Time-series forecast model notes

Version history

|  |  |  |  |
| --- | --- | --- | --- |
| **Date** | **Author** | **Revision** | **Change notes** |
| 03.12.2021 | TZ | 0 | Doc creation |
| 10.12.2021 | TZ | 0.1 | Content creation |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

Contents

[1 Introduction 3](#_Toc90050259)

[1.1 Model usage 3](#_Toc90050260)

[1.2 Required environment 3](#_Toc90050261)

[1.2.1 Python installation 3](#_Toc90050262)

[2 Functions 4](#_Toc90050263)

[2.1 run\_arima 4](#_Toc90050264)

[2.2 run\_extrap 4](#_Toc90050265)

[2.3 Example 4](#_Toc90050266)

[3 Issues and future changes 5](#_Toc90050267)

[3.1 Issues 5](#_Toc90050268)

[3.2 Future changes 5](#_Toc90050269)

# Introduction

This document describes the functionality of time-series-based degradation forecast models as part of project Dead Sea.

## Model usage

Time-series forecast models predict the future degradation trend statistically based on the past degradation data. The prediction is made simply from either autoregression of the past capacities, or linear/power-law fitting. Importantly, **these models cannot foresee future changes in the degradation trend** such as the ‘knee-point’ (aka ‘roll-off’ point), since they possess no knowledge of the general degradation behaviour of Li-ion cells or any mechanistic insight of degradation mechanisms. Thus these models can only be used for extrapolating degradation curves into the near future, but not for reliable long-term health prediction.

## Required environment

The model functions are called from Matlab, but they also require compatible Python and Python packages to be installed.

### Python installation

Python (64-bit) can be downloaded from: [Download Python | Python.org](https://www.python.org/downloads/). **Matlab 2021a supports up to Python version 3.8, whereas Matlab 2021b supports up to Python 3.9. Matlab won’t recognise the Anaconda distributions of Python.**

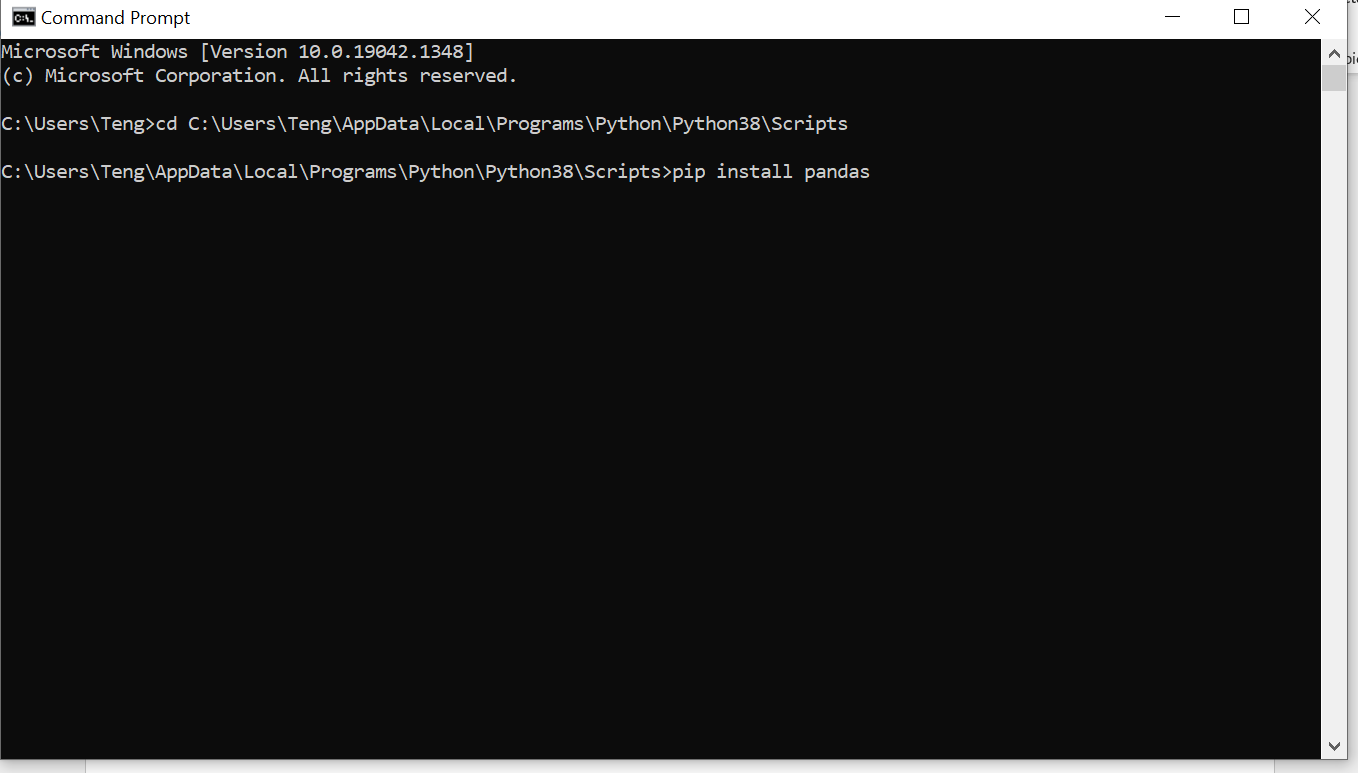
Once Python is installed, several commonly used modules can be installed using the pip command. First, launch command prompt (cmd) in Windows, and go to the path:

C:\Users\<yourusername>\AppData\Local\Programs\Python\Python38\Scripts

Then, simply type command of the following structure to install Python packages:

pip install package\_name

For example, to install pandas:



Install the following packages:

* numpy
* matplotlib
* statsmodels
* sklearn

Once Python and its packages are installed, launch Matlab and type: pynev. If installed correctly, Matlab should output a summary of linked Python version and path.

# Functions

## run\_arima

This function trains and tests an ARIMA model for degradation prediction. It calls several functions in the python script ‘arima\_functions.py’. The function trains ARIMA model using first 75% of capacity data, and test of the prediction using the latest 25% of data. ARIMA (p,d,q) models of several different orders (p – autoregressive order, d – differencing order, q – moving average order) are trained and tested, and the one with the smallest error against test data are chosen for future prediction.

prediction=run\_arima(run\_No,pred\_len). Returns predicted normalised capacities for Breathe run number run\_No, up to the next pred\_len number of cycles using ARIMA model.

prediction=run\_arima(run\_No,pred\_len,plot\_pred,plot\_test). Takes two optional plotting arguments. If plot\_pred is 1 (True), the function plots predicted future capacity trajectory. If plot\_test is 1 (True), the function plots predicted (past) capacity versus the test data. This plot gives an idea of the reliability of the model in predicting known data.

prediction=run\_arima(run\_No,pred\_len,plot\_pred,plot\_test,cell\_cap). Takes the optional argument to specify the nominal capacity of the cell. If cell\_cap is not specified, the function uses the capacity of the first/second cycle as the nominal capacity to normalise capacity.

[prediction,best\_order]=run\_arima(run\_No,pred\_len,plot\_pred,plot\_test,cell\_cap). Additionally returns the optimal orders identified for the ARIMA model. This maybe used in the future to troubleshoot poor predictions made by ARIMA.

## run\_extrap

This function finds to best linear and power-law trend to fit the existing degradation data and extrapolate into future cycles. The function only uses the latest user-specified number of cycles for fitting rather than using all past cycles.

[y,pred\_l,pred\_p]=run\_extrap(run\_No,fit\_len,pred\_len,plot\_pred,plot\_test,cell\_cap).

Outputs: y is the capacity data from Breathe run number run\_No. pred\_I is the predicted linear capacity trend up to the future pred\_len cycles. pred\_p is the predicted power-law capacity trend up to the future pred\_len cycles.

Inputs: fit\_len is number of cycles used for fitting. The function uses the capacity data from the last fit\_len number of cycles to derive linear and power-law fitting. You can specify its value to be ‘all’ to let the function fit all capacity data, but this usually results in poor prediction. The other input arguments are the same as those in run\_arima function.

## Example

Example of running both run\_arima and run\_extrap methods for degradation prediction is given in the script ‘make\_extrapolation.m’.

# Issues and future changes

## Issues

The ARIMA prediction can sometimes be very poor despite fitting existing data well. This could be due to poor starting point for prediction, for example, when the latest capacities prior to prediction are noisy or changing trend drastically. This can usually be fixed by changing the starting point for prediction (not yet implemented).

Another issue with the current ARIMA implementation is that the grid search for optimal ARIMA model order may not be rigorous. Although grid search will return the best fit model according to past data, occasionally such model may not be reasonable for future prediction.

## Future changes

To address the issues, future release will add:

* User defined input to change the starting point for prediction (away from the last cycle).
* User specified ARIMA model order. This will then bypass the grid search for optimal model order.
* User defined cycle range for training the optimal ARIMA model before making predictions.