Author: Tim Baron
6/19/2017

Dynamical Modeling
Computer Controlled Stopping of a Car

Short Abstract:

There is currently a lot of research being done on self-driving cars. This new technology presents many challenges. This project focuses on one very important and common task that self-driving cars will be doing. They will be stopping at stop signs. This project uses kinematic equations and dynamical mathematical modeling to analyze how a computer may look at the process of stopping at a stop sign given position and velocity. The results indicate that the task is possible even with noise and bias included in the model. However, it also indicates that smooth stopping may be difficult to achieve using this model.

Medium Abstract:

Academic researchers, automobile manufacturers, and technology companies are investing time and resources into self-driving cars. They are likely to be on the road in the near future. Although there are several complex tasks that a self-driving car must be able to do, one of the most fundamental tasks is to be able to stop. The car must not only stop, but it must stop in a safe way that keeps the passengers comfortable. Essentially, it must imitate the way a human driver stops.

This project will use dynamical modeling to model how a computer-controlled car may look at the process of stopping at a stop sign. To model this, the car's computer would need to know the velocity and mass of the car as well as the distance to the stop sign. Using a dynamical model, this project will analyze a car at an initial velocity that must stop a given distance away. The model will adjust the target acceleration based on the changes in velocity and distance. Noise and bias will be introduced to inject possible unknown quantities into the model and calculate a simulated actual acceleration. The model will also have the option of entering an acceleration bias for situation such as a sloped road. This model will allow for an analysis of changing conditions as the car stops.

The project results show that a computer with the proper data should be able to stop at a stop sign if programmed to correctly analyze the conditions, but there may be difficulty in achieving a smooth stop using this model.

Introduction:

Self-driving cars using onboard computers are already in development. It's likely we will see self-driving cars in the near future. Although there will be a plethora of variables that a real world driving self-driving car will need to manipulate to function, this project will look at process of stopping at a stop sign.

In order for an onboard computer to allow for a car to make a safe stop it must be able to gather and analyze the correct data. In a real world situation this would include velocity, distance, acceleration, obstacle detection, road conditions, the ability to adapt to changing conditions, and slope. For this model, we will concentrate on position, velocity, target velocity, and actual velocity.

Model:

We will use a dynamical mathematical model to analyze a computer-controlled car stopping at a stop sign. We will start with basic, real-world data for the distance, velocity, and mass of a car. We will use the kinematic equations to make our calculations. We will use randomly generated noise to inject random variables into the model. We will also include the option of adding acceleration bias in the event we want to add slope to the model.

For this model we will use the following data:

velocity (initial)	18	m/s
velocity (final)	0	m/s
position (initial)	46	m
position (final)	0	m
mass (car)	1600	kg
time increment	0.1	s

The following kinematic physics equations will allow us to determine the changes in distance, velocity, and acceleration:

The Kinematic Equations

$$d = v_i * t + \frac{1}{2} * a * t^2 \qquad v_f^2 = v_i^2 + 2 * a * d$$

$$v_f = v_i + a * t \qquad d = \frac{v_i + v_f}{2} * t$$

To calculate the target acceleration we will use the equation:

$$a_t = \frac{v_f^{\,2} - v_i^{\,2}}{2d}$$

Note:

$a_t =$ target acceleration

$a_a =$ actual acceleration

To calculate distance we will use the equation:

$$d_f = d_i - v(\Delta t) - \frac{a_{actual}(\Delta t)^2}{2}$$

To calculate velocity we will use the equation:

$$v_f = v_i + a_a(\Delta t)$$

To add noise we will add a randomly generated noise column in Excel using the formula:

=normsinv(rand())

This will add a normally distributed random variable with mean 0 and standard deviation = 1.0 that will be used to add randomness to the actual acceleration. This represents variables outside of the control of a driver such as gritty brakes or small changes in road surface.

To calculate the actual acceleration we will use the formula:

$a_a =$ targetaccel*(1+Noiseaize*normsinv(rand())) + accelbias

For example, here are some calculated actual accelerations:

accelbias =	0

Noisesize =	0.05

target acceleration (m/s^2)	normsinv(rand())	accelbias	actual accleration (m/s^2)
-3.52	0.17	0	-3.55
-3.66	1.68	0	-3.97
-3.67	0.97	0	-3.84

The model will start by calculating the target acceleration from the initial conditions of the car at time zero. Then, using the actual acceleration formula (which adds noise and bias) we will calculate the actual acceleration at the time of the initial conditions. We will use the actual acceleration in combination with the kinematic equations to calculate the distance and velocity 0.1 seconds later. At this point we will recalculate the target acceleration using the kinematic equations and the new distance and velocity. We will then once again calculate the actual acceleration and use that to calculate the distance and velocity 0.1 seconds later using the kinematic equations. This calculation pattern will continue for the rest of the model until we reach a distance of zero, which indicates the car has stopped. Pressing F9 on the keyboard will refresh the model with new random noise, which will change all of the calculations. From this, we can look at various stopping scenarios.

Results:

Initial Conditions:

velocity (initial)	18	m/s
velocity (final)	0	m/s
position (initial)	46	m
position (final)	0	m
mass (car)	1600	kg
time increment	0.1	s

acceleration (target)	-3.522	m/s^2

accelbias =	0

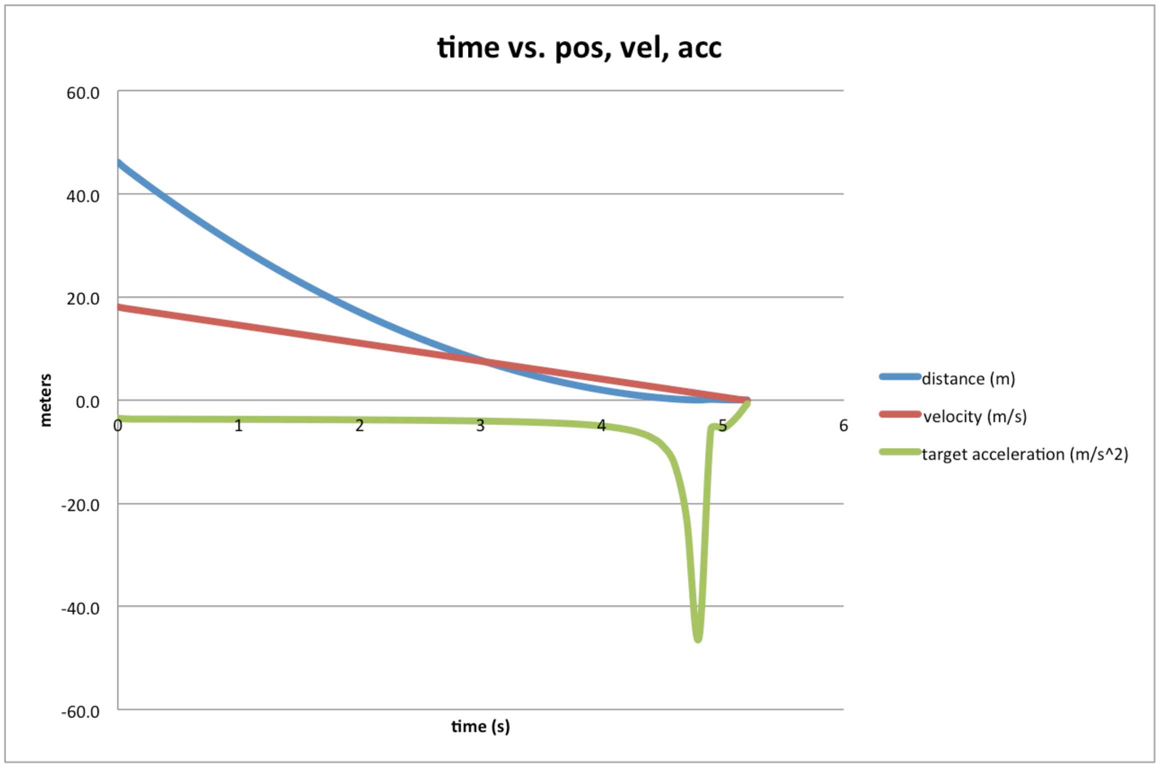
Noisesize =	0.05

An example of one randomly generated table of results:

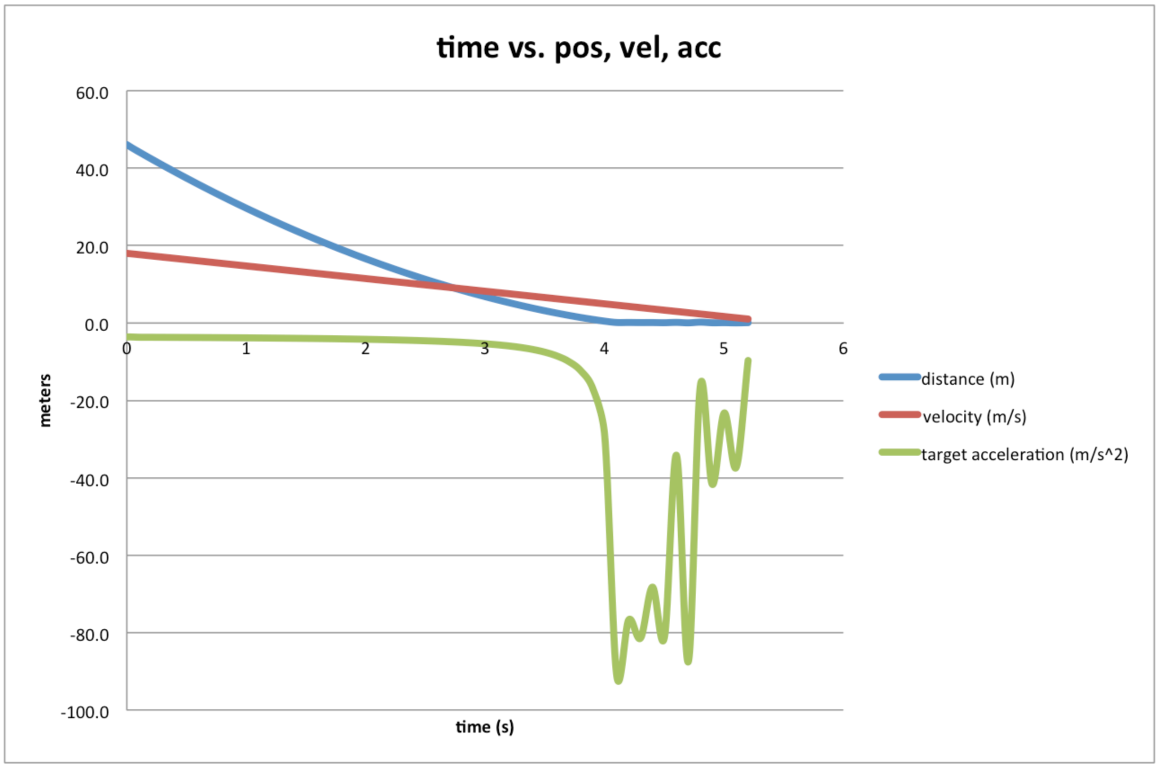
time (s)	distance (m)	distance (ft)	velocity (m/s)	velocity (mph)	KE (joules)	target acceleration (m/s^2)	normsinv(rand())	accelbias	actual accleration (m/s^2)
0	46.0	150.9	18.0	40.3	259200	-3.52	-0.39	0	-3.45
0.1	44.2	145.0	17.7	39.5	249351	-3.66	-0.35	0	-3.60
0.2	42.5	139.3	17.3	38.7	239692	-3.67	0.54	0	-3.77
0.3	40.8	133.7	17.0	37.9	230224	-3.68	-1.22	0	-3.45
0.4	39.1	128.2	16.6	37.2	220947	-3.68	2.07	0	-4.06
0.5	37.4	122.8	16.3	36.4	211861	-3.69	-0.40	0	-3.61
0.6	35.8	117.5	15.9	35.6	202965	-3.70	-0.27	0	-3.65
0.7	34.3	112.4	15.6	34.9	194261	-3.70	1.08	0	-3.90
0.8	32.7	107.3	15.2	34.1	185747	-3.71	-0.99	0	-3.53
0.9	31.2	102.4	14.9	33.3	177423	-3.72	0.75	0	-3.86
1	29.7	97.5	14.5	32.5	169291	-3.73	-2.66	0	-3.23
1.1	28.3	92.8	14.2	31.8	161349	-3.74	-1.03	0	-3.55
1.2	26.9	88.2	13.9	31.0	153599	-3.75	-0.85	0	-3.59
1.3	25.5	83.7	13.5	30.2	146038	-3.76	-1.28	0	-3.52
1.4	24.2	79.4	13.2	29.5	138669	-3.77	2.10	0	-4.17
1.5	22.9	75.1	12.8	28.7	131491	-3.78	1.23	0	-4.02
1.6	21.6	71.0	12.5	27.9	124503	-3.80	-0.51	0	-3.70
1.7	20.4	66.9	12.1	27.1	117706	-3.81	1.41	0	-4.08
1.8	19.2	63.0	11.8	26.4	111100	-3.83	-0.37	0	-3.76
1.9	18.1	59.2	11.4	25.6	104684	-3.85	0.74	0	-3.99
2	16.9	55.5	11.1	24.8	98460	-3.86	0.44	0	-3.95
2.1	15.8	52.0	10.7	24.0	92426	-3.88	0.21	0	-3.92
2.2	14.8	48.5	10.4	23.3	86583	-3.91	-1.16	0	-3.68
2.3	13.8	45.2	10.1	22.5	80931	-3.93	2.10	0	-4.34
2.4	12.8	41.9	9.7	21.7	75469	-3.96	-2.17	0	-3.53
2.5	11.8	38.8	9.4	21.0	70198	-3.99	-2.22	0	-3.55
2.6	10.9	35.8	9.0	20.2	65118	-4.02	0.27	0	-4.08
2.7	10.0	32.9	8.7	19.4	60229	-4.06	1.03	0	-4.27
2.8	9.2	30.1	8.3	18.6	55531	-4.10	0.32	0	-4.17
2.9	8.4	27.5	8.0	17.9	51023	-4.15	0.06	0	-4.16
3	7.6	24.9	7.6	17.1	46706	-4.20	-1.42	0	-3.90
3.1	6.8	22.5	7.3	16.3	42580	-4.26	-0.41	0	-4.18
3.2	6.1	20.1	7.0	15.5	38645	-4.34	0.65	0	-4.48
3.3	5.5	17.9	6.6	14.8	34900	-4.42	-1.07	0	-4.18
3.4	4.8	15.8	6.3	14.0	31346	-4.52	0.12	0	-4.55
3.5	4.2	13.9	5.9	13.2	27983	-4.64	-0.03	0	-4.63
3.6	3.7	12.0	5.6	12.5	24811	-4.79	-0.63	0	-4.64
3.7	3.1	10.2	5.2	11.7	21830	-4.97	0.30	0	-5.04
3.8	2.6	8.6	4.9	10.9	19039	-5.20	-0.39	0	-5.10
3.9	2.2	7.1	4.5	10.1	16439	-5.51	0.49	0	-5.64
4	1.7	5.7	4.2	9.4	14030	-5.92	-0.60	0	-5.74
4.1	1.3	4.4	3.8	8.6	11812	-6.51	-2.04	0	-5.85
4.2	1.0	3.3	3.5	7.8	9784	-7.45	-0.25	0	-7.36
4.3	0.7	2.2	3.2	7.1	7948	-9.02	-0.57	0	-8.76
4.4	0.4	1.3	2.8	6.3	6302	-12.21	0.69	0	-12.63
4.5	0.2	0.6	2.5	5.5	4846	-20.81	-0.07	0	-20.74
4.6	0.0	0.2	2.1	4.7	3582	-64.71	0.02	0	-64.76
4.7	0.2	0.5	1.8	4.0	2508	-14.08	-0.40	0	-13.79
4.8	0.1	0.2	1.4	3.2	1625	-30.79	1.53	0	-33.14
4.9	0.1	0.2	1.1	2.4	933	-13.71	-0.30	0	-13.51
5	0.0	0.1	0.7	1.6	432	-17.35	0.22	0	-17.54
5.1	0.0	0.2	0.4	0.9	121	-5.64	-0.56	0	-5.48
5.2	0.0	0.1	0.0	0.1	2	-2.09	-0.33	0	-2.05

In this particular case, the car took 5.2 seconds to come to a stop.

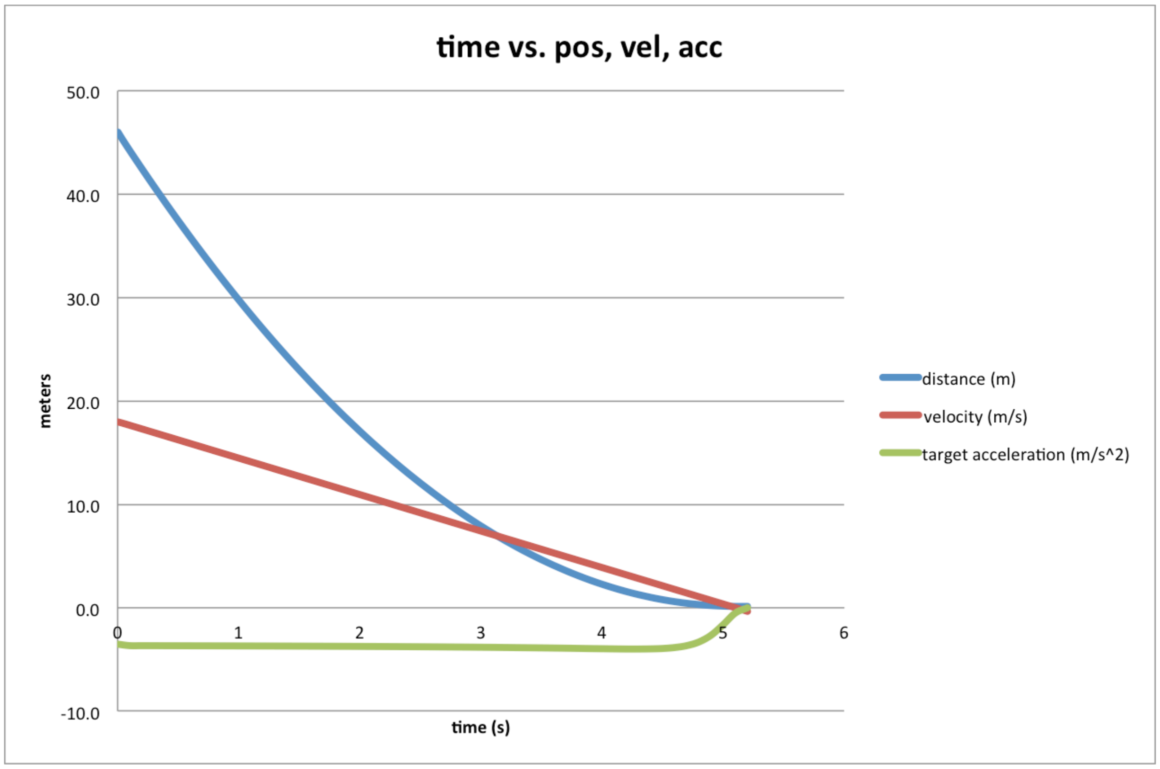
Looking at graphs of different randomly generated results shows a pattern of smooth deceleration with a deceleration spike just before the stop in many cases.



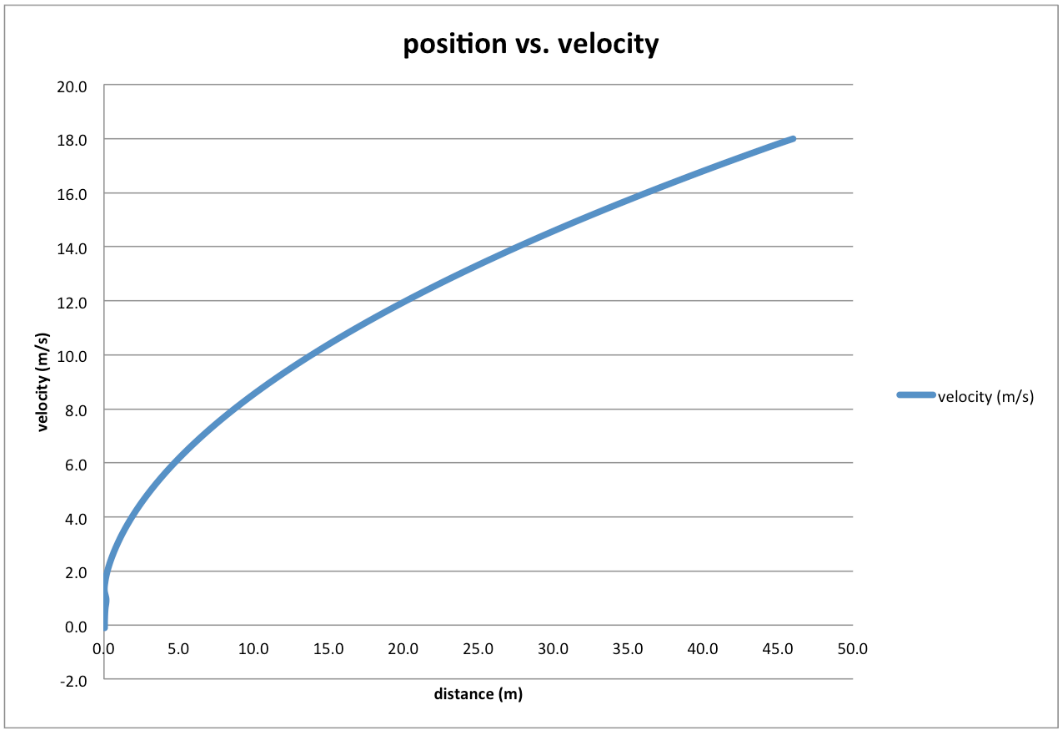
The above graph shows a gradual smooth deceleration of the car and then a single high magnitude spike in deceleration just before the stop.

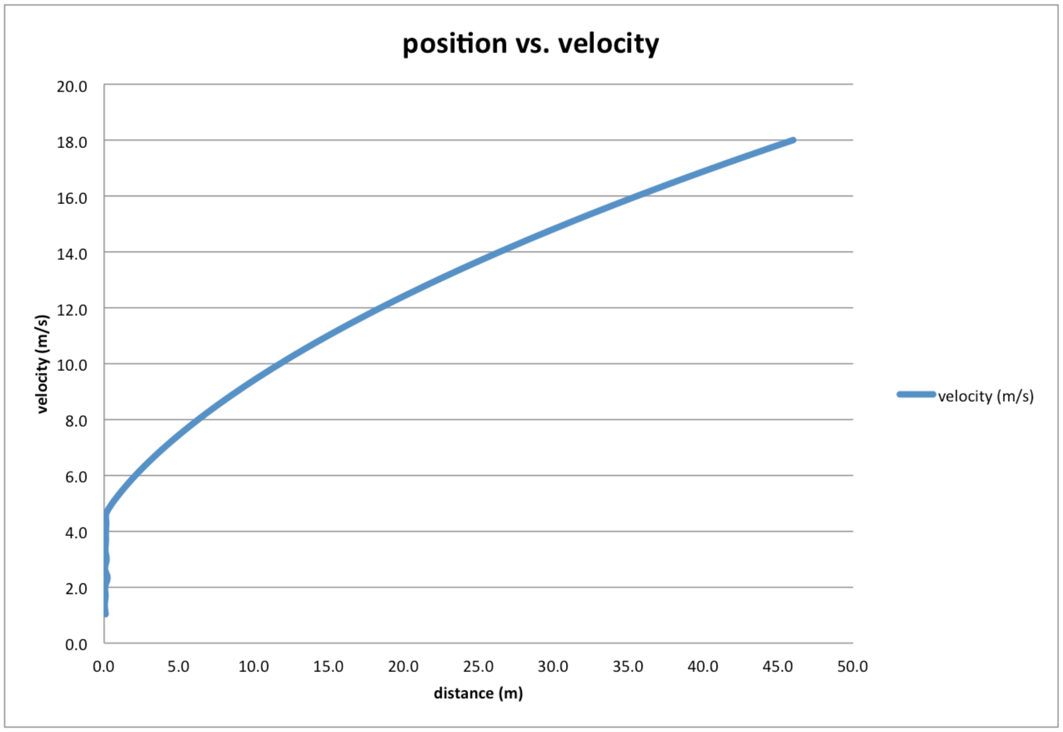


The above graph shows a gradual smooth deceleration of the car with a high magnitude spike followed by rapidly changing acceleration swings leading up to the stop.



The above graph shows a gradual smooth deceleration of the car up until the stop with no deceleration spike(s).





The above position-versus-velocity graphs show the relationship between these two quantities in the model.

Conclusion:

We can conclude that if we were to use this model to program a self-driving car to stop at a stop sign we would often see a deceleration spike just before the stop. This would likely result in, at least, discomfort for the passengers in the car. This model indicates that given certain random conditions the result could also be smooth stops or wild acceleration/deceleration spikes just before the stop.

Due to the fact that bias is randomly generated, a better model would account for the real-world variables most likely to affect the stop of the car. Once these were accounted for, we could look for additional patterns. If additional patterns appear, we could analyze those and attempt to make alterations to the model until we saw consistently smooth stops.

Bibliography:

Fundamentals of Physics Volume 1 by Halliday, Resnick, & Walker

Mathematical Modeling with Microsoft Excel by Neuwirth Arganbright

Watch: Self-driving car anticipates a crash and stops – Sarah Lee (January 5, 2017)
http://www.theblaze.com/news/2017/01/05/watch-self-driving-car-anticipates-a-crash-and-stops/

Autonomous car – (Accessed June 19, 2017)
https://en.wikipedia.org/wiki/Autonomous_car

Appendix A:

Proposal 2 was for a different project:

Credit Card Analysis – Dynamic Model

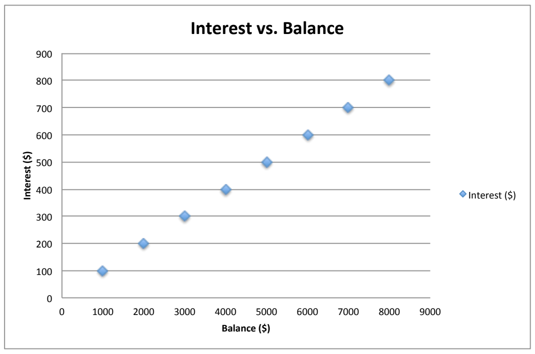
Introduction:

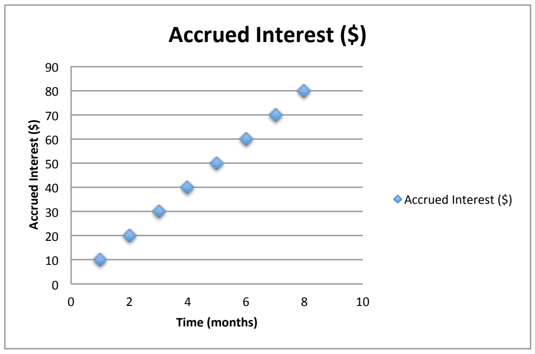
Nearly everybody has a credit card these days. Many people depend on a credit card to be able to afford necessary items and end up carrying a balance. Projecting how credit card payments affect payoff time and interest paid is of great use. This project will concentrate on forming a dynamic model that calculates these projections and has the ability to input variables for different situations.

Proposed Work:

We will start with a credit card type, balance, and interest rate. We will analyze how payments and interest affect the credit card balance. We will compare making minimum payments with larger payments. We will look at total payoff times when making different payments. We will look at extra money paid from interest in relation to how long it takes to pay off the balance.

Sketches of Possible Graphs:


Interest vs. Balance
Interest ($)
Balance ($)
◆ Interest ($)


Accrued Interest ($)
Accrued Interest ($)
Time (months)
◆ Accrued Interest ($)