate the current functionality of Pycrash along with a comparison to PC-Crash simulations and crash test data to illustrate its capabilities and provide validation data for future use.

Fundamental Calculations

Pycrash can be used to perform various fundamental calculations as you would use a spreadsheet application. These modules allow the user to evoke equations described in accident reconstruction publications for tasks such as deriving A and B stiffness values from crash test data and solving for closing speeds in using impulse-momentum equations. Additionally, processing the load cell and acceleration data from the ascii files obtained from the National Highway Traffic Safety Administration (NHTSA) Crash Test Database is automated. Pycrash users can take advantage of higher-level open-source scientific packages (e.g., Scipy) to perform a wide range of analyses on the results produced by Pycrash.

Vehicle Motion

Pycrash will simulate vehicle motion when provided initial conditions and driver inputs (brake, steer, throttle). Tire forces are calculated using a linear model as described by Steffan et al. (1996) for PC-Crash [30]. Ackerman steering is used to determine the steer angle of each tire based on a prescribed steering wheel angle and steering ratio [17]. Weight-shift due to longitudinal and lateral acceleration is accounted for using a simple pendulum model based on a prescribed center of gravity (CG) height. The CG height within Pycrash is the height at which the vehicle mass acts rigidly with the vehicle chassis. Therefore, the greater the assumed CG height, the more weight shift will be applied for a given level of vehicle acceleration. This differs from an actual vehicle with suspension in which the CG height also has meaning in terms of its distance to the roll center of the vehicle. The planar vehicle motion predicted by Pycrash in its current state was evaluated using several simulations generated using PC-Crash.

PC-Crash is an accident reconstruction software program that has been validated for simulating vehicle dynamics and vehicle impacts [25]. This software uses a discrete time step, forward integrating model to simulate vehicle motion. PC Crash computes tire forces using, among of other things, vertical loads, suspension effects, slip rates, topography, and available friction. These forces are then applied to the vehicle by way of its inertial mass properties. For the comparison simulations with Pycrash, the linear tire model was used. This model assumes a linear growth of tire cornering forces using a slope set by the program at 0.1 g per 1 degree of slip. This growth saturates at the user defined global friction value. Throughout the comparison simulations, the Kinetics simulation model at a 5 ms integration time step was used. Simulations were conducted assuming flat terrain, and global friction was set to 0.76 g. All vehicle motion simulations were performed with a time step of 0.01 seconds.

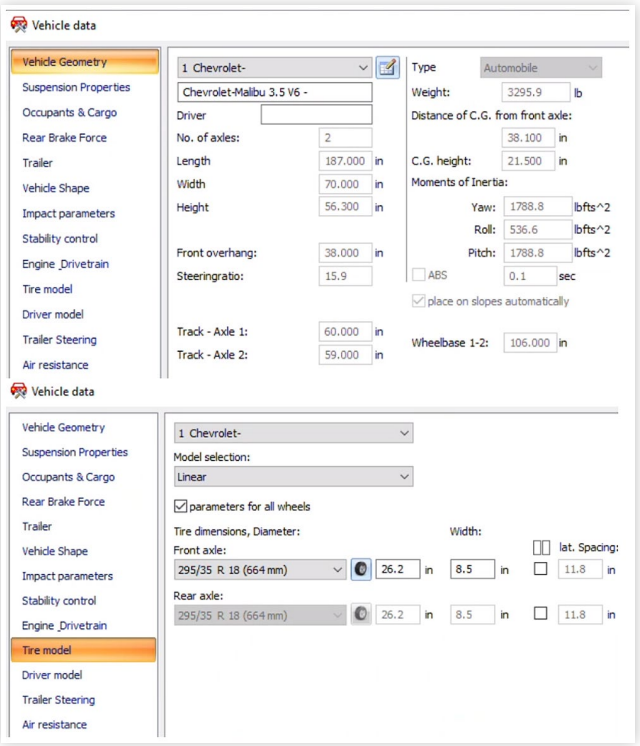
The specifications for the simulated vehicle were based on the data measured from a 2004 model year Chevrolet Malibu LT (VIN - 1G1ZU54854F135916) (Figure 2) that was previously testing in various steering maneuvers [3]. This test vehicle was equipped with a 3.5-liter, 6-cylinder gasoline engine and a four-speed, front-wheel-drive automatic transmission.

The same vehicle geometric and inertial properties were used within PC Crash and Pycrash for the comparative simulations (Figure 3).

Two limit maneuvers were simulated using PC-Crash; a single steer maneuver and a fishhook maneuver. For the single steer maneuver, the simulated vehicle traveled straight at a speed of 15 and 30 mph, and a 360-degree leftward steering input (at the steering wheel) was generated over one second and held.

FIGURE 2 Kineticorp 2004 Chevrolet Malibu Test Vehicle

FIGURE 3 PC Crash vehicle inputs



The second type of simulation involved performing a NHTSA defined "Fishhook test," This procedure is titled "Laboratory Test Procedure for Dynamic Rollover - The Fishhook Maneuver Test Procedure" and is a part of the New Car Assessment Program [2]. This test uses a vehicle specific steering angle which will produce 0.3 g of lateral acceleration (Figure 4). The full procedure includes tire scrubbing protocols, and methods to repeatedly and safely determine this steering angle without jeopardizing the test driver. The test was being performed with a simulator, so much of these auxiliary protocols were not needed and were not performed.

A preliminary simulation was performed at 50 mph to determine the steering angle required to produce 0.3 g of lateral acceleration. The method outlined in the "3.1.2. Maneuver Description (Option #2, Preferred)" was used (2). This steer angle was then multiplied by the Steering Scalar factor of 6.5 to arrive at the Fishhook steering maneuver steering angle (steering angle A in Figure 4). This steering angle was input at 720 deg/sec and held until the roll rate of the vehicle went below 1.5 deg/sec (T1). A counter-steer of the same magnitude was then applied, again at 720 deg/sec and then held for 3 seconds (T2). The steering wheel was then returned to 0 degrees over 2 seconds (T3) (Figure 5). Fishhook simulations were run at 35, 40 and 45 mph. These test speeds were used as 35 mph produced a plow out and the vehicle spun out at 45 mph in PC-Crash. The effects of CG height were explored using the simulations of the fishhook maneuvers. During development of the Pycrash vehicle model, lowering the vehicle CG height was found to improve its response during more demanding steering maneuvers. To evaluate the effects of CG height in the more demanding fishhooks tests,

FIGURE 4 NHTSA fishhook steering plot

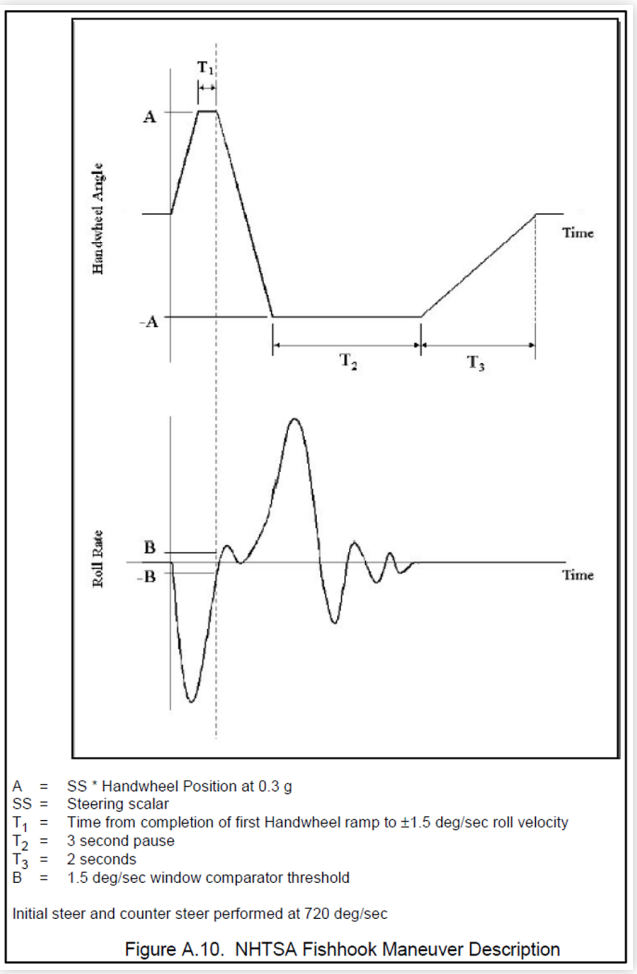
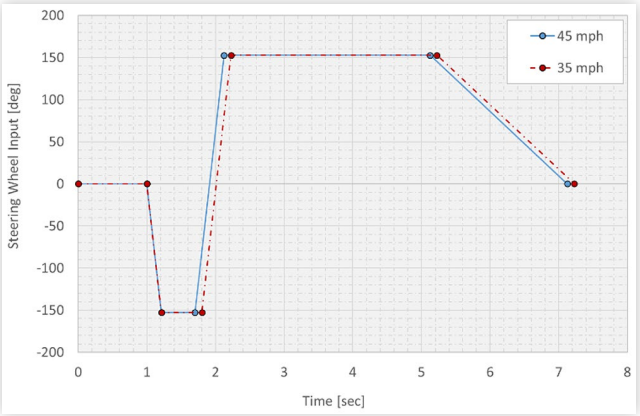


FIGURE 5 Simulated NHTSA Fishhook steering plot



a height of 1.8 ft, based on vehicle specifications and 1 foot were used as inputs. Since the Pycrash vehicle model does not currently have a suspension, altering the simulated CG height is a way to affect the overall response of the vehicle and, therefore, was evaluated here.

Impact Simulation

Planar impacts can be simulated using two methods in Pycrash. For 1-dimensional analyses, single degree of freedom models can be used to determine the velocity changes of the vehicles and the crash pulse. For 2-dimensional analyses, such as intersection collisions, conservation of momentum methods can be used to determine the translational and rotational velocity changes of the vehicles. These methods are outlined below along with initial evaluations using PC-Crash simulations and crash test data.

Single Degree of Freedom Model Fundamentally, damaged-based analyses of impact severity are based on the mutual stiffness between the impacting vehicles and the resulting crush [7, 23, 28, 29, 16, 15, 22, 10]. The model implemented in Pycrash is similar to the two-vehicle collision model described by Brach (2003) in which the impact related acceleration and velocity of the vehicles is determined based on a mutually applied force [7]. Pycrash provides a means to simulate Single Degree of Freedom (SDOF) collisions using a mutual stiffness value for the collision and, when provided a vehicle-specific stiffness, will calculate the crush sustained by each vehicle. Braking can also be applied by each vehicle. The mutual stiffness of the collision can be provided through a single stiffness value (k [ft/b]) [4, 15], the common A and B values [10, 20], or through a table of force-deflection data that may be derived from load-cell barrier crash test data [31, 32] or quasistatic testing of bumper components [4, 5, 16, 28, 29]. Pycrash has a module for processing the ascii formatted data files obtained by the National Highway Traffic Safety Administration (NHTSA) test site [1]. The Pycrash module will load the datafiles containing load cell and accelerometer data, integrate the accelerometer data and create tables of force-displacement by summing across the rows and columns chosen by the user (Figure 6).

Once a stiffness relationship is chosen (solid line in Figure 6), whether from load cell data, or other published sources, that data can be applied directly to a Pycrash vehicle to create a vehicle specific stiffness or it can be combined with additional data to create a mutual stiffness for the collision. Pycrash iteratively calculates the force and resulting acceleration of each vehicle using the provided stiffness data as a lookup table to determine the force produced at each iterated

FIGURE 6 Barrier load cell data from crash test and linear approximations used to simulate vehicle collision in Pycrash

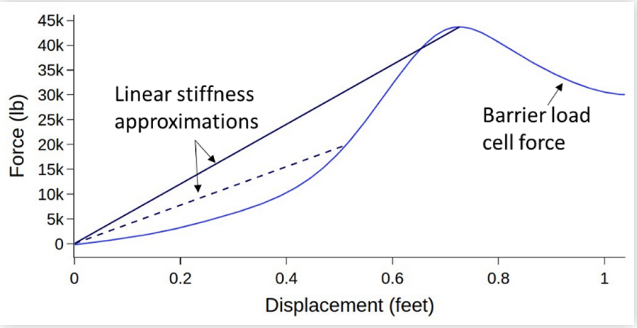


FIGURE 7 Example of a collision mutual force-displacement response derived from load cell barrier data and constant vehicle stiffness data

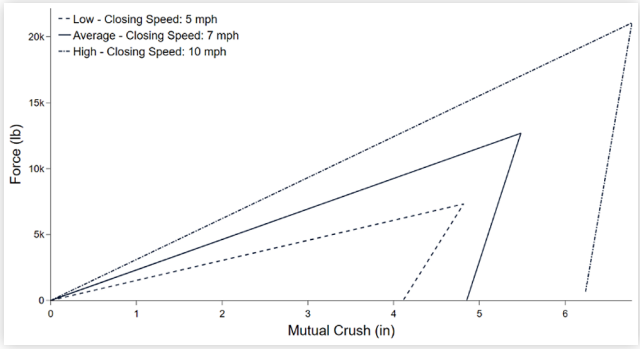
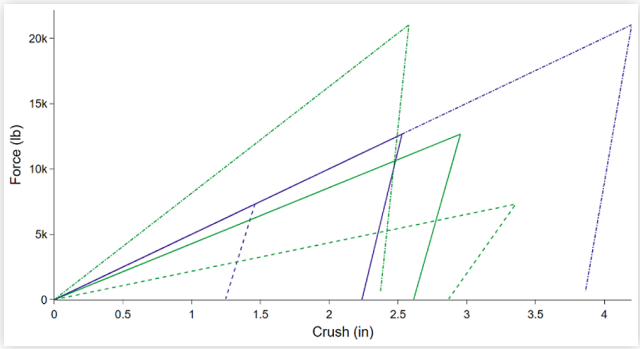


FIGURE 8 Example of vehicle-specific force-displacement response calculated using the SDOF model in Pycrash. Blue line represents a constant stiffness for each run with varying levels of residual crush. Green line represents variable stiffness for each model run resulting in similar levels of mutual crush



level of crush, producing a mutual force-deflection response (Figure 7). If vehicle-specific stiffness values are provided, then vehicle-specific crush will be calculated (Figure 8). Vehicle-specific crush is calculated using the vehicle-specific crash values provided to determine the vehicle crush required to produce the mutually applied force.

Pycrash is designed to run in iterations of three so that a low, mid and, high range model can be generated. The closing speeds can be chosen to produce the desired crush at each stiffness level and the resulting vehicle responses plotted (Figure 9, Figure 10).

Currently, this model is limited to a SDOF, future improvements will allow the forces to be applied at the point of impact as defined in the momentum analyses.

Impulse Momentum Vehicular impacts are also simulated within Pycrash using the impulse momentum planar collision (IMPC) method. IMPC is used to calculate the translational and rotational momentum transfer between two vehicles in a collision that is assumed to take place instantaneously. A two-vehicle impact can be simulated with the vehicles positioned at their point of impact with their pre-impact speeds and yaw rates defined (Figure 1), or with initial motion of each vehicle with the IMPC method applied when

FIGURE 9 Velocity predicted using the SDOF model of an impact described in Figure 7. Solid line represents average model response

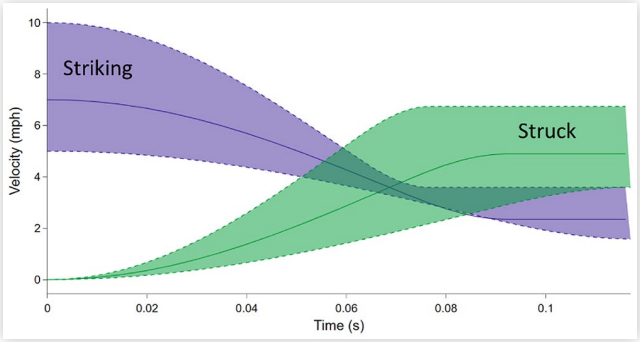
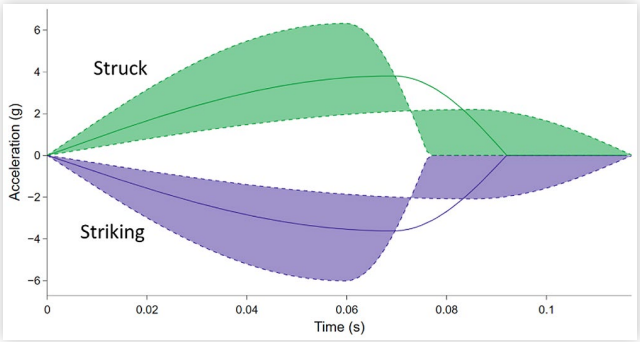


FIGURE 10 Acceleration predicted using the SDOF model of an impact described in Figure 7. Solid line represents average model response



impact is detected. The IMPC method in Pycrash uses the Carpenter and Welcher (2019) formulation that accounts for dissipated energy from sliding [12]. Similar to the Brach et al. (2007) [6] application, the Carpenter method requires the user to input the angle of the tangential contact plane, the coefficient of restitution, the available inter-vehicular Coulomb friction coefficient, and a point on each vehicle that defines the impulse center. In additional to the coefficient of restitution, an inter-vehicular Coulomb friction value is required which specifies the relationship between the normal and tangential impulse vectors. To illustrate Pycrash functionality and the relationship between the impulse ratio required by Brach et al. (2007) and the friction value in Carpenter and Welcher (2019), a series of four simulations were performed based on four frontal-oblique impacts [26]. These tests involved passenger vehicles and SUVs, both traveling at about 35 mph with a frontal impact angle of about 30 degrees. In the previously published research, the authors simulated the crash tests using the Brach impulse-momentum formation and determined the inputs that produced similar results to the crash test data. They determined that, the critical impulse ratio ranged between -0.51 and -0.4 in the four tests 100%. When simulating these four crash tests, we input the critical impulse ratio determined in the prior work. The results of these simulations were used to show the basic functionality of Pycrash and assess the relationship between the impulse ratio and friction inputs required by the two IMPC methods.

Results

All output data from Pycrash are easily accessible by the user to create custom plots or as inputs in additional analyses. The plots shown here are produced from built-in functions.

Vehicle Motion

Pycrash modeled the 15 and 30 mph steer tests very well, predicting the vehicle position within two and 4.5 feet respectively, of the circles created by the vehicle CG at the end of the 11 s PC-Crash simulations (Figure 11).

At the vehicle level, the forward and rightward acceleration calculated by Pycrash closely matched that of PC-Crash (Figure 12). The greatest difference in acceleration between the two models was 0.02 and 0.05 g for the forward and lateral acceleration respectively.

There was greater variation between the Pycrash and PC-Crash CG positions for the fishhook tests. In the 35 mph run, the variation in vehicle position was mostly caused by a difference in the vehicle heading following the maneuver. At this time (~6.5 s), the difference in heading angle was 15 degrees which resulted in an increase in the variation between the simulations with time (Appendix 2a). At the time the fishhook maneuver had been completed, the resultant difference in position was 14 feet; this difference grew linearly as the simulation continued in time. The Pycrash model produced similar vehicle acceleration to the PC-Crash simulation over the entire duration of the maneuver (Figure 13).

The lateral tire forces produced by the Pycrash model for the 35 mph fishhook maneuver explain the variation seen in the acceleration response (Figure 14). The Pycrash simulation

FIGURE 11 Pycrash simulation of the 30 mph steer test (top) and comparison between Pycrash (black) and PC-Crash CG locations (below)

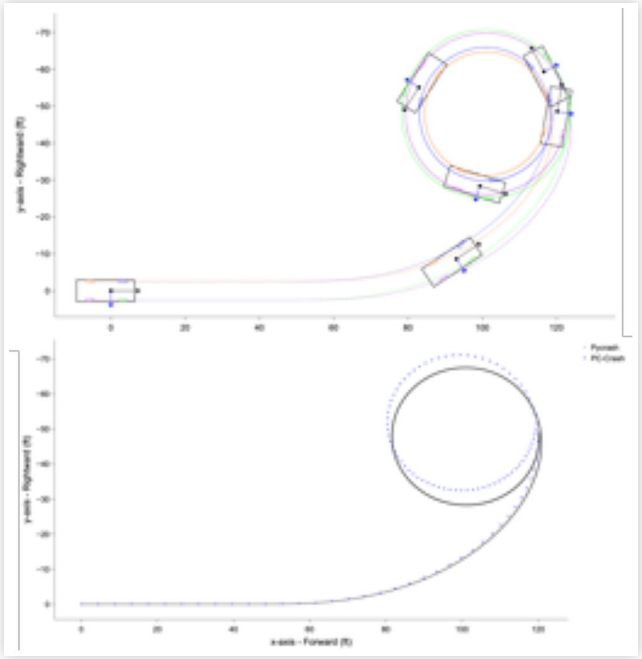


FIGURE 12 Forward (top) and rightward vehicle acceleration for the Pycrash and PC-Crash simulations of the 30 mph steer test (CG=1 ft)

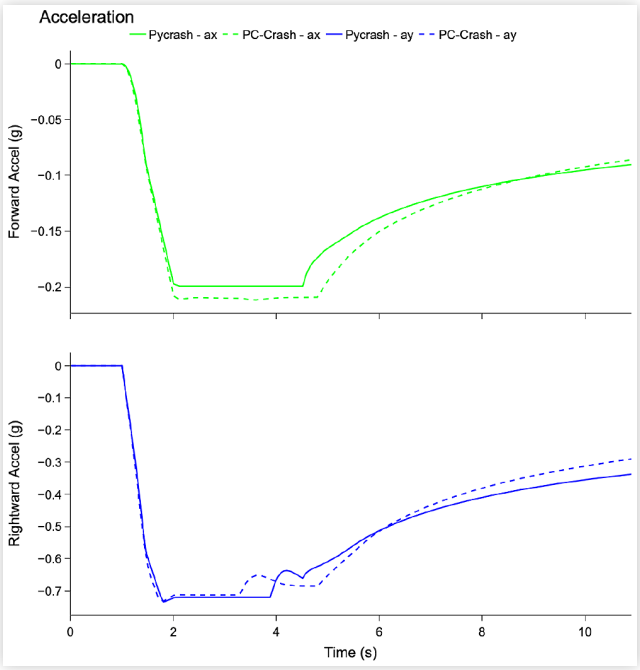
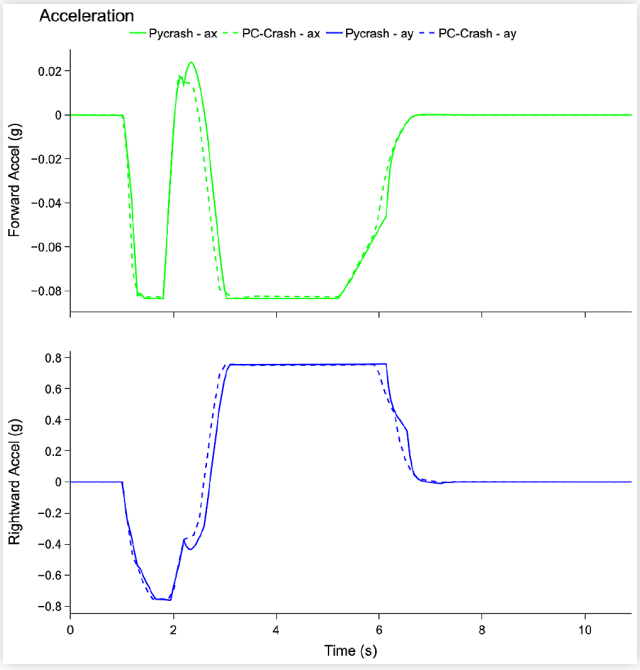


FIGURE 13 Forward (top) and lateral acceleration for the Pycrash and PC-Crash simulations of the 35 mph fishhook maneuver (CG=1 ft)



initially produced an over-estimate of the left front tire and an underestimate for the right front tire lateral forces. During the hold portion of the maneuver, a continual offset between the lateral tire forces is seen with the front tires having the greatest variation. On average, the lateral forces predicted

FIGURE 14 Lateral tire forces predicted by the Pycrash (CG=1 ft) and PC-Crash simulations for the 35 mph fishhook maneuver

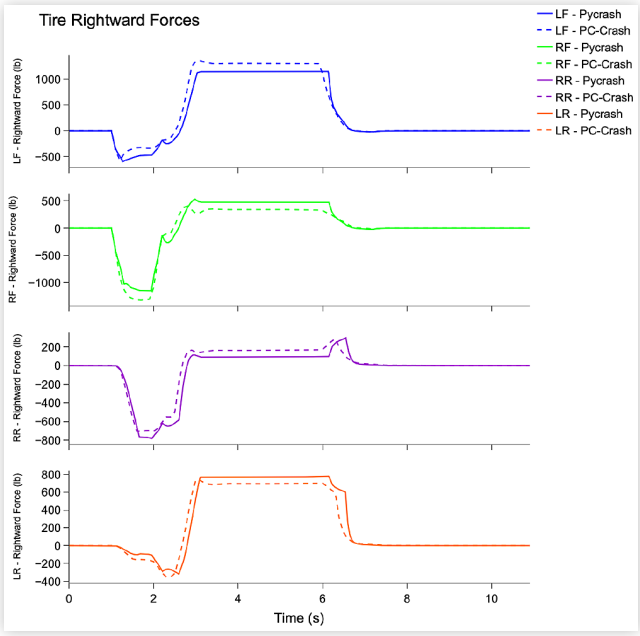
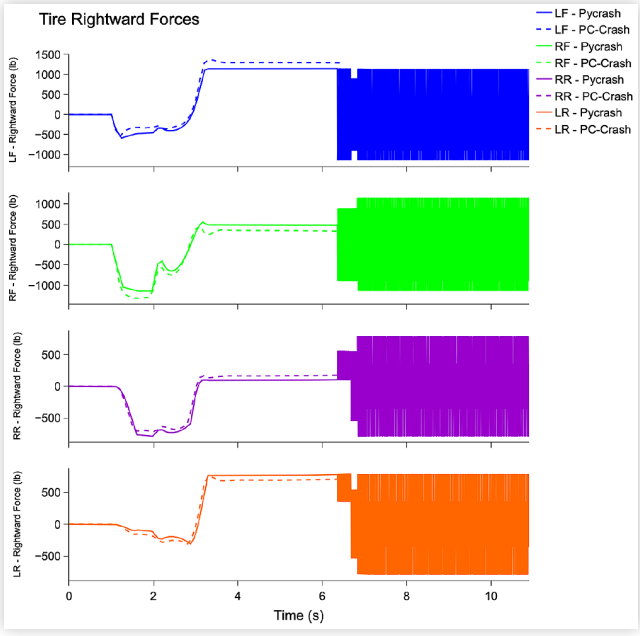


FIGURE 15 Lateral tire forces predicted by the Pycrash (CG=1 ft) and PC-Crash simulations for the 45 mph fishhook maneuver. Solid bars result when the model oscillates at zero



between the two models differ by about 2.6% at the front axle. This trend in tire forces was observed for the 40 mph fishhook maneuver simulation as well.

Similar to the 35 mph test, the 40 mph fishhook simulation resulted in a variation of the heading angles at the end of the maneuver, causing the vehicle CG positions to diverge as the vehicle continued to travel. The Pycrash simulation at 40

mph resulted in additional yaw over the PC-Crash model, causing the Pycrash model to lose more speed and travel a shorter distance after the fishhook maneuver was completed (Appendix 2b).

The PC-Crash vehicle spun-out during the 45 mph fishhook maneuver which was also predicted by the Pycrash model (Appendix 2d). While the PC-Crash vehicle was essentially stationary following the maneuver, the Pycrash model had a rearward speed of 9.5 mph, which increased the relative distance between the two vehicle simulations (Appendix 2c). To evaluate the effects of the Pycrash CG height, the fishhook maneuvers were also simulated with the Pycrash CG at 1 foot. The Pycrash vehicle with the lower CG height showed improved performance for the three conditions. The tire forces predicted by the Pycrash model in the 45 mph fishhook test showed the good agreement with the PC-Crash model (Figure 14).

Impulse Momentum

A series of four frontal-oblique crash tests were simulated using the Carpenter and Welcher impulse momentum formation within Pycrash. The results were compared to those published in a prior study using the Brach method by applying the same inputs and using the impulse ratio in the Brach method as the coefficient of friction in Pycrash (Table 1). In each test case, the Carpenter and Welcher method used within Pycrash indicated that sliding did occur and produced similar values for tangential energy loss, compared to the values produced by the Brach method. Overall, the tangential energy loss differed by 0.5% between the two methods. The change in speed and angular rate predicted by both models were essentially identical.

TABLE 1 Impulse momentum results for the Brach and Carpenter methods

		Totals delta-V (mph)		delta-Omega (deg/s)		
Test	IMPC Method	Veh 1	Veh 2	Veh 1	Veh 2	Tangential E-loss (ft-lb)
4363	Brach (Rose 2006)	39.5	29.1	31.3	-182	88158
	Carpenter (Pycrash)	39.5	29.1	31	-182.6	87835
4364	Brach (Rose 2006)	40.1	26.6	-110	-148	72434
	Carpenter (Pycrash)	40.1	26.6	-108	-150.5	71187
4438	Brach (Rose 2006)	39.7	27.8	-68	-166	69145
	Carpenter (Pycrash)	39.7	27.8	-67.45	-166	69425
4474	Brach (Rose 2006)	40.5	28.7	40	-194	65803
	Carpenter (Pycrash)	40.4	28.7	40.1	-194	65582

Discussion

The calculations and analyses performed by Pycrash are based on fundamental physics and reconstruction methods found in the published literature [10, 11, 12, 13, 16, 17, 27, 30, 31]. The impulse momentum calculations using the IMPC equations of Carpenter and Welcher (2019) were compared to those obtained using the planar impact mechanics equations of Brach [6]. When choosing the same value for the inter-vehicular friction as the impulse ratio, the Pycrash impulse momentum model essentially predicted the same values for delta-V and change in angular rate as the prior research using the Brach impulse momentum formulation. This was an expected result since the both the impulse ratio and the coefficient of friction values determine the relationship between the normal and tangential momentum along the defined impact plane.

The results of the vehicle motion simulations showed that the simplified suspension model works well for basic steering inputs and even more complex steering inputs at the speeds evaluated here. For the simulations not involving spinout, the average response between the tires in the Pycrash vehicle closely matched the overall response predicted by PC-Crash while producing balancing variation at the tire level. The fishhook simulation at 45 mph showed that Pycrash may not provide accurate vehicle displacement prediction for more transient maneuvers at higher speeds, when complex steering inputs are needed. The source of the variation for the Pycrash vehicle is the simplified suspension model that does not account for variability in suspension from front to rear or left to right that can occur during high-speed steering maneuvers. The weight shift in Pycrash occurs instantaneously with longitudinal and lateral acceleration which produces the variations seen in the higher-speed maneuver responses evaluated here. While the rest location predicted between the two vehicles showed variation, this was due in part to the run-out produced after the maneuvers. The acceleration predicted by the Pycrash model closely matched that of the PC-Crash simulations.

When applying Pycrash to events that are beyond the scope of this paper, engineering judgement should be made to determine the applicability of the Pycrash model and its ability to create the expected vehicle motion. To facilitate this effort, the validation data provided along with Pycrash can be used to assess the accuracy of the simulations for a given application (see contact information). Since Pycrash is open source, additional validation data can be applied and used to assess and improve its accuracy in the future.

The SDOF impact simulation provides a flexible means of simulating impacts using standard A and B coefficients, linear spring stiffnesses or tabular force-displacement data. The output of these models can be used to assess the dissipated energy of a collision based on mutual or individual vehicle stiffnesses and crush. Pycrash can be used to iteratively simulate a series of collisions to determine the closing speed associated with a defined level of crush for the subject vehicles.

The model incorporates restitution so that absorbed and dissipated energy can be calculated to account for varying levels of restitution.

Pycrash is an open-source software and can be installed onto any computer using the Python Package Index (pypi.org). The main code can also be downloaded so users can add functionality specific to their use case or make improvements to the main code. Currently, Github (github.com) is used to host Pycrash which provides version control for all files and scripts associated with Pycrash. Changes to the main code can be tracked which allows contributors to make improvements to the main code once approved and any alterations by users downloading the code can be identified.

Conclusions

The current functionality of Pycrash provides a useful tool for fundamental reconstruction calculations, impact response using impulse-momentum methods and basic vehicle motion. Accident reconstructionists do not need to know Python in order to use or improve Pycrash. The current functionality of Pycrash provides a useful tool for fundamental reconstruction calculations, impact response using impulse-momentum methods and basic vehicle motion. As it is an open-source package, it is hoped that reconstructionists will use it and make or suggest improvements.

References

1. National Highway Traffic Safety Administration, "Vehicle Crash Test Database," 2020, https://www-nrd.nhtsa.dot.gov/database/veh/veh.htm.
2. National Highway Traffic Safety Administration, US Department of Transportation, "Laboratory Test Procedure for Dynamic Rollover," The Fishhook Maneuver Test Procedure, New Car Assessment Program, 2013.
3. Beauchamp, G., Pentecost, D., Koch, D., and Bortles, W., "Speed Analysis of Yawing Passenger Vehicles Following a Tire Tread Detachment," *SAE Int. J. Adv. & Curr. Prac. in Mobility* 1(3):883-917, 2019, https://doi.org/10.4271/2019-01-0418.
4. Bonugli, E., Watson, R., Freund, M., and Wirth, J., "Expanded Characterization of Force-Deflection Properties of Vehicle-to-Vehicle Systems," *Society of Automotive Engineers*, 2017, https://doi.org/10.4271/2017-01-1417.
5. Bonugli, E., Wirth, J., Funk, J., Cormier, J. et al., "Characterization of Force Deflection Properties for Vehicle Bumper-to-Bumper Interactions," SAE Int. J. Trans. Safety 2014-01-1991, 2014, https://doi.org/10.4271/2014-01-1991.
6. Brach, R., Welsh, K., and Brach, R., "Residual Crush Energy Partitioning Normal and Tangential Energy Losses," *Society of Automotive Engineers*, 2007, https://doi.org/10.4271/2007-01-0737.
7. Brach, R.M., "Modeling of Low-Speed, Front-to-Rear Vehicle Impacts," *Society of Automotive Engineers*, 2003, https://doi.org/10.4271/2003-01-0491.
8. Brach, R.M. and Brach, R.M., "Tire Models for Vehicle Dynamic Simulation and Accident Reconstruction," *Society of Automotive Engineers*, 2009, https://doi.org/10.4271/2009-01-0102.
9. Burhaumudin, M.S., Samin, P.M., Jamaluddin, H., Rahman, R.A., and Sulaiman, S., "Modeling and Validation of Magic Formula Tire Model," in *Int Conf Auto, Mech, Mat Eng* May, 113-18, 2012.
10. Campbell, K.L., "Energy Basis for Collision Severity," *Society of Automotive Engineers*, 1974, https://doi.org/10.4271/740565.
11. Carpenter, N. and Welcher, J., "Stiffness and Crush Energy Analysis for Vehicle Collision and its Relationship to Barrier Equivalent Velocity (BEV)," *Society of Automotive Engineers*, 2001, https://doi.org/10.4271/2001-01-0500.
12. Carpenter, N. and Welcher, J., "Inter-Vehicular Sliding Friction and Crush Energy Losses in Impulse Momentum Planar Collision," *Society of Automotive Engineers*, 2019, https://doi.org/10.4271/2019-01-0422.
13. Cipriani, A., Bayan, F., Woodhouse, M., Cornetto, A. et al., "Low Speed Collinear Impact Severity: A Comparison Between Full Scale Testing and Analytical Prediction Tools with Restitution Analysis," Society of Automotive Engineers 2002-01-0540, 2002, https://doi.org/10.4271/2002-01-0540.
14. Day, T., ROberts, S., and York, A., "SIMON: A New Vehicle Simulation Model for Vehicle Design and Safety Research," *Society of Automotive Engineers*, 2001, https://doi.org/10.4271/2001-01-0503.
15. Deyerl, E., Cheng, L., and Gatti, J., "Two-Dimensional Collision Simulations of Low-Speed Crash Tests," *Society of Automotive Engineers*, 2013, https://doi.org/10.4271/2013-01-0793.
16. Funk, J., Bonugli, E., Guzman, H., and Freund, M., "Comparison of Quasistatic Bumper Testing and Dynamic Full Vehicle Testing for Reconstructing Low Speed Collisions," *SAE Int. J. Passeng. Cars - Mech. Syst* 7(3), 2014, https://doi.org/10.4271/2014-01-0481.
17. Gillespie, T.D., "Fundamentals of Vehicle Dynamics," *SAE International*, 1992, https://doi.org/10.4271/r-114.
18. Harris, C.R., Millman, K.J., van der Walt, S.J., Gommers, R. et al., "Array Programming with NumPy," *Nature* 585(7825):357-362, 2020, doi:10.1038/s41586-020-2649-2.
19. Kluyver, T., Ragan-Kelley, B., Perez, F., Granger, B., Bussonnier, M., Frederic, J., Kelley, K., Hamrick, J., Grout, J., Corlay, S., Ivanov, P., Avila, D., Abdalla, S., and Willing, C., "Jupyter Notebooks - A Publishing Format for Reproducible Computational Workflows," in *20th International Conference on Electronic Publishing*, 2016 .
20. Lee, E.L., Lee, P.J., Erickson, M.S., and Hayes, W.C., "Increase in Vehicle Front, Rear and Side Stiffness Coefficients in the Past Twenty Years Necessitates New Representative Database," SAE Technical Paper 2014-01-0351, 2014, https://doi.org/10.4271/2014-01-0351.

21. McKinney, W. "Data Structures for Statistical Computing in Python," in *Proc. 9th Python in Science Conf*, 2010.
22. Neptune, J.A. and Flynn, J.E., "A Method for Determining Accident Specific Crush Stiffness Coefficients," *Society of Automotive Engineers*, 1994, https://doi.org/10.4271/940913.
23. Nishimura, N., Simms, C.K., and Wood, D.P., "Impact Characteristics of a Vehicle Population in Low Speed Front to Rear Collisions," *Accid Anal Prev* 79:1-12, 2015, doi:10.1016/j.aap.2015.02.001.
24. Radionova, L.V. and Chernyshev, A.D., "Mathematical Model of the Vehicle in MATLAB Simulink," *Procedia Engineering*, 2015.
25. Rose, A. and Carter, N., "An Analytical Review and Extension of Two Decades of Research Related to PC-Crash Simulation Software," *Society of Automotive Engineers*, 2018, https://doi.org/10.4271/2018-01-0523.
26. Rose, N.A., Fenton, S.J., and Beauchamp, G., "Restitution Modeling for Crush Analysis: Theory and Validation," *Society of Automotive Engineers*, 2006, https://doi.org/10.4271/2006-01-0908.
27. Rose, N.A., Fenton, S.J., and Ziernicki, R.M., "Crush and Conservation of Energy Analysis: Toward a Consistent Methodology," *Society of Automotive Engineering*, 2005, https://doi.org/10.4271/2005-01-1200.
28. Scott, M.W., Bain, C., Manoogian, S., Cormier, J., and Funk, J., "Simulation Model for Low-Speed Bumper-to-Bumper Crashes," *Society of Automotive Engineers*, 2010, https://doi.org/10.4271/2010-01-0051.
29. Scott, W., Bonugli, E., Guzman, H.M., and Swartzendruber, D., "Reconstruction of Low-Speed Crashes using the Quasi-Static Force vs. Deformation Characteristtics of the Bumpers Involved in the Crashes," *SAE Int. J. Passeng. Cars - Mech. Syst.* 5(1), 2012, https://doi.org/10.4271/2012-01-0598.
30. Steffan, H. and Moser, A., "The Collision and Trajectory Models of PC-Crash," *Society of Automotive Engineers SAE No. 960886*, 1996, https://doi.org/10.4271/960886.
31. Struble, D.E., *Automotive Accident Reconstruction Practices and Principles, Ground Vehicle Engineering Series* (Boca Raton (FL): CRC Press, 2014).
32. Struble, D.E., Welsh, K.J., and Struble, J.D., "Crush Energy Assessment in Frontal Underride/Override Crashes," *Society of Automotive Engineers*, 2009, https://doi.org/10.4271/2009-01-0105.
33. Plotly Technologies, *Collaborative Data Science* (Montréal, QC: Plotly Technologies Inc., 2015). https://plot.ly.
34. Turnip, A. and Fakhrurroja, H., "Estimattion of the Wheel-Ground Contact Tire Forces usign Extended Kalman Filter," *International J of Instrumentation Science* 2(2):34-40, 2013.
35. Virtanen, P., Gommers, R., Oliphant, T.E., Haberland, M. et al., "SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python," *Nat Methods* 17(3):261-272, 2020, doi:10.1038/s41592-019-0686-2.
36. Zhou, J., Peng, H., and Lu, J.. 2008, "Collision Model for Vehicle Motion Prediction After Light Impacts," *Vehicle System Dynamics* 46 Supp:3-15.

Contact Information

The source code for Pycrash as well as the code used to create the simulations in this paper and additional demonstrative Jupyter notebooks are located on Github (https://github.com/joe-cormier/pycrash). Pycrash can also be installed using pip from PyPi (https://pypi.org/). Collaborators and additional validation data are welcome, please contact Joe Cormier (jcormier@biocorellc.com) if interested.

Appendix 1: Example Code and Usage with Jupyter Notebook

APPENDIX 1a Segment of Jupyter Notebook used to create vehicle and simulation of the 30 mph steer test.

Vehicle inputs are created manually as shown, or through automated prompts when executing various tasks in Pycrash

Create Vehicle Input

`vehicle_input_dict` is a Python dictionary containing necessary vehicle inputs

```
# PC Crash vehicle specifications
vehicle_input_dict = {"year":2004,
"make":"Chevrolet",
"model":"Malibu",
"weight":3298,
"brake":0,
"steer_ratio":15.9,
"init_x_pos":0,
"init_y_pos":0,
"head_angle":0,
"width":70 / 12,
"length":187 / 12,
"hcg":21.5 / 12,
"lcgf":38.1 / 12,
"lcgr":67.9 / 12,
"wb":106 / 12,
"track":60 / 12,
"f_hang":38 / 12,
"r_hang":43 / 12,
"tire_d":26.2 / 12,
"tire_w":8.5 / 12,
"izz":2040,
"fwd":1}
```

"Vehicle" creates a Pycrash vehicle based on inputs within *vehicle_input_dict* Vehicle properties (*vx_initial, hcg*) are set

```
veh1 = Vehicle('Veh1', vehicle_input_dict)
veh1.time_inputs(t, throttle, brake, steer)
veh1.vx_initial = 45
veh1.hcg = 21.5 / 12  # <- set CG height
```

```
Vehicle inputs for Veh1 applied succesfully
Driver inputs applied to Veh1
```

Jupyter Notebook interface contains markdown text between code

Create Single Motion Instance

- simulation is run when instance is created which takes a few seconds
- model is generating data at each time step as well as all position data to draw vehicle position

"SingleMotion" creates a Pycrash simulation of a single vehicle motion using the vehicle created above

```
simulation_name = '30 mph steer'
print(f'Creating Simulation: {simulation_name}')
run = SingleMotion(simulation_name, veh1)
```

Settings / simulation status provided when code above is executed

```
Creating Simulation: 30 mph steer
Maximum allowable friction: 0.76
Time step for vehicle motion (s) : 0.01
Maximum tire slip angle (deg): 10.00
Driver input for Veh1 of shape = (1091, 4)
Vehicle motion will be simulated for 10.9 seconds
```

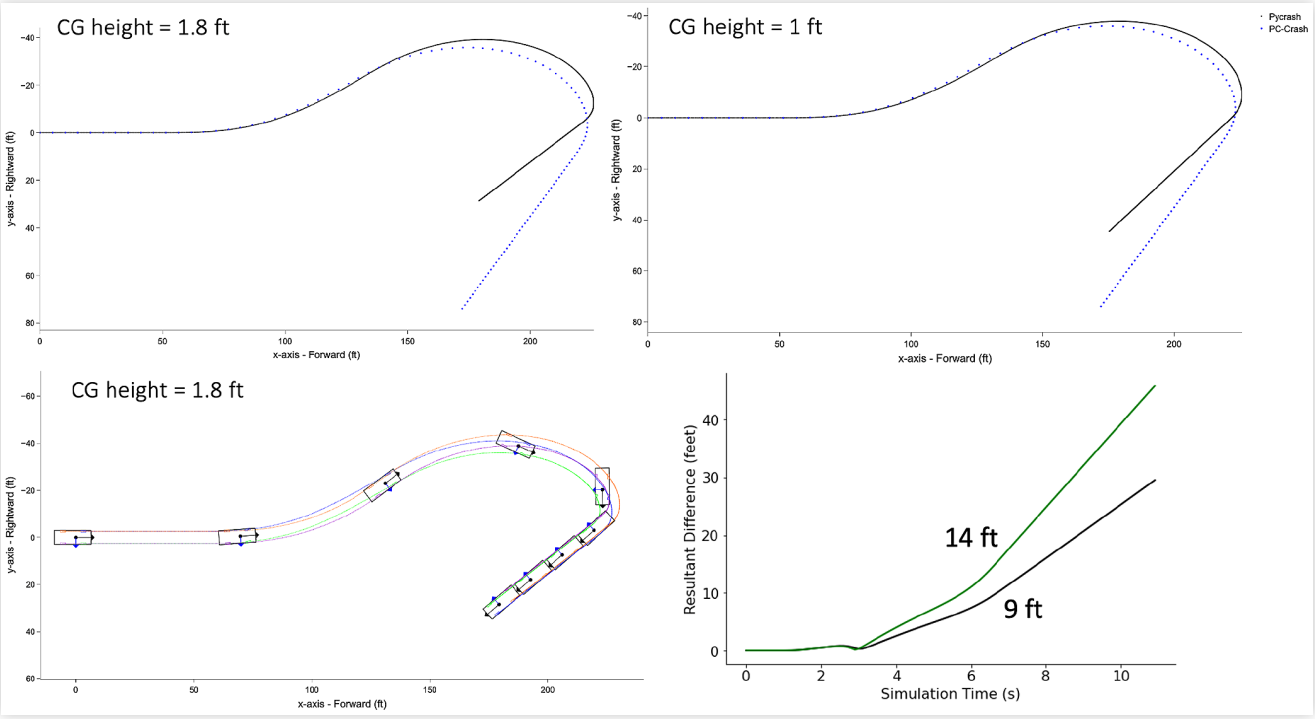
Execute function for plotting motion

```
plot_motion_interval(run.veh, num_itter = 15)
```

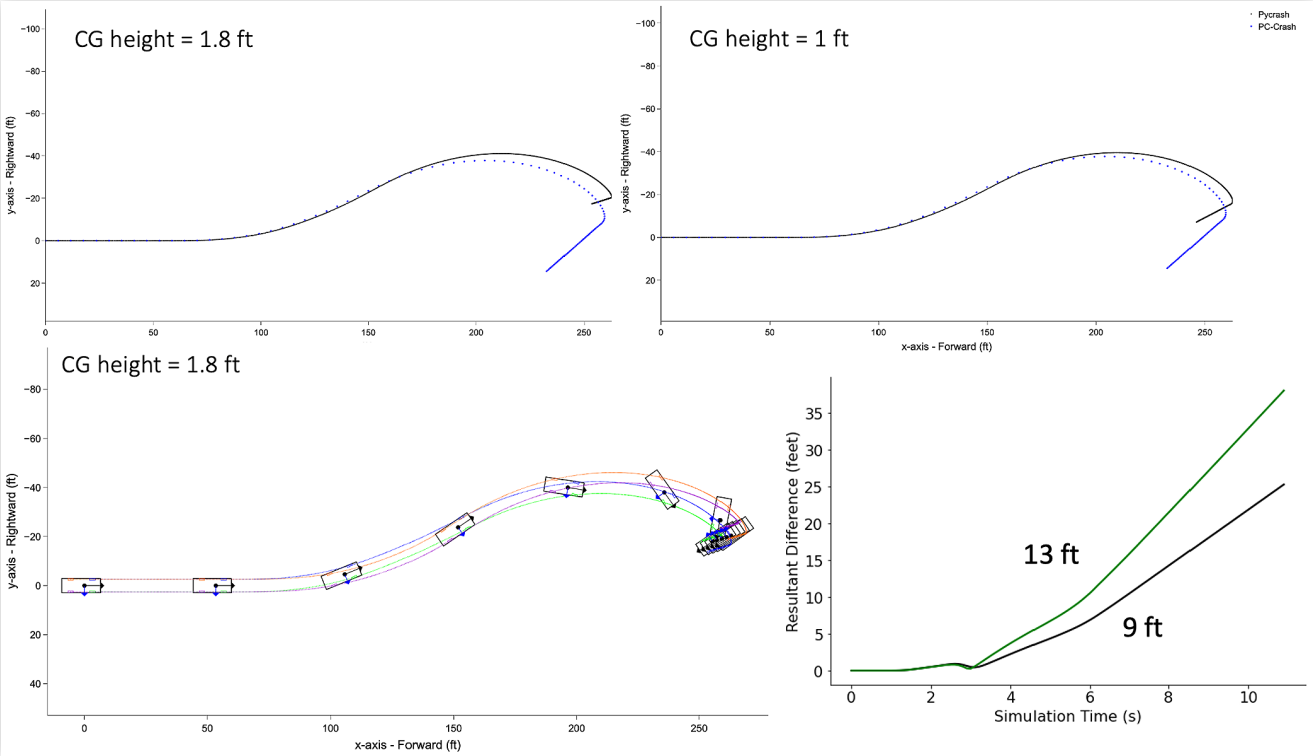
Plot is generated in browser window

Appendix 2: Pycrash and PC-Crash Fishhook Simulation Results

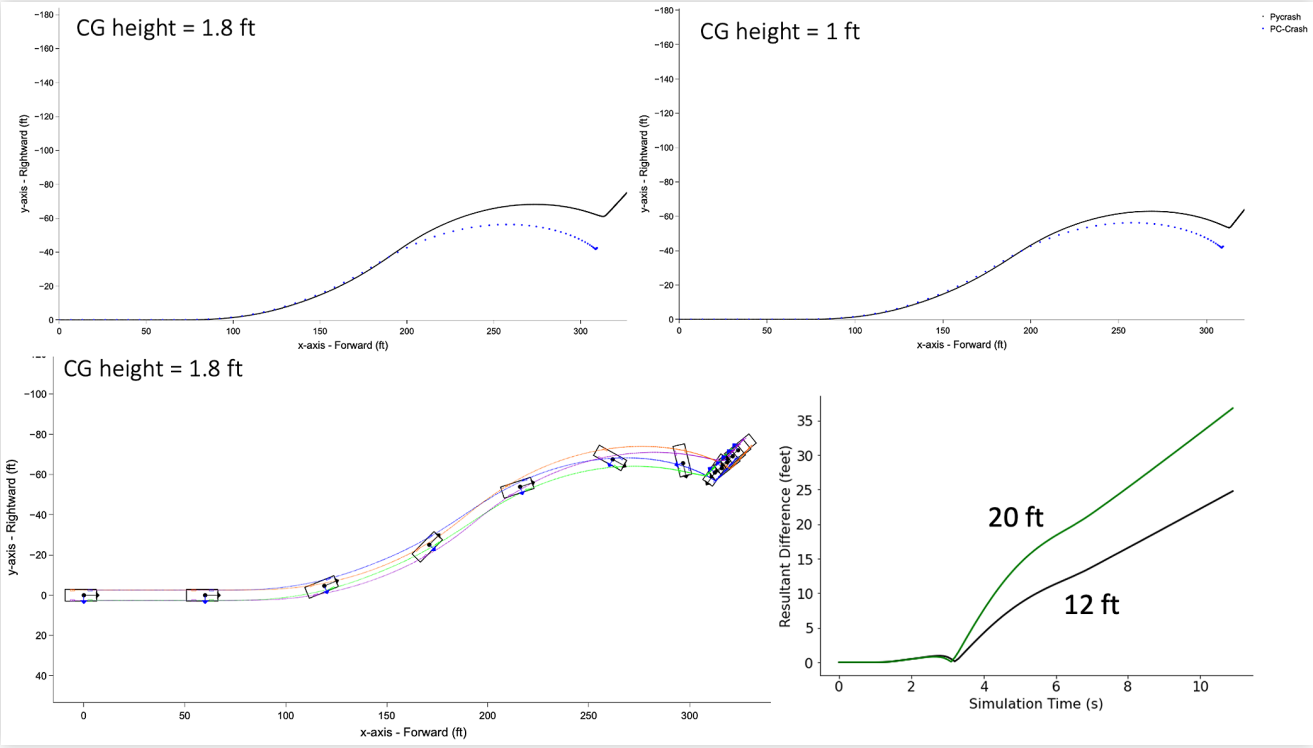
APPENDIX 2a 35 mph fishhook results at a CG height of 1.8 and 1 foot for the Pycrash vehicle showing relative position between Pycrash and PC-Crash vehicles. Lower right indicates resultant difference between CG location for the Pycrash and PC-Crash models. Green line is the model with a 1.8 ft CG height, labels indicate distance at 6.5 seconds



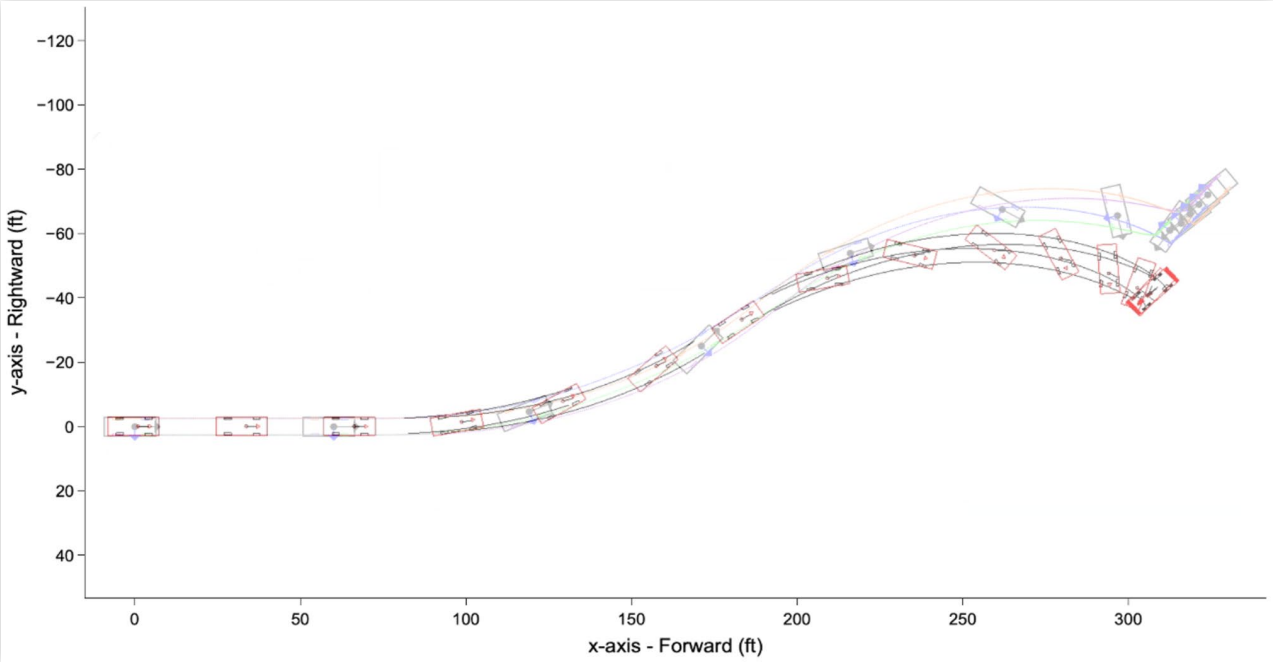
APPENDIX 2b 40 mph fishhook results at a CG height of 1.8 and 1 foot for the Pycrash vehicle showing relative position between Pycrash and PC-Crash vehicles. Lower right indicates resultant difference between CG location for the Pycrash and PC-Crash models. Green line is the model with a 1.8 ft CG height, labels indicate distance at 6.5 seconds



APPENDIX 2c 45 mph fishhook results at a CG height of 1.8 and 1 foot for the Pycrash vehicle showing relative position between Pycrash and PC-Crash vehicles. Lower right indicates resultant difference between CG location for the Pycrash and PC-Crash models. Green line is the model with a 1.8 ft CG height, labels indicate distance at 6.5 seconds



APPENDIX 2d PC-Crash vehicle motion during the 45 mph fishhook simulation overlayed onto the Pycrash model



ISSN 0148-7191