Particle Filter Speed Controller for a Closed-Loop Vehicle System

Ronald Picard
Vanderbilt University
Nashville, TN, USA

Contents

***Abstract*—Particle filters provide an effective stochastic method for predicting an unknown distribution; this makes them suitable for predicting the underlying properties of dynamical systems. Utilizing these predicted properties in combination with Ackermanns formula can provide the ideal control law for producing the desired system response. We present a particle filter speed controller for predicting the underlying properties of a vehicle system and then providing the appropriate control law to set the steady-state system response equal to the reference value.**

***Index Terms*—particle filters, stochastic analysis, control systems, property estimation**

I. Introduction

Particle filters provide an effective stochastic method for predicting an unknown distribution using conditional probability. Therefore, this technique is suitable for predicting the underlying properties of dynamical systems. It is easy to verify the success of a particle filter by performing closed-loop simulation analysis with a state-space model who's properties are unknown to the particle filter controller. Utilizing these predicted properties in combination with Ackermans formula provides a control input for producing the desired system response. This technique provides illustrates a generalizable method to predict the properties of similar classes of dynamical systems. Cruise control systems are of much importance in modern vehicles. With an implementation of a particle filter speed controller, system properties (i.e. mass, damping coefficient, etc.) values do not need to be known up front but can be predicted and converged upon rapidly.

II. Problem Background

In this section we introduce some fundamental topics required to understand the different sections of this work.

A. Basics

1) Closed-Loop Control Systems: Closed-loop control systems involve feedback (usually negative) in which the measured response is subtracted from the reference value to provide an error signal to the controller. An example negative feedback control block diagram is provided in Figure 1.

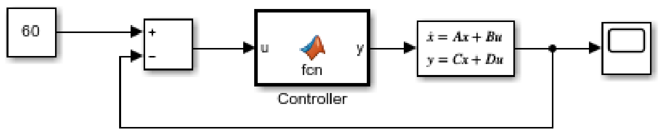

Fig. 1: Example Closed-Loop Negative Feedback Control System

2) State-Space Control Systems: A state-space model is a linear algebraic approach to analyze a system described by differential equations. The common form for the single-input single-output state-space model is shown below in Equation 1; where matrix A is the state-variable transition matrix based on the current state vector $\vec{x}$, matrix B is the state-variable transition matrix based on the input u, matrix C is the output matrix based on the state vector $\vec{x}$, and matrix D is output matrix based on the input u. [1]

$$\begin{bmatrix} \dot{\vec{x}} & = A\vec{x} + Bu \\ y & = C\vec{x} + Du \end{bmatrix} \tag{1}$$

3) Control Law: The control law takes on the form shown in Equation 2; where $\vec{K}^T$ represents a gain row vector, e is the error signal (reference signal minus system output). When the dot product of $\vec{K}^T$ and e are taken, the result is a control law input scalar that will stabilize the system according to the performance specifications.

$$u = \vec{K}^T \cdot e \tag{2}$$

4) Ackermanns Formula: Ackermanns Formula (see Equation 3) is a linear algebraic formula for calculating the control law gain row vector $\vec{K}^T$, based on the desired system pole locations. P_c is the controllability matrix defined by Equation 4, and $\lambda(A)$ is the characteristic equation with the desired pole locations. [1]

$$\left[\vec{K} = [0 \quad 0 \quad ... \quad 0 \quad 1] \cdot P_c^{-1} \cdot \lambda(A)\right] \tag{3}$$

$$\left[P_c = [B \quad AB \quad A^2B \quad ... \quad A^{n-1}B]\right] \tag{4}$$

5) Conditional Probability: Conditional probability is the probability that an event A will occur given that an event B has occurred; written $P(A|B)$. Bayes formula can be used to calculate the conditional probability (see Equation 5). [2] In control systems this probability can be calculated indirectly as the relative difference between the control input and the system response (see Equation 6). The smaller the difference, the higher the probability that the underlying properties (particles) that were used to calculate the control law (which is used to calculate the predicted response) are correct.

$$P(A|B) = \frac{P(A)P(B|A)}{P(B)} \tag{5}$$

$$V_{probability} = abs(V_{predicted} - V_{response}) \tag{6}$$

6) Particle Filters: Particle Filters are a stochastic technique used for estimating an unknown distribution. Particle filters use a finite number of particles to represent a distribution. Particles consist of a value and a weight. In the controls domain, a particle represents a guess about what some underlying state or property is, and the weight represents the probability (likelihood) that that guess is correct. After a particle is sampled (i.e. a guess is made) a corresponding action (i.e. control input) is taken using that guess. The result of that action (i.e. response of the system) is compared to the expected (predicted) result based on the particle (guess), and the weight (likelihood) of the particle is updated based on the conditional probability (likelihood) that the result (i.e. response of the system) would have resulted if the particle (guess) is correct.

7) Simulink: Simulink is a model-based design and simulation tool that provides a graphical framework for developing, simulating, and running control software with automatic code generation. The tool allows for the model-based creations of actor-oriented diagrams in which results can be displayed to a graph. The tool allows simulation time steps to be fixed-step. The following are a list of Simulink Components used for software associated with this paper: constant, scope, state-space, MATLAB function block.

III. Technical Objectives

A list of technical objects are provided below.

1) Derive, develop, and implement (in Simulink) a realistic state-space model of a vehicle.
2) Derive, develop, and implement (in Simulink with MATLAB Functions) a symbolic ideal control law formula and a discretized velocity prediction formula utilizing Ackermann's formula and the state equations, respectively.
3) Design, develop, and implement a particle filter speed controller (in Simulink with MATLAB Functions) for predicting the underlying properties of a state-space model and use these properties to control the system.
4) Perform experimental trials (in Simulink) with varied particle filter speed controller parameters and analyze the accuracy of the underlying properties predictions, as well as the standard deviation among the converged particles.

IV. Problem Description

The problem we explore in this work involves a simulated closed-loop control system for a vehicle speed controller.

A. Closed-Loop System Model

The Simulink model developed for this system is shown in Figure 2. The model includes a negative feed-back control loop. A reference speed value of 60 is subtracted from the system response speed (plus a uniform random sensor noise) to provide an error signal that is sent into the particle filter controller. See subsection IV-E for more details on the particle filter controller. The particle filter controller provides a control law signal that is subjected to a non-linear saturation function. The saturated signal is sent into the state-space model (plant) that represents the vehicle system. The output is then subject to a small uniform random noise that presents the uncertainty of the velocity sensor measurements.

B. State-Space Model

The state-space model for the vehicle system may be derived from the Newton's second law by taking the sum of all vector forces on a vehicle. Figure 3 illustrates the forces on a vehicle. We assume vehicle is on a flat surface; therefore, the vertical forces (force of gravity and the normal force) cancel out, leaving only the horizontal forces. The horizontal forces are the force due to acceleration, $m\dot{v}$ (where m is the mass of the vehicle in kg, and $\dot{v}$ is the acceleration at time t in $\frac{m}{s^2}$), the force due to friction and wind drag, bv (where b is the damping coefficient in $\frac{Ns}{m}$, and v is the velocity at time t in $\frac{m}{s}$), and the input force due to the engine providing torque on the wheels, u, in newtons. The sum of all forces results in equation 7, which is a differential equation describing the system. Setting $x1$ equal to v in equation 9, and then $\dot{x_1}$ equal to $\dot{v}$ in equation 10 we obtain equation 7 rewritten as the state-space equation 12. Setting y equal to v in equation 8 we obtain state-space output equation 13. [3] State-space equations 12 and 13 reveal the A, B, C, and D state-space matrices provided in 14 that are needed for our state-space model block in Simulink.

$$m\dot{v} + Bv = u \tag{7}$$

$$y = v \tag{8}$$

$$x_1 = v \tag{9}$$

$$\dot{x_1} = \dot{v} \tag{10}$$

$$\dot{x_1} = \frac{-b}{m}x_1 + \frac{1}{m}u \tag{11}$$

$$\dot{x} = \left[\frac{-b}{m}\right]x + \left[\frac{1}{m}\right]u \tag{12}$$

$$y = \left[1\right]x \tag{13}$$

$$A = \left[\frac{-b}{m}\right], B = \left[\frac{1}{m}\right], C = \left[1\right], D = \left[0\right] \tag{14}$$

C. Symbolic Ackermman's Formula

A symbolic form of Ackermman's formula is used to calculate the control law. Since the controllability matrix is Rank 1, P_c reduces to equation 15. Recall that P_c must be inverted for 3, which results in 16. The characteristic equation of a closed-loop system may be found by equation 17 and results in equation 18. Since this is a first order equation, we have first order system, and the characteristic equation that is utilized for the pole placement in Ackermman's formula can be simplified to equation 19. Substituting equation 16 and equation 19 into equation 3 we achieve equation 20. Substituting matrix A from 14 into equation 20 we obtain 21. Substituting equation 21 into equation 2 we obtain equation 22. Stable poles reside in the left-hand side of the complex S-plane; therefore, we choose a stable pole of -1.5 in equation 23. Substituting our pole from 23 into equation 22 we obtain equation 24. Equation 24 provides the symbolic control law formula for our system.

$$P_c = [B] = [1/m] \tag{15}$$

$$P_c^{-1} = [m] \tag{16}$$

$$\lambda(s) = det(sI - (A - B\vec{K^T})) \tag{17}$$

$$\lambda(s) = s - (\frac{-b}{m} - \frac{1}{m}\vec{K^T}) = s + (\frac{b}{m} + \frac{K}{m}) \tag{18}$$

$$\lambda(s) = s + -pole \tag{19}$$

$$\vec{K^T} = [1] \cdot [m] \cdot (A + -pole) \tag{20}$$

$$k = [1] \cdot [m] \cdot [(\frac{-b}{m}) + -pole] \tag{21}$$

$$u = k \cdot e = [1] \cdot [m] \cdot [(\frac{-b}{m}) + -pole)] \cdot e \tag{22}$$

$$pole_{stable} = -1.5 \tag{23}$$

$$u = [1] \cdot [m] \cdot [(\frac{-b}{m}) + -(-1.5)] \cdot e \tag{24}$$

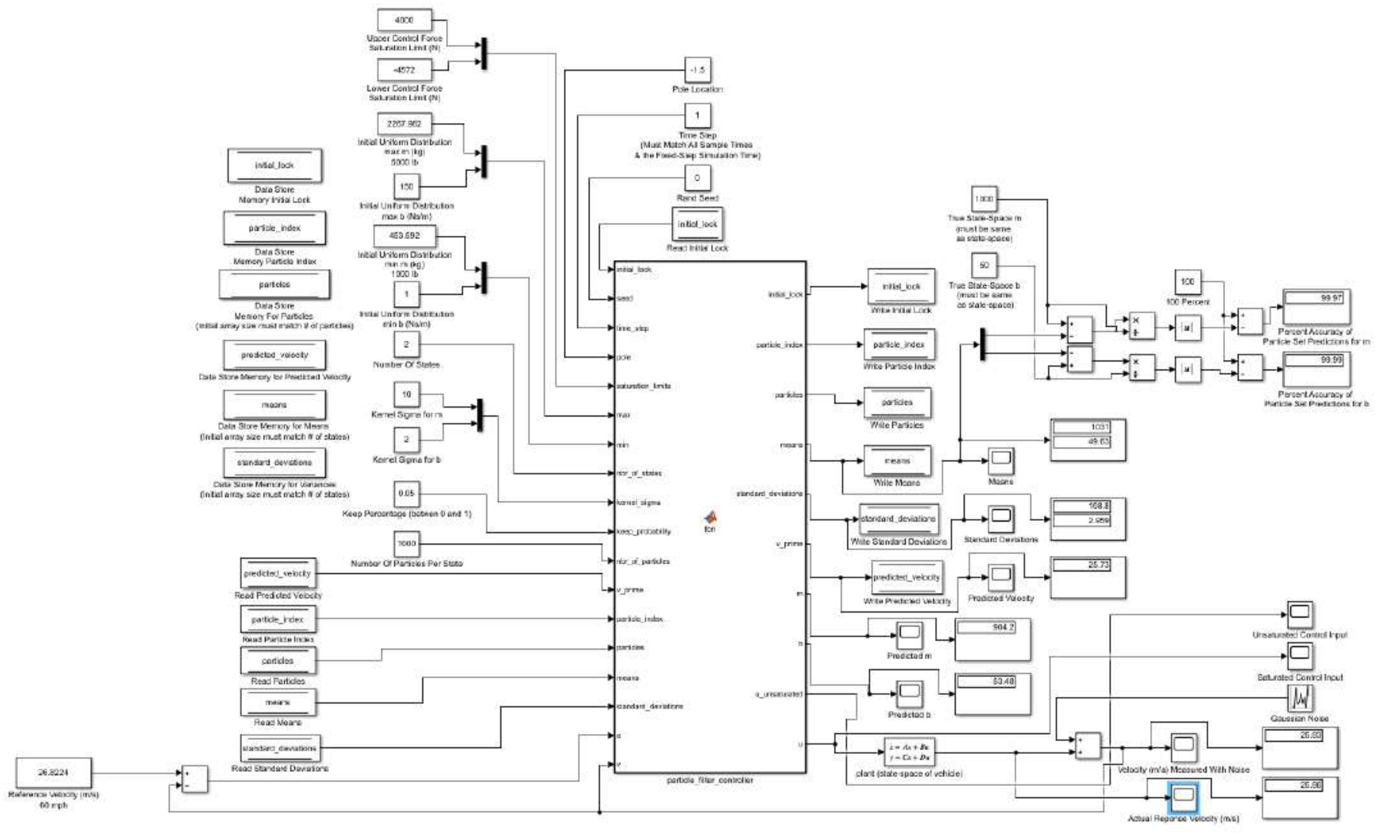
Fig. 2: Simulink Model

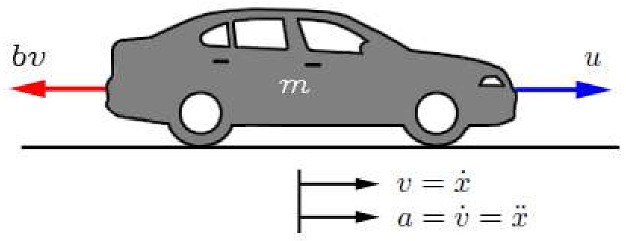

Fig. 3: Vehicle Forces [3]

D. Discretized State-Space Equations

In order to predict the response to a control input based on a particle (i.e. property guess) we must first discretize the system. Equation 12 may be rewritten as equation 25, and equation 25 may be rewritten as equation 26. Equation 26 is the discretized state-space equation that predicts the velocity based on the previous measured velocity and a given time step. See subsection IV-F assumptions about the time steps.

$$\frac{x_1(t+1) - x_1(t)}{t} = \frac{-b}{m}x_1 + \frac{1}{m}u \tag{25}$$

$$x_1(t+1) = x_1(t) + (\frac{-b}{m}x_1 + \frac{1}{m}u)t \tag{26}$$

E. Particle Filter

1) Particle Filter Use: The particle filter designed in this work is used to find the underlying properties on our the state-space system model from subsection IV-B; the mass, m, of the system in kg, and the damping coefficient, b, of the system in $\frac{Ns}{m}$. The values are can be isolated utilizing our particle filter algorithm which will find the most likely values for m and b as time progresses. IV-E2 describes the particle filter algorithm used.

2) Particle Filter Algorithm:

1) Initialize N number of particles, P (discrete distribution), with a uniform random distribution between the upper and lower bounds of the problem space.
   a) Generate N number of mass m values uniform randomly between upper and lower bounds of m in an array. (Note: Initialize a second array for the particle weights.)
   b) Generate N number of damping coefficient b values uniform randomly between upper and lower bounds of b in an array. (Note: Initialize a second array for the particle weights.)
2) For $i = 1$ to N: (Use Particles)
   a) Sample a particle, p_i, from the P.
      i) Sample mass value m_i and damping coefficient b_i from the respective arrays.
   b) Apply a control input, u_i, based on p_i.

      i) Calculate the control law u_i using equation 24 with the sampled m_i and b_i values.
   c) Compute the conditional probability (see subsection 5) that the system response that would have occurred if p_i is the true value.
      i) Calculate the predicted velocity, $v_{predicted}$,using the discretized state-space equation 26.
      ii) Calculate the particle weights with the particles using equation 6, and store them using the same index in the secondary arrays.
3) Sort the particles by the highest probability. (Filter Particles)
   a) Sort the m and b arrays by ascending order of the weight arrays. (Note: In our implementation, the particles with the lowest weights have the highest probabilities.)
4) Keep first K number of the most probable particles based on the weights.
   a) Keep the first K elements of the m and b arrays.
5) For $i = 1$ to $(N - K)$: (Resample Particles)
   a) Sample a particle, p, at uniform random from the K particles.
      i) Sample a mass value, m, at uniform random from the K left over particles.
      ii) Sample a damping coefficient value, b, at uniform random from the K left over particles.
   b) Sample a new particle, p_{new}, based on a Gaussian distribution around the sampled particle p with a specified standard deviation (kernel sigma). Then add the p_{new} to the discrete set of K particles.
      i) Sample a new mass value, m_{new}, based on a Gaussian distribution around the sampled mass value, m, with a specified standard deviation (kernel sigma). Then replace the element in the $K + i^{th}$ index of the m array with the m_{new} value.
      ii) Sample a new damping coefficient value, b_{new}, based on a Gaussian distribution around the sampled damping coefficient value, b, with a specified standard deviation (kernel sigma). Then replace the element in the $K + i^{th}$ index of the b array with the b_{new} value.
6) Repeat 2 through 5.

3) Particle Filter Implementation: Figure 2 shows all of the tunable parameters the particle filter. The most important items are listed in below. The particle filter can be tuned to use a specific number of particles per state and a specific number of states (states mean properties, so our case has 2 states). The desired pole location for the closed-loop system that is utilized in equation 22 may be specified. Minimum and maximum bound on the initial uniform random particle distributions may be specified for each state. In general, these bounds should reasonable based upon the a prioi knowledge of the system. The kernel sigma refers to the standard deviation used for sampling of a p_{new} from a Gaussian distribution around p in step 5b of the particle filter algorithm. A different kernel sigma may be specified for each state. The keep percentage refers to the number of particles kept, K, in step 4 of the particle filter algorithm. Figure 2 uses 1000 particles with 2 states, keep percentage of 0.05 (5%), a kernel sigma for m of 10, a kernel sigma for b of 2, a maximum mass bound of 2267.962 kg (5000 lb), a minimum mass bound of 453.592 kg (1000 lb), a maximum damping coefficient of 150 $\frac{Ns}{m}$, and a minimum damping coefficient of 1 $\frac{Ns}{m}$. See section VI for intuition behind choosing specific parameter values.

1) Number of Particles per State.
2) Keep Percentage.
3) Kernel Sigmas.
4) Number of States.
5) Minimum and Maximum Bounds on the Initial Uniform Random Particle Distributions.
6) Upper and Lower Saturation Limits on the Control Output.
7) Pole Location.

F. Key Assumptions

There are a few key assumptions that need to be addressed in our simulation. These assumptions are listed in below.

1) The true mass and damping coefficient values must lay within the respective initial distribution bounds.
2) The time step used in discretized state-space equation 26 the must match the Simulink simulation fixed-step value.
3) The nonlinear saturation bounds much be reasonable for controlling the system.
4) Compared Trials are all run with the same random seed.

V. Simulation Environment

A. MATLAB and Simulink

The developed model uses a Simulink block diagram (see Figure 2) with a MATLAB function controller that uses several other MATLAB Functions. The developed function files along with a basic description are provided in below.

1) particle_filter_controller.m: Performs steps 2 through 5 of the particle filter algorithm with the help of other function calls.
2) particle_resampler.m: Performs step 5 of the particle filter algorithm.
3) particle_generator.m: Perform step 1 of the particle filter algorithm.
4) update_particle_weight.m: Performs step 2c of the particle filter algorithm.
5) sort_b_like_a.m: Performs step 3 of the particle filter algorithm.

B. GitHub

The code repository containing all needed files is publicly available on GitHub under an MIT license (see appendix A.) The software was developed using MATLAB and Simulink Version 9.2 R2018b, May 24th, 2018.

TABLE I: Vehicle Parameters

Trial	Weight kg	Damping Coefficient $\frac{Ns}{m}$
1	1000	50
2	1000	50
3	1000	50
4	1000	50
5	1000	50
6	1000	50

TABLE II: Particle Filter Parameters

Trial	Number Of Particles	Kernel Sigma	Keep Percentage
1	50	2	0.1
2	50	2	0.05
3	200	2	0.1
4	200	2	0.05
5	1000	2	0.1
6	1000	2	0.05

TABLE III: Results - Part 1: Convergence Accuracy Without Noise

Trial	Weight Accuracy %	Damping Accuracy %	Time Steps (s)
1	99.84	99.97	500
2	99.82	100	500
3	99.98	100	2000
4	99.98	100	2000
5	99.98	100	10000
6	99.96	100	10000

TABLE IV: Results - Part 2: Convergence Std. Without Noise

Trial	Weight Std.	Damping Std.	Time Steps (s)
1	20.4	1.922	500
2	24.91	1.772	500
3	20.62	1.714	2000
4	23.44	1.92	2000
5	36.27	1.903	10000
6	56.89	1.849	10000

C. How to run

To run this software download or clone the GitHub repository. Open the Simulink solution file named, "project.slx" in Simulink Version 9.2 R2018b, May 24th, 2018 or newer, and click the run button.

VI. Experiments

A set of 6 Trails are presented to illustrate the functionality of the particle filter. The trials are all run twice, once without sensor noise, and once with a uniform random sensor noise addition between -0.1 and 0.1. Table I provides the true state-space parameter values used in each trial. As seen, these values are kept constant so that trials may be compared to each other. Table II provides the particle filter parameters used in each trial. The results of these trial experiments are captured in section VII. Trials 1 and 2 explore the results with 50 particles and a keep percentage of 0.1 (10%) and 0.05 (5%), Trials 3 and 4 explore the results with 200 particles and a keep percentage of 0.1 (10%) and 0.05 (5%), and Trials 5 and 6 explore the results with 1000 particles and a keep percentage of 0.1 (10%) and 0.05 (5%). An upper saturation control limit of 4000 N is placed on the controller. [4] predicts that the average rate of acceleration for ordinary cars is between 3 and 4 $\frac{m}{s}$. Taking the higher value of 4 and multiplying it by our 1000 kg car we obtain an upper saturation limit of 4000 N for the control force. [5] explains that many safety experts assume a maximum breaking deceleration of -15 $\frac{ft}{sec^2}$, which in metric units amounts to -4.572 $\frac{m}{s^2}$. Multiplying this value by 1000 kg gives us a low saturation limit of -4570 N for the control force. The discrete time step of both the simulation and controller using equation 26 is set to 1 for all trials. The the desired pole location used in equation 20 is set to -1.5 for all trials. All trials share a kernel sigma of 10 for the the mass particles, and 2 for the damping coefficient particles.

VII. Performance Analysis

Tables III and V provide the accuracy of the particle filter property predictions for the mass and damping coefficients of the underlying system after 10 resampling steps. As seen, the accuracy in every case reaches about 99% regardless of the number of particles (50 to 1000) used or the addition of uniform random sensor noise. This allows for an acceptable control law to be developed with equation 24 in the particle filter controller. As seen in section B all steady state velocities converged to the approximate reference velocity of 26.8224 $\frac{m}{s}$. Tables IV and VI provide the standard deviation of the converged mass and damping coefficient particle sets after 10 resampling steps. As seen, Trials 3 through 5 (200 and 1000 particle trials) see in increase in the standard deviation when noise is added to the sensor feedback loop, however they are relatively similar. Interestingly, Trials 1 and 2 (50 particle trials) see a decrease in standard deviation with the addition of sensor noise, indicating that low numbers of particles perform better small amounts of sensor noise are added. The differences in standard deviations between 0.1 (10%) and 0.05 (5%) keep percentages of trials with the same number of particles varies by the number of particles used and with the addition of sensor noise. For the trials without sensor noise the standard deviation of the 0.05 (5%) keep percentage trials is lower than that of the 0.1 (10%) keep percentage trials. This is to be expected because a are kept are closer together than the particles that are removed, and the lower the keep percentage, the closer the remaining particles will be. For the trials with sensor noise, for trials 1 through 4 the standard deviation of the 0.05 (5%) keep percentage trials is lower than that of the 0.1 (10%) keep percentage trials (as in the trials without sensor noise); however, for trials 5 and 6 (1000 particle trials), the standard deviation of the 0.05 (5%) keep percentage trials is higher than that of 0.1 (10%) keep percentage trials. This reveals the variability that additional sensor noise can have particle filter distributions with high numbers of particles. See appendix section B for figures of each trial.

VIII. Conclusion

A. Summary

In this work we showcased the design, implementation, and use of a particle filter cruise controller that predicts the underlying property's of a vehicle system, and provides the corresponding control law input that sets the system speed to the reference speed. We explored 6 trial runs without sensor

TABLE V: Results - Part 1: Convergence Accuracy With Noise

Trial	Weight Accuracy %	Damping Accuracy %	Time Steps (s)
1	99.83	99.97	500
2	99.82	99.98	500
3	99.88	99.98	2000
4	99.95	99.99	2000
5	99.97	100	10000
6	99.97	99.99	10000

TABLE VI: Results - Part 2: Convergence Std. With Noise

Trial	Weight Std.	Damping Std.	Time Steps (s)
1	13.32	2.523	500
2	11.73	2.382	500
3	178.3	2.611	2000
4	81.17	2.793	2000
5	77.78	2.862	10000
6	108.8	2.959	10000

noise and with the addition of uniform random sensor noise. The results showed that each of the 6 trials converged on the mass and damping coefficient properties of the system with above 99.9% accuracy after 10 resampling steps, both with and without the addition of sensor noise. We showed that we can reduce the standard deviation of the converged particle sets by decreasing the keep percentage of the particle filter algorithm, except in the case of noise addition with a high number of particles.

B. Future Work

It would be interesting to test different state-space models with more properties, including more non-linear state-space systems. It would also be interesting to find out the effect of different sensor noise distributions such as Gaussian on the results. In addition, it would be interesting to use different non-constant reference values (e.g. multiple step functions at different times) and waveforms (e.g. ramp function) to test the responsiveness of the particle filter.

C. Author's Note: What I learned

I learned, in great detail, how to develop a robust particle filter speed controller using a particle filter algorithm. I also learned how to design a system model from the ground up; including state-space derivations, realistic property estimation values, realistic non-linear control saturation values, and how to implement a function to calculate ideal control law gain, on demand, using a symbolic form of Ackermann's formula. In addition, I leaned the effects of tuning the parameters of the particle filter controller (keep percentage, number of particles, initial distribution bounds, kernel sigmas, etc.) and sensor noise have on the accuracy and standard deviations of the converged discrete particle distributions for the underlying system properties.

Appendix

A. GitHub URL

The software developed for this work is publicly available on GitHub at https://github.com/rpicard92/particle-filter-state-estimate-cruise-control.

B. Trial Figures

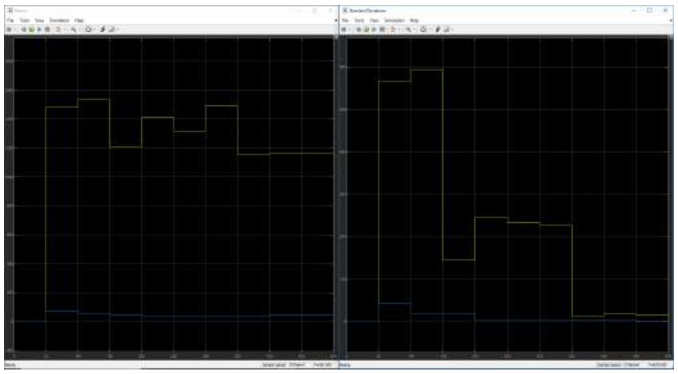
Fig. 4: Trail 1 Without Noise: M and B Particle Prediction Means and Standard Deviations Per Resampling

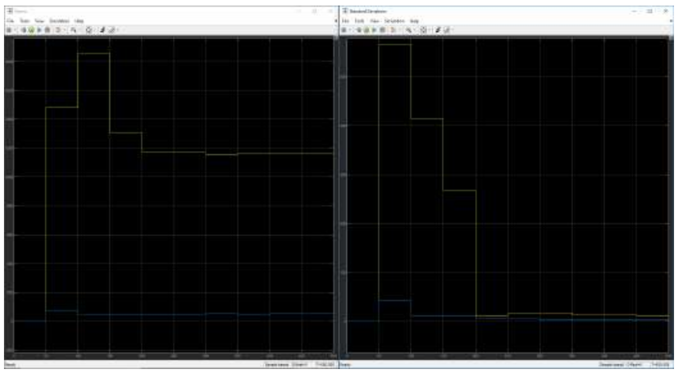
Fig. 7: Trail 1 With Noise: M and B Particle Prediction Means and Standard Deviations Per Resampling

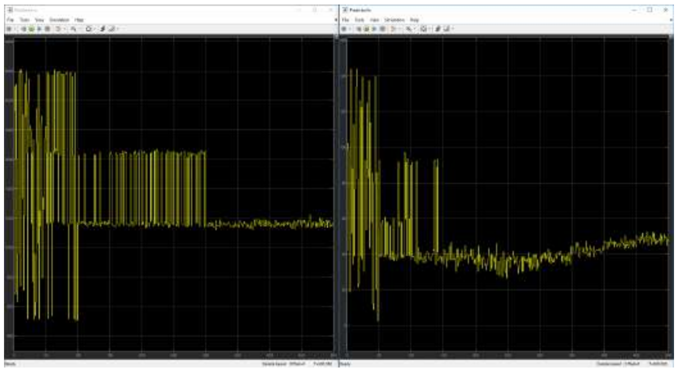
Fig. 5: Trail 1 Without Noise: Raw M and B Particle Predictions

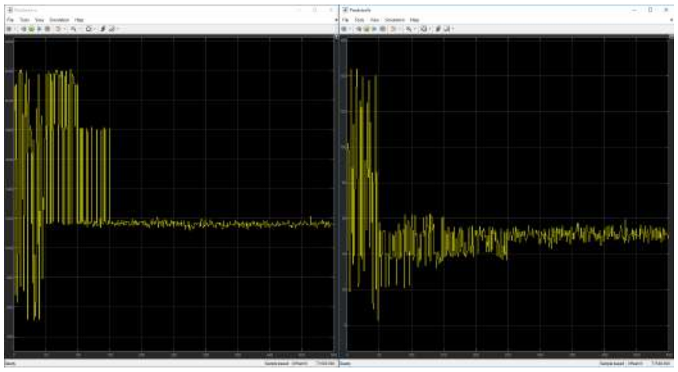
Fig. 8: Trail 1 With Noise: Raw M and B Particle Predictions

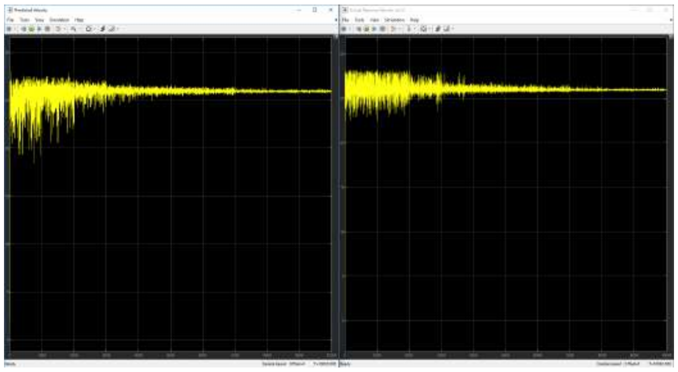
Fig. 6: Trail 1 Without Noise: Predicted Velocity and Actual Response Velocity

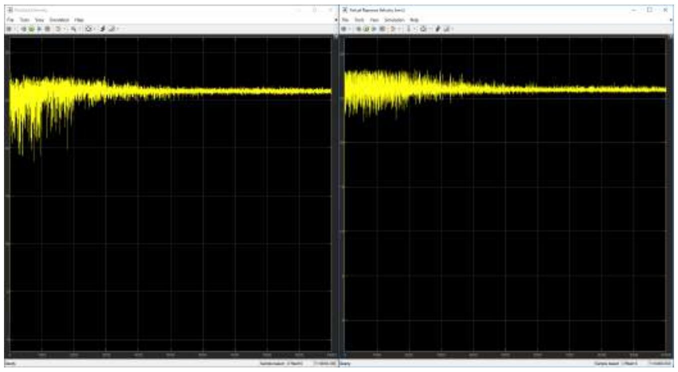
Fig. 9: Trail 1 With Noise: Predicted Velocity and Actual Response Velocity

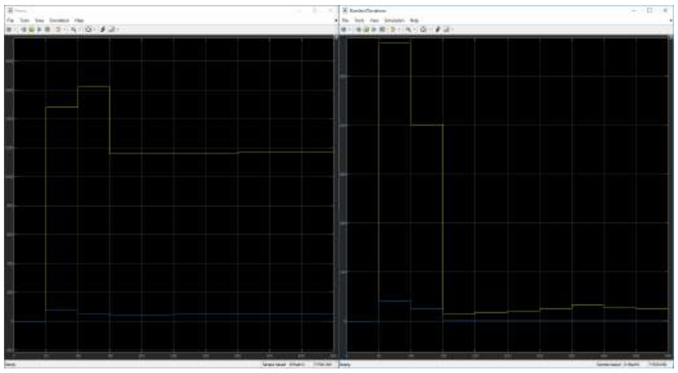
Fig. 10: Trail 2 Without Noise: M and B Particle Prediction Means and Standard Deviations Per Resampling

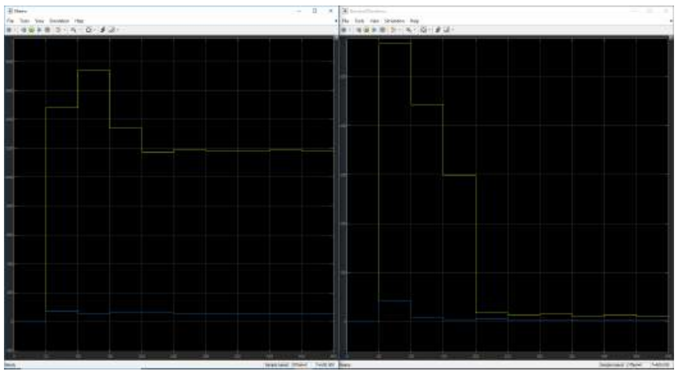
Fig. 13: Trail 2 With Noise: M and B Particle Prediction Means and Standard Deviations Per Resampling

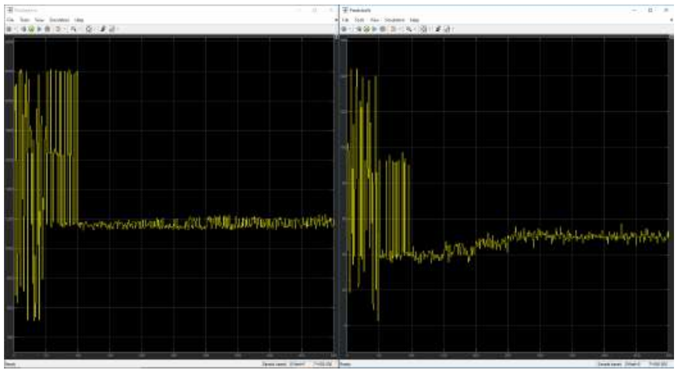
Fig. 11: Trail 2 Without Noise: Raw M and B Particle Predictions

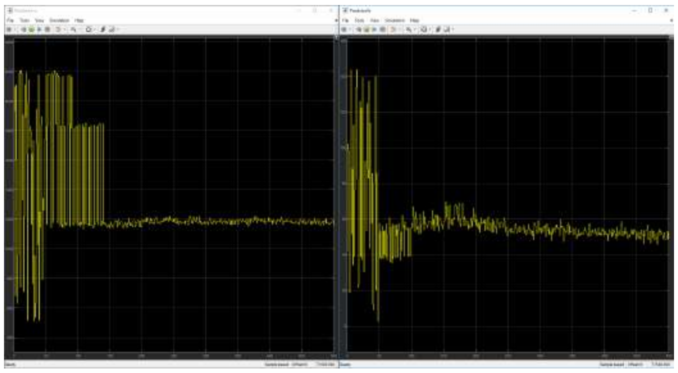
Fig. 14: Trail 2 With Noise: Raw M and B Particle Predictions

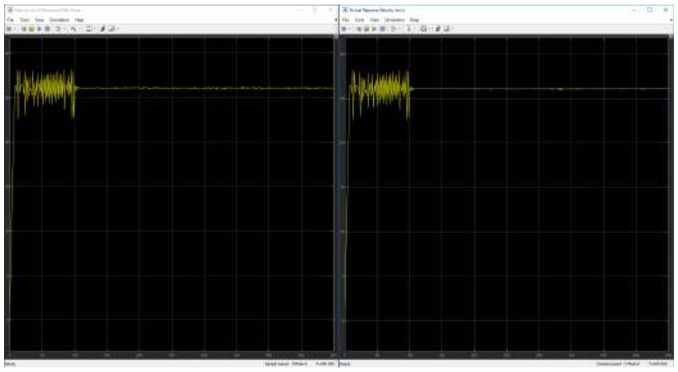
Fig. 12: Trail 1 Without Noise: Predicted Velocity and Actual Response Velocity

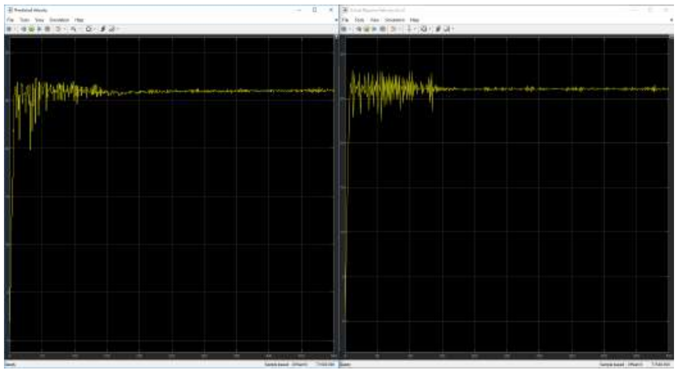
Fig. 15: Trail 1 With Noise: Predicted Velocity and Actual Response Velocity

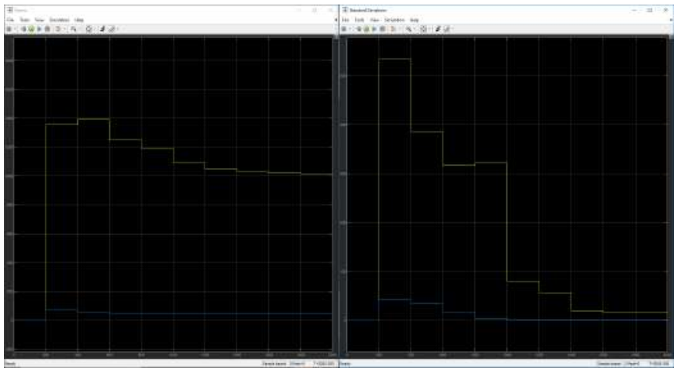
Fig. 16: Trail 3 Without Noise: M and B Particle Prediction Means and Standard Deviations Per Resampling

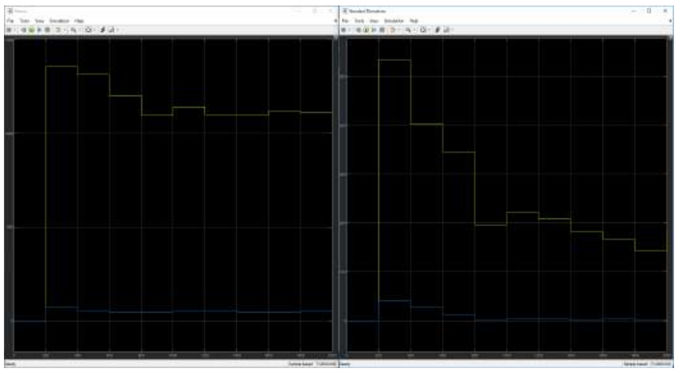
Fig. 19: Trail 3 With Noise: M and B Particle Prediction Means and Standard Deviations Per Resampling

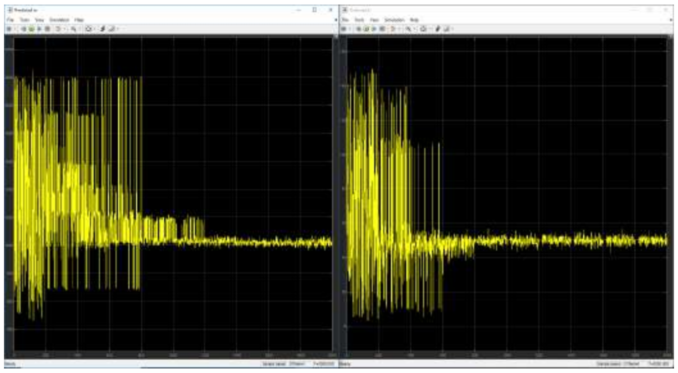
Fig. 17: Trail 3 Without Noise: Raw M and B Particle Predictions

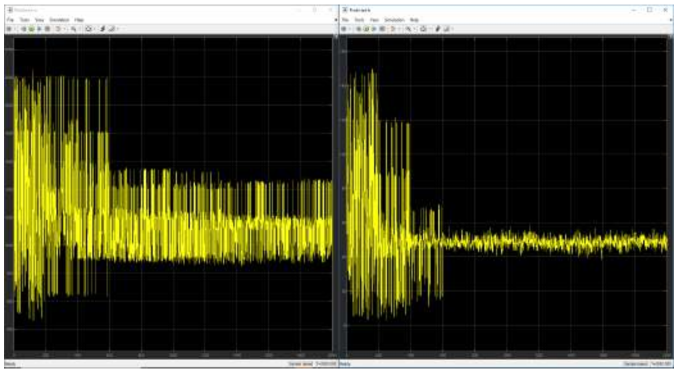
Fig. 20: Trail 3 With Noise: Raw M and B Particle Predictions

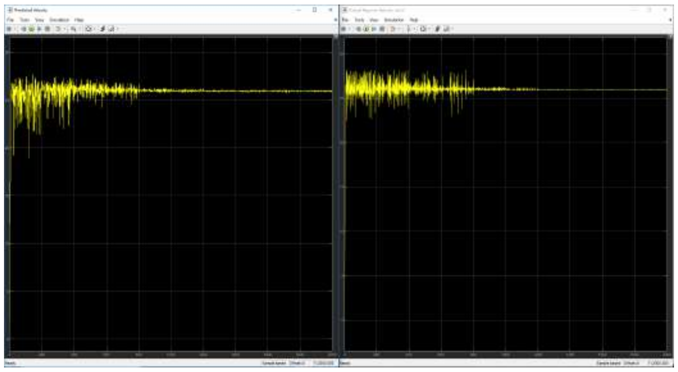
Fig. 18: Trail 3 Without Noise: Predicted Velocity and Actual Response Velocity

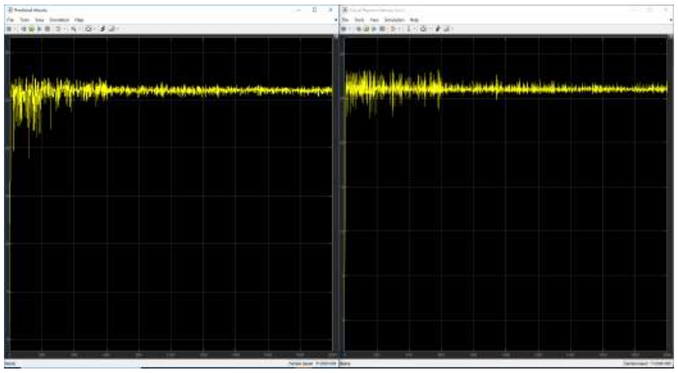
Fig. 21: Trail 3 With Noise: Predicted Velocity and Actual Response Velocity

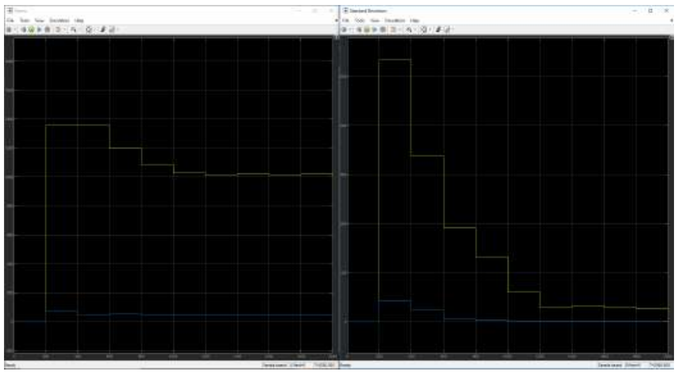
Fig. 22: Trail 4 Without Noise: M and B Particle Prediction Means and Standard Deviations Per Resampling

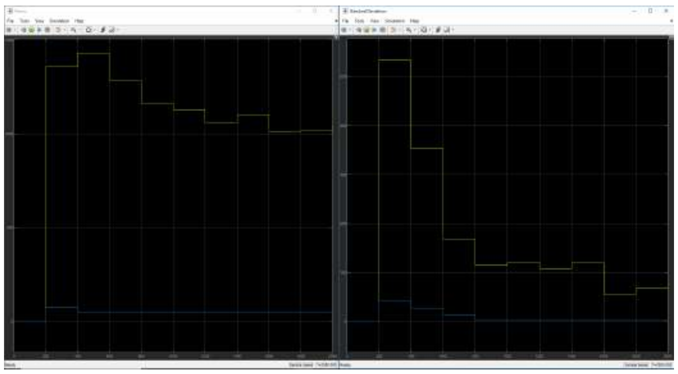
Fig. 25: Trail 4 With Noise: M and B Particle Prediction Means and Standard Deviations Per Resampling

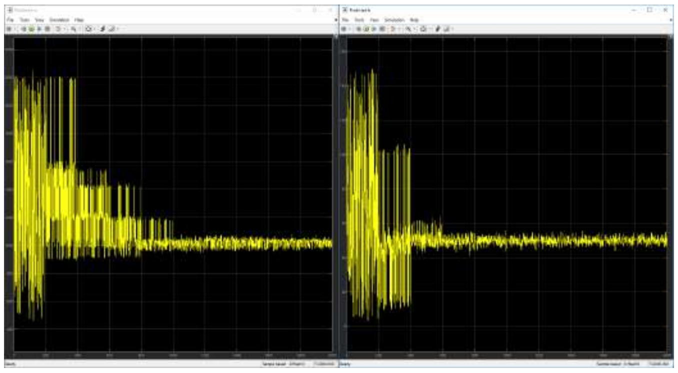
Fig. 23: Trail 4 Without Noise: Raw M and B Particle Predictions

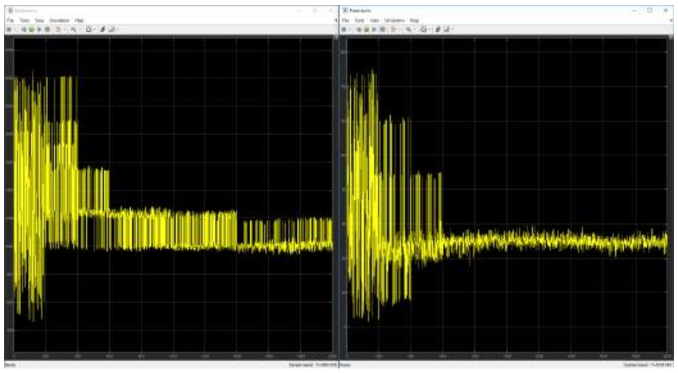
Fig. 26: Trail 4 With Noise: Raw M and B Particle Predictions

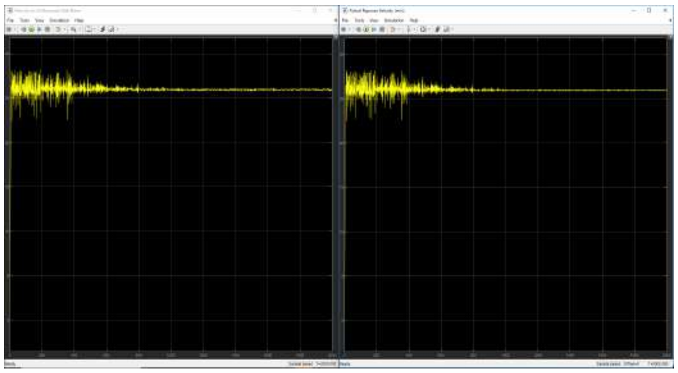
Fig. 24: Trail 4 Without Noise: Predicted Velocity and Actual Response Velocity

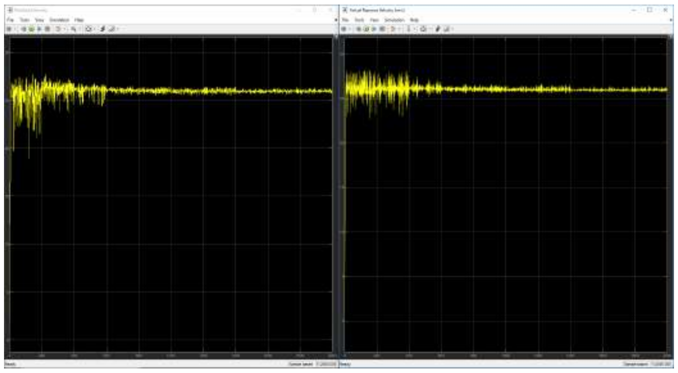
Fig. 27: Trail 4 With Noise: Predicted Velocity and Actual Response Velocity

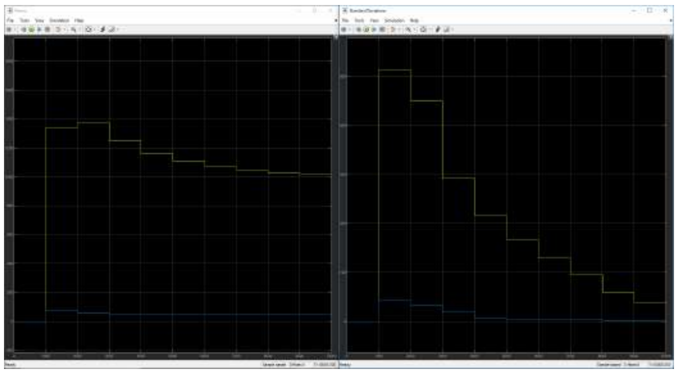
Fig. 28: Trail 5 Without Noise: M and B Particle Prediction Means and Standard Deviations Per Resampling

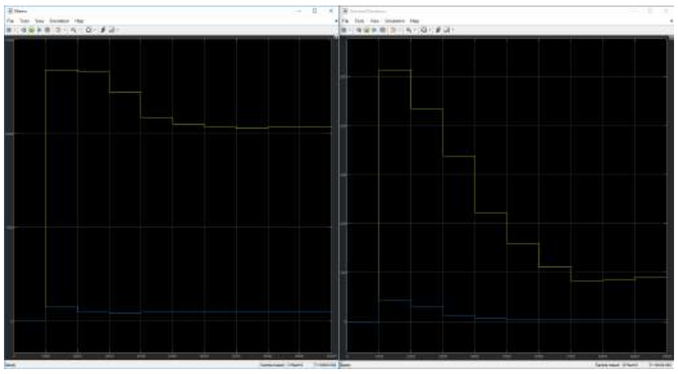
Fig. 31: Trail 5 With Noise: M and B Particle Prediction Means and Standard Deviations Per Resampling

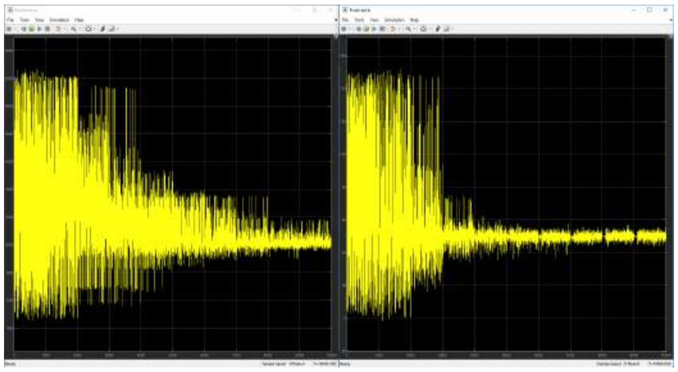
Fig. 29: Trail 5 Without Noise: Raw M and B Particle Predictions

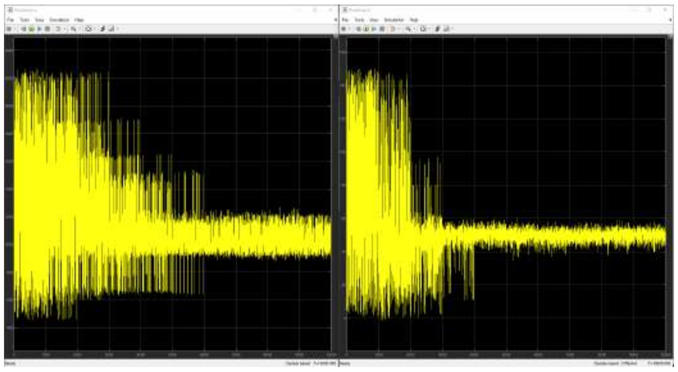
Fig. 32: Trail 5 With Noise: Raw M and B Particle Predictions

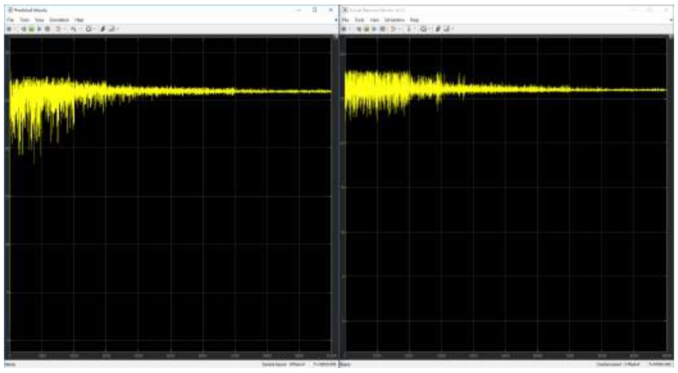
Fig. 30: Trail 5 Without Noise: Predicted Velocity and Actual Response Velocity

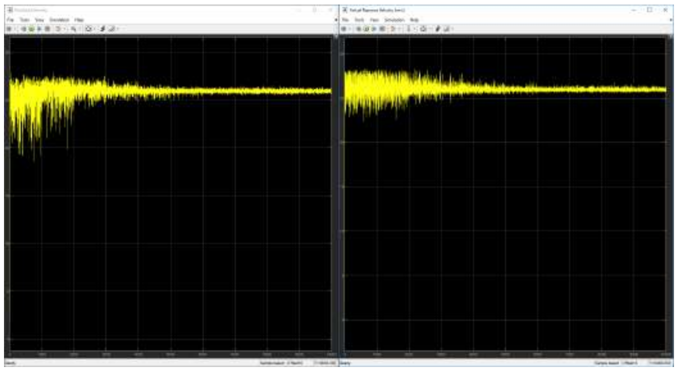
Fig. 33: Trail 5 With Noise: Predicted Velocity and Actual Response Velocity

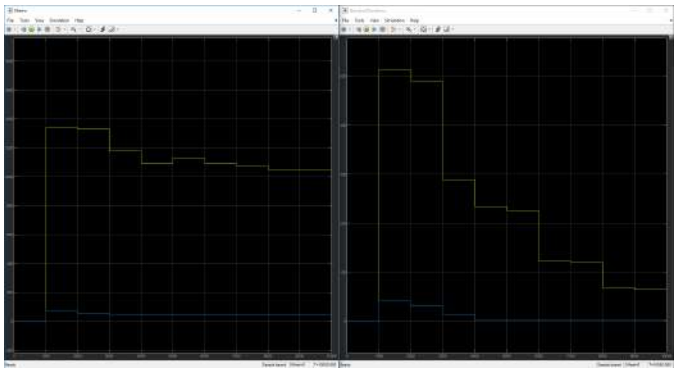
Fig. 34: Trail 6 Without Noise: M and B Particle Prediction Means and Standard Deviations Per Resampling

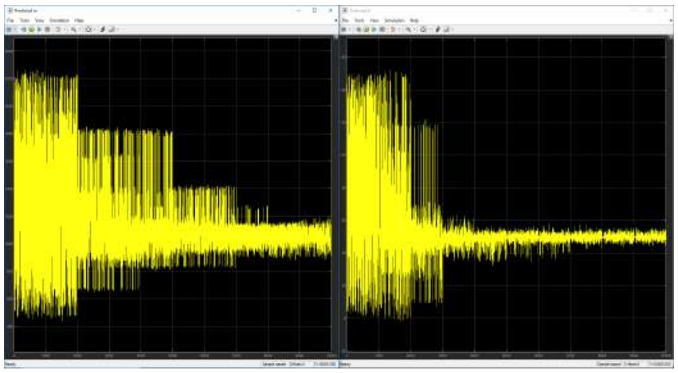
Fig. 35: Trail 6 Without Noise: Raw M and B Particle Predictions

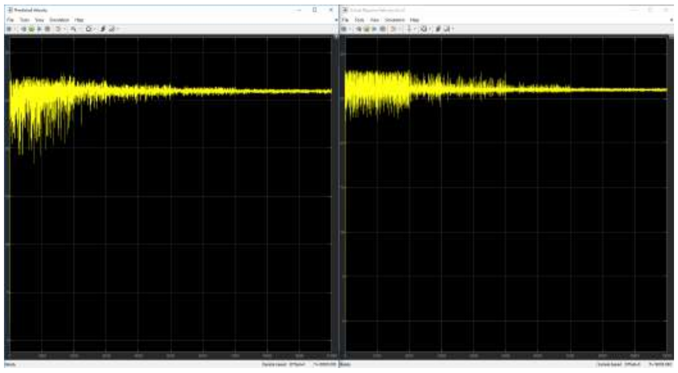
Fig. 36: Trail 6 Without Noise: Predicted Velocity and Actual Response Velocity

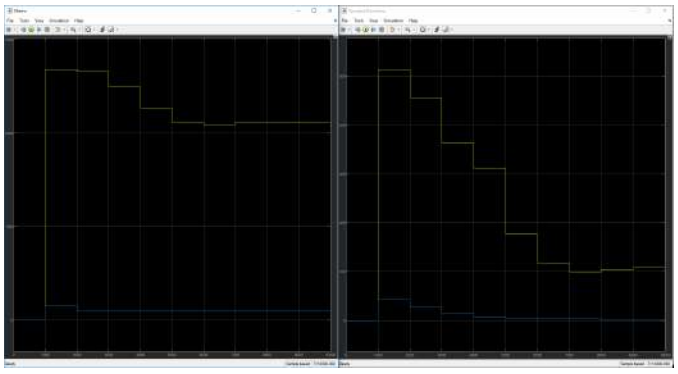
Fig. 37: Trail 6 With Noise: M and B Particle Prediction Means and Standard Deviations Per Resampling

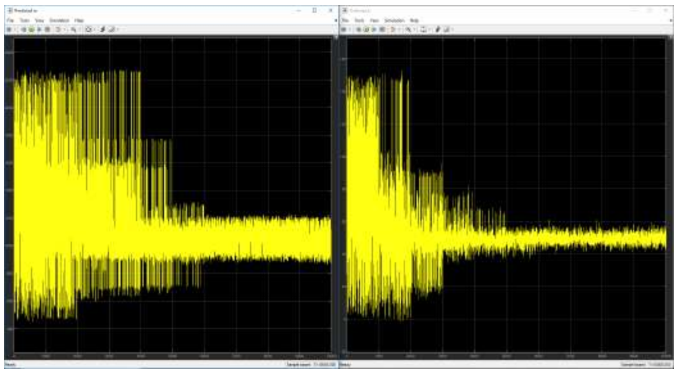
Fig. 38: Trail 6 With Noise: Raw M and B Particle Predictions

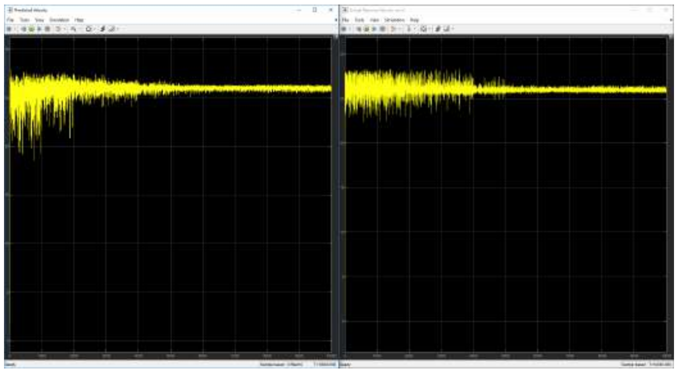
Fig. 39: Trail 6 With Noise: Predicted Velocity and Actual Response Velocity

References

[1] R. C. Dorf and R. H. Bishop, Modern Control Systems, 9th ed. Upper Saddle River, NJ, USA: Prentice-Hall, Inc., 2000.

[2] "Bayes' theorem." [Online]. Available: https://www.mathsisfun.com/data/bayes-theorem.html

[3] "Cruise control: System modeling." [Online]. Available: http://ctms.engin.umich.edu/CTMS/index.php?example=CruiseControl&section=SystemModeling

[4] "Acceleration of a car." [Online]. Available: https://hypertextbook.com/facts/2001/MeredithBarricella.shtml

[5] "Acceleration parameters." [Online]. Available: https://hypertextbook.com/facts/2001/MeredithBarricella.shtml