OpenBDLM: an open-source MATLAB software for time-series analysis using Bayesian dynamic linear models

Ianis Gaudot, Luong H. Nguyen, James-A. Goulet
Polytechnique Montreal

November 16, 2018

Contents

1 What is OpenBDLM ?

OpenBDLM is a MATLAB open-source software designed to use Bayesian Dynamic Linear Models for long-term time series analysis (i.e time step in the order of one hour or higher). OpenBDLM is capable to process simultaneously any time series data to interpret, monitor and predict their long-term behavior. OpenBDLM also includes an anomaly detection tool which allows to detect abnormal behavior in a fully probabilistic framework. OpenBDLM is available for download from GitHub.

Keywords: time series analysis and forecasting, linear gaussian state-space models, time-series decomposition, anomaly detection, filtering, smoothing, Bayes, gaussian conditionnals

2 Installing OpenBDLM

These instructions will get you a copy of the project up and running on your local machine for direct use, testing and development purposes.

2.1 Prerequisites

MATLAB (version 2016a or higher) installed on Mac OSX or Windows

2.2 Installing

1. Remove from your MATLAB path all previously OpenBDLM versions
2. Extract the ZIP file (or clone the git repository) somewhere you can easily reach it.
3. Add "OpenBDLM-master" folder and all the subfolders to your path in MATLAB:
   - using the "Set Path" dialog box in MATLAB, or
   - by running `addpath` function from the MATLAB command window

3 Getting started

3.1 Call OpenBDLM

- enter in the folder "OpenBDLM-master"
- type `[data, model, estimation, misc] = OpenBDLM_main();` in the MATLAB command line.

The OpenBDLM starting menu should appear on the MATLAB command window (see Listing 1).

3.2 Demo

In the MATLAB command line, type `run_DEMO;` to run a demo. Some messages on the MATLAB command window showing that the programs runs properly should appear on the MATLAB command window (see Listing 2).

4 OpenBDLM inputs and outputs

4.1 Inputs

OpenBDLM_main accepts three types of input

```
_____________________________________________
     Starting OpenBDLM
_____________________________________________
            Structural Health Monitoring using
            Bayesian Dynamic Linear Models
_____________________________________________
- Start a new project:

     *      Enter a configuration filename
     0   -> Interactive tool

- Type D to Delete project(s), V for Version control, Q to Quit.

     choice >>
```

Listing 1: OpenBDLM starting menu on MATLAB command window when calling `[data, model, estimation, misc] = OpenBDLM_main();`

- no input `OpenBDLM_main();` The program then runs in interactive mode, in which online user's interactions from the command line is required to perform the analysis.
- a configuration file as input, `OpenBDLM_main('CFG_DEMO.m');`. The configuration file is used to initialize the project, and the program then runs in interactive mode. Configuration file must follow a specific format
- a cell array as input `OpenBDLM_main({'''CFG_DEMO.m''','3','1','''Q'''});`. The program runs in batch mode, in which pre-loaded commands stored in the input cell-array are sequentially read by the program to perform the analysis.

4.2 Outputs

4.2.1 data, model, estimation, misc

OpenBDLM_main returns four variables:

- `data` stores the time series data used for the analysis.
- `model` stores all the information about the model used for the analysis (current model structure and model parameters values)
- `estimation` stores the computed hidden states estimation using the current data and model.
- `misc` stores internal variables

```
Starting OpenBDLM_V???...
Starting a new project...
Building model...
Simulating data...
Plotting data...
Saving database (binary format) ...
Saving database (csv format) ...
Saving project...
Printing configuration file...
Saving database (binary format) ...
Saving project...
See you soon !
```

Listing 2: Output on MATLAB command window when running `run_DEMO.m`

4.2.2 DATA, CFG, PROJ and LOG files

OpenBDLM_main reads and/or create four types of files:

- Data file DATA_*.mat: MAT binary file that store the time series data. These files are located in the "data/mat" subfolder.
- Configuration file CFG_*.m : Matlab script used to initialize and export a project in human readable format. These files are located in the "config_files" subfolder.
- Project file PROJ_*.mat : MAT binary file that stores a full project for further analysis (basically a project file stores the structure data, model, estimation, and misc). These files are located in the "saved_projects" subfolder.
- Log file LOG_*.txt : Text file that records information about the analysis. These files are located in the "log_files" subfolder.

5 Data loading

5.1 Input data format

OpenBDLM supports two types of input data format depending whether the time-series are synchronized, or not. The time-series are synchronous if they all share the same timestamps. Conversely, time-series are asynchronous if they do not all share the same timestamps.

5.1.1 Input data formatting for asynchronous time series

Comma Separated Values files One Comma Separated Values (CSV) file should be provided for each time series. The file is a two columns file that

```
'name'              ,   '2000-01-01-22-00-00'
737422              ,   0.40
737423.5            ,   0.21
737424              ,   0.548
7374245.25          ,   NaN
7374246             ,   0.57
```

Listing 3: CSV file example

should be organized as shown in Listing 3 The first line of the file is the header. In the header, the first field should contain the label of the time-series given as a quoted delimited string, as 'name'. The second field is the date of the first timestamp given as a quoted delimited string, formatted as 'YYYY-DD-MM-HH-MM-SS'. For the remaining lines, the first field is the date given as a serial date number in number of days, given as a real number, and the second field is the magnitude of the physical quantity measured, given as a real number. The missing data should be indicated as NaN number. CSV files must be stored in the "OpenBDLM-master/data/csv" subfolder.

5.1.2 Input data formatting for synchronous time series

MATLAB .MAT files The MATLAB binary .MAT file must contain three MATLAB variables called `labels`, `timestamps`, and `values`.

- `labels` is $1 \times M$ cell array containing the reference name associated with each time-series, where M is the number of time series.
- `timestamps` is $N \times 1$ array containing the timestamps given as serial date number from January 0, 0000, where N is the number of data samples.
- `values` is $N \times M$ array containing the data amplitude values.

MATLAB binary .MAT files must be stored in the "OpenBDLM-master/data/mat" subfolder.

Comma Separated Values files See Paragraph 5.1.1 for CSV files formatting.

5.2 Output data format

Output data formatting is MATLAB binary .MAT file (see Paragraph 5.1.2) and CSV files (see Paragraph 5.1.1).

5.3 Data loading functions

- [data, misc, dataFilename]=DataLoader(misc) Create a data file

- [dataOrig, misc]=readMultipleCSVFiles(misc) Read data from multiple CSV files
- [dat, label]=readSingleCSVFile(FileToRead, varargin) Read a single CSV file
- [misc, dataFilename] = saveDataBinary(data, misc, varargin) Save data in a binary Matlab .mat file
- [misc] = saveDataCSV(data, misc, varargin) Save time series data in separate CSV files

6 Data editing and pre-processing

Data editing prepares the data for analysis. Data editing includes time series selection, data analysis time period selection, missing data removal, and data resampling. The time synchronization is performed automatically through the data editing process, when multiple time series are processed simultaneously.

```
— Choose from

    1  ->  Select time series
    2  ->  Select data analysis time period
    3  ->  Remove missing data
    4  ->  Resample
    5  ->  Change synchronization options

    6  ->  Reset changes
    7  ->  Save changes and continue analysis

    choice >>
```

Listing 4: OpenBDLM data editing menu

6.1 Selection of time-series

The selection of time-series allows to select only a subset of the time series in memory. The time series are automatically synchronized as time-series are added to (or removed from) the dataset.

6.2 Selection data analysis time period

The selection of the period of analysis allows to select only the data between two dates, given as 'YYYY-DD-MM' format. If the second requested date exceeds the date corresponding to the last timestamp of the original dataset, padding with NaN values are performed. The timestep for the NaN padding should be user's provided.

6.3 Removing missing data

It is possible to control the maximum amount of NaN missing data allowed at each time slice. By default, the maximum amount of missing data is 100%, given with respect to the total number of time-series.

6.4 Data resampling

Data resampling changes the sampling rate of the time-series according to a given timestep provided by the user. If the requested timestep is higher than the original data timestep, NaN values are added. Conversely, if the requested timestep is lower than the original data timestep, OpenBDLM averages the data amplitude values within non-overlapping fixed time windows, each having the duration of the requested timestep. The first time window starts at the first timestamp, and the new timestamps are assigned at the times corresponding to the middle of each time window.

6.5 Time synchronization options

By default, the time synchronization in OpenBDLM is done by adding NaN values. The time synchronization is controlled by the NaNThreshold and tolerance variables. NaNThreshold is given in percent with respect to the total number of time-series. The variable tolerance gives the duration (in number of days) after which two timestamps are not considered equal. The default values for NaNThreshold and tolerance are 100% and 10^{-6} days, respectively.

6.6 Data editing functions

- [data, misc]=chooseTimeSeries(data, misc) Request the user to select some time series
- [timesteps]=computeTimeSteps(timestamps) Compute timestep vector from timestamps vector
- displayData(data, misc) Display on screen the content of the data in memory
- [FileInfo] = displayDataBinary(misc, varargin) Display the list of DATA_*.mat files
- [data, misc, dataFilename]=editData(data, misc, varargin) Control script to edit dataset (selection, resampling, etc..)
- [data, misc]=mergeTimeStampVectors(dataOrig, misc, varargin) Create a single time vector from a set of time series
- [data_resample, misc]=resampleData(data, misc, varargin) Resample dataset according to a given timestep.
- [data, misc]=selectTimePeriod(data, misc) Select data between two dates

7 Model building

7.1 Purpose

7.2 Model class

7.3 Dependencies

7.3.1 Observed covariate

7.3.2 Hidden covariate

7.4 Model components

7.4.1 Local level

7.4.2 Local trend

7.4.3 Local acceleration

7.4.4 Local level compatible trend

7.4.5 Local level compatible acceleration

7.4.6 Local trend compatible acceleration

7.4.7 Periodic

7.4.8 Kernel regression

7.4.9 Residual - first order autoregressive

7.5 Model building functions

8 Model parameters learning

8.1 Purpose

8.2 Posterior

8.2.1 Prior

8.2.2 Likelihood

8.3 Model parameters bounds and transformed spaces

8.3.1 Logarithmic transformation

8.3.2 Sigmoid transformation

8.4 Gradient-based optimization techniques

8.4.1 Newton-Raphson approach

8.4.2 Stochastic Gradient Ascent approach

8.5 Constrain model parameters between each others

8.6 Model parameters learning functions

9 Hidden states estimation

9.1 Purpose

9.2 Kalman equations

9.3 UD computations

9.4 Filtering

9.5 Smoothing

9.6 Hidden states estimation functions

10 Data simulation

10.1 Purpose

10.2 Data simulation functions

11 Visualization tools

11.1 Purpose

11.2 Data availability plots

11.3 Hidden states plots

11.4 Export figures options

11.5 Visualization tools functions

12 Version control

12.1 Purpose

12.2 Version control functions

13 OpenBLDM options

14 Examples

14.1 Example 1

Simulated data with: one time series, one model class, {[12 31 41]}.

14.2 Example 2

Simulated data with: two time series with dependencies, one model class, {[11 41], [11 31 41]}.

14.3 Example 3

Simulated data with one time series, two model classes, {[21 31 41]} and {[12 31 41]}.

14.4 Example 4

Real data with one time series, one model class, {[12 51 41]}.

15 List of functions

16 Older versions

17 Last changes