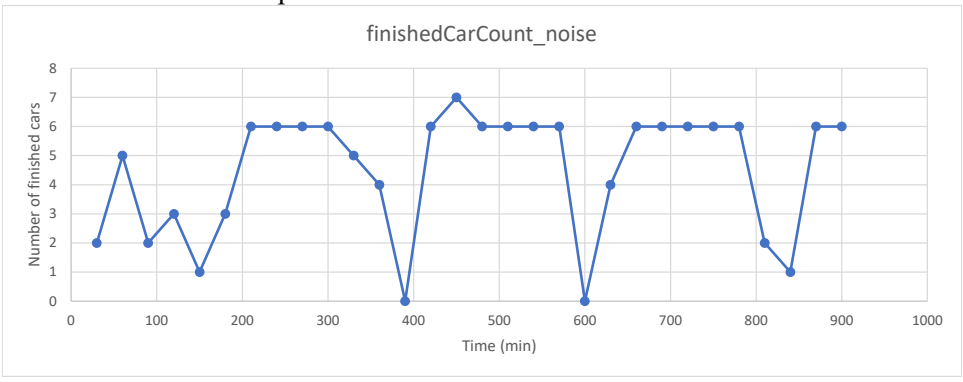
Homework 3 (Due: Nov. 13 - Thursday, 11:59pm)
CSC8840, Fall 2025
Dr. Xiaolin Hu

Consider a car wash center that is a variation of the "carwash center with schedule" model in the DEVSJAVA Lab.

- Only cars are serviced. We do not consider truck in this homework.
- The center washes a new car (if there is car to be washed) if it is not at break time. The center takes breaks based on the following rules: 1) if it has worked for 120 or more minutes and there is no car in the queue, it takes a break; 2) if the center has worked for 180 or more minutes, it takes a break regardless of whether the queue is empty or not. Note that if the center does not take a break during the middle of washing a car – it always needs to finish washing the car that it has started.
- The break time is a constant 40 minutes.
- The time it takes for the center to wash a car is a random number, modeled by a truncated normal distribution that has mean $\mu = 5.0$ (minutes), variance $\sigma^2 = 1.0^2$ and lies within the range of [3.0, 7.0].
- The carwash center is modeled by the `carWashCenter_VaryingSchedule` model, which is provided to you (in the PFDataAssimilation package).
- Cars arriving at the center randomly, modeled by a Poisson distribution that has an arriving rate λ. For example, when $\lambda = 1/6.5$, it means in average 1 car arrives in every 6.5 minutes. This is modeled by the `carGenr` model, which is provided to you.

The carwash center is not open to the public, so one cannot see what is going on within the carwash center. However, an observer (sensor) is available at the exit of the carwash center to report the number of cars that have been finished. The observer reports every 30 minutes to tell the number of finished cars in the past 30 minutes. Below is a sample observation data.



This homework uses Particle Filter to assimilate the observation data to estimate the real-time state of the carwash center, including the number of cars in the queue, the phase (working or break), and elapsed time

in the phase. Such estimation is useful to initialize simulations for predicting the future behavior of the carwash center.

We use the identical twin experiment to carry out this work. The identical twin experiment uses a simulation model (often the same simulation model used in data assimilation) to act as the "real system" to generate observation data. This simulation model is called the "true system" model. In this homework, the "true system" model uses a set of random seeds that are unknown to the data assimilation. To generate observation data and the true state data, run the test.java by instantiating the carWashSys_Real model. The resulting data are saved by the transducer_Real_dataSaving.java into the observationData.txt. Note that we have predefined two test cases (Case I and Case II), whose random seeds are defined in configData.java. Feel free to define your own test cases but do not make changes to the two pre-defined cases.

Your task is to program the PF algorithm to carry out the data assimilation. A skeleton of the PF algorithm is provided to you. You need to finish the details of the PF algorithm and carry out experiment to estimate the state for Case I and Case II. Note: an example of the PF algorithm is provided in the PF_Vehicle_Temperature.java to help you understand the PF algorithm.

This homework includes three parts as defined below.

1. Part 1: finish implementing the PF_skeleton.java and collect state estimation results for Case I and Case II. Show your results in figures similar to the figures at the end of this document (sources of the sample figures: Dynamic Data-Driven Simulation: Real-Time Data for Dynamic System Analysis and Prediction). Briefly analyze the results (a small paragraph for each figure).
2. Part 2: apply the PF algorithm to a variation of the `carWashCenter_VaryingSchedule` model, where the average time of washing a car is related to the queue size: the larger the queue size, the faster the washing speed. To obtain the t variation of the `carWashCenter_VaryingSchedule` model, simply restore the commented-out code in the `getCarWashTime()`. Do not make changes to the PF algorithm that you have implemented. Similar as in Part I, collect the state estimation results for Case I and Case II, and show your results in figures and provide brief analysis.
3. Part 3 (bonus points): 1) use the original `carWashCenter_VaryingSchedule` model to act as the "true system" model to generate observation data, and use the variation of the `carWashCenter_VaryingSchedule` model to carry out data assimilation. 2) use the variation of the `carWashCenter_VaryingSchedule` model to act as the "true system" model to generate observation data, and use the original `carWashCenter_VaryingSchedule` model to carry out data assimilation.

Submission Instruction:

- Submit your implementation of the PF algorithm (a single java file). Name your java class as PF_2025_xxx, where xxx is your first name initial plus last name. For example, if your name is Tony Imbesi, the class name should be PF_2024_TImbesi.
- Besides the PF algorithm source code, submit a PDF document showing your data assimilation results (in figures) and brief analysis.
- Due date: **Nov. 13 - Thursday, 11:59pm**.
- On Nov. 17, the TA will demonstrate your PF algorithm. We will run different "real system" scenarios and use your algorithm to estimate the real system's state. Based on the demonstration, we will assign an initial grade for your submission.
- Late submission would be accepted within one week after the due date, and will receive a late penalty of 15%.

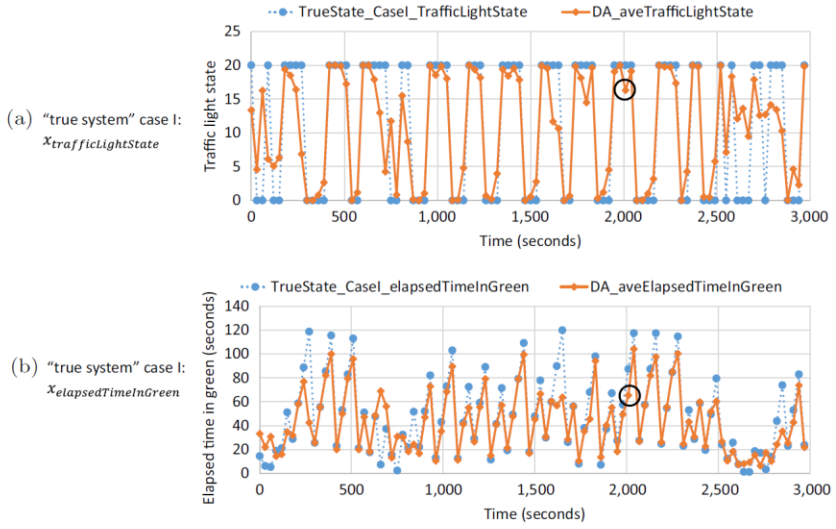


Figure 7.13. State estimation result for $x_{trafficLightState}$ and $x_{elapsed\ TimeInGreen}$. In the figure, the "true" state values are denoted with the prefix "TrueState_"; the results from data assimilation are denoted with the prefix "DA_".

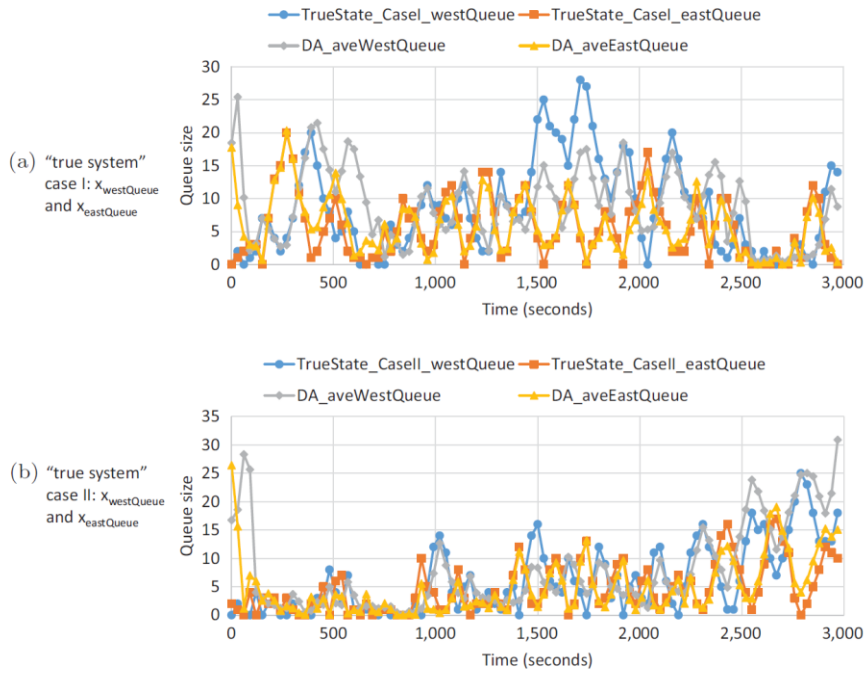


Figure 7.14. State estimation results for $x_{westQueue}$ and $x_{eastQueue}$.