README for "Scaling the Kalman filter for large-scale traffic estimation" JAVA source code

Ye Sun and Daniel B. Work

July 22, 2016

Abstract

This document describes the implementation of the distributed local Kalman consensus filtering algorithm for traffic density estimation, introduced in the article "Scaling the Kalman filter for large-scale
traffic estimation" by Sun and Work, submitted to the IEEE Transactions on Control of Network Systems. The source code is available to be downloaded at https://github.com/yesun/DLKCF.

1 License

2 Publishing results using this software

We kindly ask any future publications using this software include a reference to the following publication:

Y. Sun and D.B. Work, "Scaling the Kalman filter for large-scale traffic estimation," *submitted to the IEEE Transactions on Control of Network Systems*, 2016.

3 General Instruction

3.1 General usage notes

1. This release of software is intended to complement a paper submission to the the IEEE Transactions on Control of Network Systems;
2. This software is intended to be run on Eclipse. It was developed using Eclipse 4.3.0, and the development environment is Java SE Development Kit 7u25.

3.2 How to use the software

1. Download the folders *DLKCF_compare_LKF*, and import it into Eclipse. The other two folders are the Luenberger observer and the DLKCF used to compare with the Luenberger observer, which are for the previous versions of the manuscript.
2. The folder *MatlabPlot* includes matlab code to generate plots (from the csv files generated by the code) for the true densities and the density estimates;
3. To generate Fig. 1a in the paper, run *Test.java* in any of the three projects and plot the generated *trueState.csv* in the folder *results* using *Main.m* in the folder *MatlabPlot*. Note that *trueState.csv* should be put in the folder *Result_Plot*, same for all other plots generated using *Main.m*;
4. The csv file currently in *Result_Plot* can serve as a demonstration to generate Fig. 1a (*trueState.csv*) in the manuscript.

4 Package List

1. *DoubleMatrix*, some operations on matrix and *GaussianGenerator.java* for generating Gaussian (and other types of) noise;

2. *trueSolutionCTM*, where true solution of the entire road network is computed by the Godunov scheme in equation (3)-(4) of the paper;

3. *trueSolution*, true solution in each section obtained by selecting the part of the true solution computed by *trueSolutionCTM* that is within the local section.

4. *section*, used to specify the mode, and to compute the matrices A, B^{ρ}, and B^{q} in the SMM;

5. *model*, used to specify parameters used for estimation (e.g. model/measurement error covariance matrix, output matrix, initial guess of the estimation, and system dynamics used in estimation);

6. *filters*, the filtering algorithms for each agent, the entire estimation process is in *Estimation.exportResult*, which includes all the steps in estimation and functions to export result.

5 Parameters Instruction

5.1 Network setup

5.1.1 The DLKCF and the DLKCF-0

1. *Number of cells in each section*: 28;

2. *Number of overlapping cells between adjacent sections*: 10;

3. *Number of sections in the entire freeway network*: 7;

4. *The index of the section with a status transition located initially*: 3;

5. *Length of the time steps*: 1;

6. *Length of each cell*: 1000d/((double)*cells*), where $cells = 136$ is the total number of cells in the entire freeway network.

Parameters changing instructions: The above listed parameters for the DLKCF can be changed in lines 18-39 in *Test.java* in *DLKCF_compare_LKF* (the DLKCF-0 and the DLKCF corresponds to variables indexed by 1 and variables with no indexes, respectively).

5.1.2 The local KF

1. *Number of overlapping cells between adjacent sections*: 1;

2. *Number of sections in the entire freeway network*: 5;

3. *The index of the section with a status transition located initially*: 2;

4. Other parameters are the same as in Section 5.1.1.

Parameters changing instructions: The above listed parameters for the local KF can be changed in lines 18-39 in *Test.java* in *DLKCF_compare_LKF* (the local KF corresponds to variables indexed by 2).

5.2 Initial condition of the true state

Initial density (normalized) of the 136 cells (indexed from 0 to 135) are detailed below:
cell0-cell4: 0.2; cell5-cell67: 0.8; cell68-cell129: 0.2; cell130-135: 0.35
Parameters changing instructions: For the DLKCF, DLKCF-0, and the local KF, the initial condition can be changed in the method *TrueSolutionCTM.initial* in the package *trueSolutionCTM*.

5.3 Boundary condition of the true state

5.3.1 The upstream boundary condition

The inflow from the upstream boundary is given by the minimum of a sinusoidal flow and the receiving capacity of the upstream cell:

$$\text{inflow} = \min\left\{0.1125 + 0.1125 \times \sin(\frac{k\pi}{4000} + \pi), R(\rho_k^0)\right\},$$

where k is the time index, $R(\rho_k^0)$ is the receiving capacity of the upstream cell at the current time step.

5.3.2 The downstream boundary condition

It is assumed that the density of the downstream cell is the same as its downstream neighboring cell, and the outflow is computed by the minimum of the sending capacity of the downstream cell and the receiving capacity of its downstream neighboring cell.
Parameters changing instructions:
The boundary condition can be changed in *Estimation.exportResult* (lines 672-721).

5.4 Parameters and perturbed parameters in the traffic model

1. *Maximal density*: 1 (true), 0.9 (perturbed for sections indexed by even numbers) and 1.1 (perturbed for sections indexed by odd numbers);

2. *Critical density*: 0.225 (true), 0.2 (perturbed for sections indexed by even numbers) and 0.3 (perturbed for sections indexed by odd numbers);

3. *Maximal speed*: 1 (true), 1.2 (perturbed for sections indexed by even numbers) and 0.9 (perturbed for sections indexed by odd numbers).

Parameters changing instructions:

1. Changing parameters in the traffic model used to generate true solution can be done in the constructors *TriangularTrueCTM* and/or *TriangularTrue*;

2. Changing parameters used in the estimators can be done by setting the default parameters in the classes *FF*, *CC*, *CF*, and *FC* that extend the class *Section*;

3. The perturbed parameters in **DLKCF_SectionV_B** can be set in the method *setSections()* of the class *TrueSolution.java*.

5.5 Important settings for estimation

1. *Model error covariance matrix*: $0.0025I$ (DLKCF, DLKCF-0 and local KF);
   Parameters changing instructions: The model error covariance matrix can be updated at each time step by *updateModelVar()* in *RoadModel.java*;

2. *Sensors are located* at cells (locally) indexed by 0, 9, 18, 27 in each local section;
   Parameters changing instructions: The sensor locations can be changed in the method *getMeasurementsAll()* in *TrueSolution.java* by selecting the cells whose densities are queried;

3. *True measurement error covariance matrix* (used to generate sensor data): $0.0009I$ (for the low quality sensors, the measurement error standard deviation is changed to 0.09);
   Parameters changing instructions: The true measurement error covariance matrix can be changed in *initial()* in *TrueSolutionCTM.java*;

4. *Measurement error covariance matrix* (used in the filter for estimation): When low quality agents exist, the measurement error covariance matrix is set to be $0.0009I$ for low-quality sensors. When low quality agent does not exist, the measurement error covariance matrices are assumed to be the same with the true measurement error covariance matrices;
   Parameters changing instructions: The measurement error covariance matrix used in estimation, when different from the true measurement error covariance matrix (applicable in the presence of low-quality agents) can be changed in in the method *initialThis()* in *D.java*;

5. *Initial estimates* is defined as a linear interpolation added by some randomly generated Gaussian noise (see *defineVarAndMean()* in *RoadModel.java* for details). When the initial mode of a local section is in FC, the initial estimates of the first three cells in that section are set to be zero, and the initial estimates of the other cells in that section are set to be one;
   Parameters changing instructions: The initial estimates can be changed in *defineVarAndMean()* in *RoadModel.java*;

6. *Upper bound $\hat{c}$of the magnitude of the consensus gain* is set to be 0.01;
   Parameters changing instructions: The upper bound $\hat{c}$ can be changed by setting the value for the variable *threshold* in the method *initial()* in *Filter.java*.

5.6 Important settings in constructors

This subsection is applicable when low quality sensors and agents may exist, and parameters in the traffic model may be perturbed.

1. The last parameter in the constructor *TriangularTrueCTM* is 0 if there is no low quality sensors, and is set to be 1 if the reverse is true;

2. The last three parameters in the constructor *TriangularTrue* stand for whether parameters in the traffic model is perturbed (0 for false and 1 for true), whether low quality agents exits in the network (0 for false and 1 for true) and whether the agents run independently without inter-agent communication.