AutoRegressive Integrate Moving Average (ARIMA) Model

– A Matlab Implementation

Fengji Luo, The University of Sydney

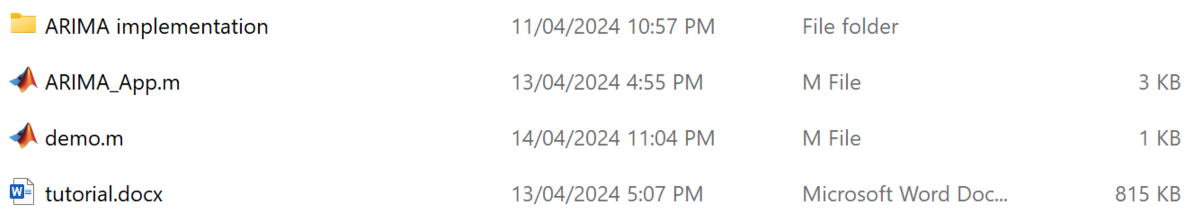
1 Introduction

The ARIMA model is a statistical analysis model for time series forecasting. A *time series* is a series of data points indexed in time order. Time series forecasting refers to as the prediction of future data values by analyzing the trends of the past, based on the assumption that the time series' future trends will be related to the historical trends. ARIMA model analyzes historical time series trends and make predictions by combing an auto-regressive model and a moving average model. Please refer to references [1, 2] for more details about the ARIMA model. This document is a tutorial of explaining how to use a Matlab implementation of the ARIMA model to do time series forecasting.

2 Package Installation and Overview

The Matlab implementation of the ARIMA model can be installed following two steps:

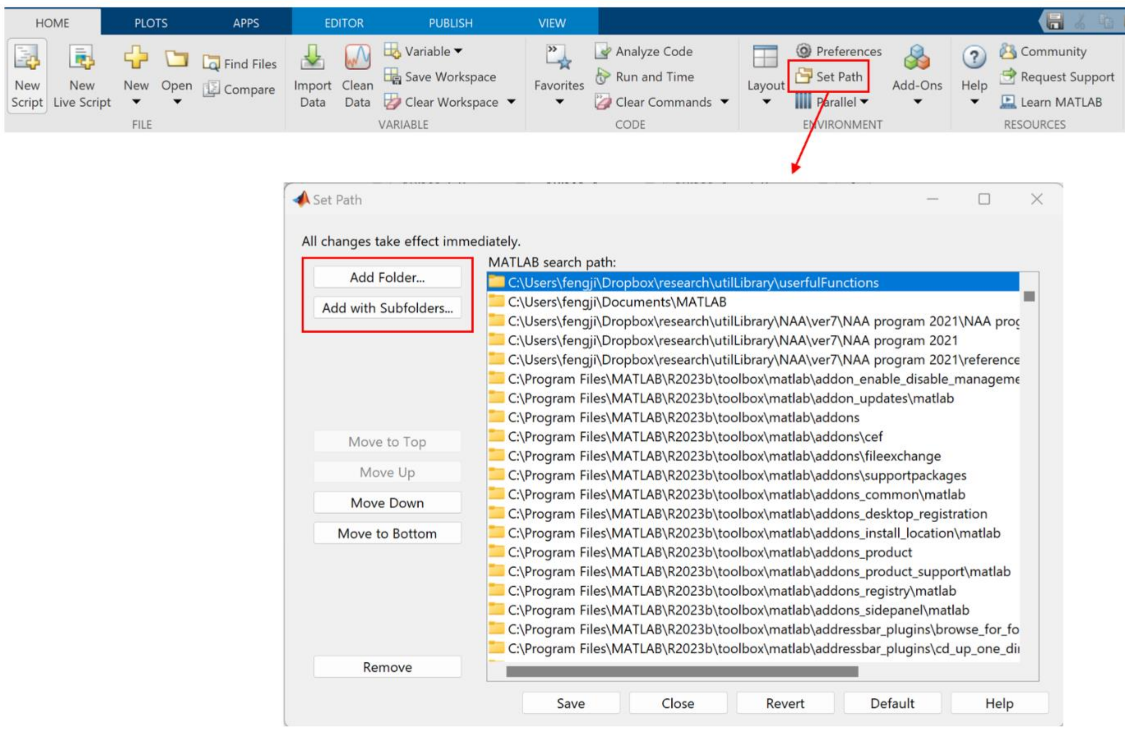
1. Copy or download the package to a computer. The content in the package root folder is as follows:



ARIMA implementation	11/04/2024 10:57 PM	File folder	
ARIMA_App.m	13/04/2024 4:55 PM	M File	3 KB
demo.m	14/04/2024 11:04 PM	M File	1 KB
tutorial.docx	13/04/2024 5:07 PM	Microsoft Word Doc...	815 KB

Figure 1 Content in the root folder of the Matlab-based ARIMA package

2. Add the package root folder and the "ARIMA implementation" folder into the Matlab path:



All changes take effect immediately.

MATLAB search path:

```
C:\Users\fengji\Dropbox\research\utilLibrary\userfulFunctions
C:\Users\fengji\Documents\MATLAB
C:\Users\fengji\Dropbox\research\utilLibrary\NAA\ver7\NAA program 2021\NAA proc
C:\Users\fengji\Dropbox\research\utilLibrary\NAA\ver7\NAA program 2021
C:\Users\fengji\Dropbox\research\utilLibrary\NAA\ver7\NAA program 2021\reference
C:\Program Files\MATLAB\R2023b\toolbox\matlab\addon_enable_disable_manageme
C:\Program Files\MATLAB\R2023b\toolbox\matlab\addon_updates\matlab
C:\Program Files\MATLAB\R2023b\toolbox\matlab\addons
C:\Program Files\MATLAB\R2023b\toolbox\matlab\addons\cef
C:\Program Files\MATLAB\R2023b\toolbox\matlab\addons\fileexchange
C:\Program Files\MATLAB\R2023b\toolbox\matlab\addons\supportpackages
C:\Program Files\MATLAB\R2023b\toolbox\matlab\addons_common\matlab
C:\Program Files\MATLAB\R2023b\toolbox\matlab\addons_desktop_registration
C:\Program Files\MATLAB\R2023b\toolbox\matlab\addons_install_location\matlab
C:\Program Files\MATLAB\R2023b\toolbox\matlab\addons_product
C:\Program Files\MATLAB\R2023b\toolbox\matlab\addons_product_support\matlab
C:\Program Files\MATLAB\R2023b\toolbox\matlab\addons_registry\matlab
C:\Program Files\MATLAB\R2023b\toolbox\matlab\addons_sidepanel\matlab
C:\Program Files\MATLAB\R2023b\toolbox\matlab\addressbar_plugins\browse_for_fo
C:\Program Files\MATLAB\R2023b\toolbox\matlab\addressbar_plugins\cd_up_one_di
```

Figure 2 Add the package into Matlab path

As shown in Figure 1, there are following files/folders in the package:

1. "ARIMA implementation" folder. This folder contains the underlying codes of the ARIMA model. Users of the package do not need to directly touch the files in this folder.

2. "ARIMA_App.m" file. It is a function for the user to use the ARIMA model to perform time series forecasting. This function will internally invoke the underlying codes in the "ARIMA implementation" folder. Note: the "_App" in the file name represent "application", meaning this is the interface for the user to use the ARIMA model to build up time series forecasting applications. The characters "_App" are add in the file to differentiate it from the underlying ARIMA codes in the "ARIMA implementation" folder.

3. "demo.m" file. It provides an example to demonstrate how to invoke the "ARIMA_App.m" function to do time series forecasting.

3 Use of the Package

The user invokes the "ARIMA_App" function to perform time series forecasting. This can be done following the steps below:

3. 1 Prepare the Input Time Series

The first step is to prepare the time series data that will be inputted into the model. An input time series should be represented as a column vector:

$$\text{inputTimeSeries} = [y_1, y_2, \ldots, y_T]$$

where T is the total number of data items in the time series; x_t (t=1:T) represents the data item in the tth time interval. For example, the following Matlab code defines an input time series consisting of 10 data items (i.e., T=10):

```
inputTimeSeries = [1.2, 2.4, 3.0, 4.8, 3.3, 2.7, 2.5, 2.4, 3.2, 5.0];
```

Code Example 1 Define an input time series

The input time series can also be loaded from external files, such as excel files.

Note: the input time series may not be the one to which the ARIMA model is eventually applied, depending on whether differencing operations will be performed for the input time series (see Section 3.4). If no differencing operation performed, then the input time series will be passed to the ARIMA model (which is implemented in the "ARIMA.m" file in the "ARIMA implementation" folder; otherwise, the differenced time series, which is generated from the input time series, will be passed to "ARIMA.m" file, and the difference time series will be the actual time series to which the ARIMA model applies.

3.2 Specify Forecasting Window

It is considered that the input time series contains the data items historically observed, for any time interval t>T, it is considered as a future time point when the data item not available. Based on the input time series, the user needs to specify a time range within which the user wants the ARIMA model to generate forecasts. This time range is called the *forecasting window* and can be represented as [fw_start, fw_end], where "fw_start" and "fw_end" are two positive integers specifying the boundaries of the forecasting window. The data value at every time interval fw_start $\leq t \leq$ fw_end will be forecasted by the ARIMA model.

It is noticeable that the forecasting can be made for a future time interval (i.e., t>T) or a time interval in the past (i.e., $t \leq T$). For the latter case, the model will generate a data value that is considered as a fitted value of the actual data value. The following example sets a forecasting window [11, 13] subjected to the time series defined in Code Example 1, in which all the 3 time intervals are future time intervals.

```
fw_start = 11;
fw_end = 13;
```

Code Example 2 Specify a forecasting window

Code example 3 specifies another forecasting window, which covers all the 10 historical time intervals in the input time series. This means to ask the ARIMA model to generate a fitted value for each historical time interval.

```
fw_start = 1;
fw_end = 10;
```

Code Example 3 Specify another forecasting window

Specifically, when "fw_start" and "fw_end" are set to be a same value, this means to ask the ARIMA model to perform forecasting for a single time interval.

3.3 Specify Model Parameters

The ARIMA model is associated with multiple parameters, and their values need to be pre-specified before using the model. The parameters include:

(1) "c" – This parameter is a positive real number or a character string "default", representing the level of forecasting. Without any special prior knowledge, it can be set to be "default" – with this setting, c will be equal to the average value across the data items in the time series that is eventually inputted into the ARIMA function (in the "ARIMA implementation" folder).

(2) "p" – It is a positive integer, representing the order of the autoregressive sub-model, i.e., the number of the most recent data values of the forecasting time interval t, from which the forecasting will be generated for time interval t.

(3) "q" – It is a positive integer, representing the order of the moving average sub-model, i.e., the number of the most recent data values of the target forecasting time interval t; from the forecasting errors of these data values, the forecasting will be generated for time interval t.

(4) Parameters $\phi_1, \ldots, \phi_p$. In the package, these parameters are put in a column vector called "faiArray".

(5) Parameters $\theta_1, \ldots, \theta_q$. In the package, these parameters are put in a column vector called "thetaArray".

To use the package, the above parameters need to be put into a Matlab structure array with five fields: "c", "p", "q", "faiArray" and "thetaArray". The following example shows how to set up and package the model parameters:

```
params.c = 'default';
params.p = 2;
params.q = 2;
params.faiArray = [-0.8, -0.2];
params.thetaArray = [0.4, -0.3];
```

Code Example 4 Specify model parameters

Selection of model parameters

When p=1 or 2, the following empirical rules are usually applied to the selection of the parameter ϕ_1 and ϕ_2: (i) when p=1, $-1 < \phi_1 < 1$; (ii) when p=2, $-1 < \phi_2 < 1$, $\phi_1 + \phi_2 < 1$, and $\phi_2 - \phi_1 < 1$.

When q=1 or 2, the following empirical rules are usually applied to the selection of the parameter θ_1 and θ_2: (i) when p=1, $-1 < \theta_1 < 1$; (ii) when p=2, $-1 < \theta_2 < 1$, $\theta_1 + \theta_2 > -1$, and $\theta_1 - \theta_2 < 1$.

For the cases of $p \geq 3$ and/or $q \geq 3$, the ranges to be used for the parameter values become more complex, and futher information on these can be found in the dedicated references, e.g. [2].

3.4 Specify the Order of Differencing Operations

The ARIMA model should be applied to *stationary time series*, which refers to as time series whose statistical properties (such as the mean and deviation) do not depend on the time at which the series is observed. For example, the white noise is a typical stationary time series. The input time series could not be stationary, as it could contain meaningful variation trends and patterns. In this case, to apply the ARIMA model, differencing operation(s) need to be performed to generate a stationary time series from the non-stationary input time series.

Denote the input time series $\boldsymbol{y}$ as $\boldsymbol{y} = [y_1, y_2, \dots, y_T]$, for the first order differencing, a differenced time series $\boldsymbol{y}^{(1)}$ is generated from $\boldsymbol{y}$:

$$\boldsymbol{y}^{(1)} = [y_2^{(1)}, y_3^{(1)}, \dots, y_T^{(1)}]$$

$$y_t^{(1)} = y_t - y_{t-1}, \forall t = 2{:}T$$

The first order differenced time series describes the changes of the input time series. Obviously, the first item in the first order differenced time series is $y_2^{(1)}$, calculated from y_2 and y_1. If there are T data values in the input time series, then there are T-1 data values in the first order difference time series. Similarly, the second order differencing operation does the same thing to $\boldsymbol{y}^{(1)}$ to generate a further differenced time series $\boldsymbol{y}^{(2)}$:

$$\boldsymbol{y}^{(2)} = [y_3^{(2)}, y_4^{(2)}, \dots, y_T^{(2)}]$$

$$y_t^{(2)} = y_t^{(1)} - y_{t-1}^{(1)}, \forall t = 3{:}T$$

The second order differenced time series describes the “change of the change” of the input time series, and it contains T-2 data values.

For most real-world time series, a first order differencing operation is sufficient to generate a stationary time series $\boldsymbol{y}^{(1)}$. In few cases, it needs to perform the differencing operation twice to generate a stationary time series $\boldsymbol{y}^{(2)}$. It is very uncommon that third or even higher order of differencing operations need to be performed to generate a stationary time series.

In the package, the user can define an integer larger than or equal to zero to specify the order of differencing operations he/she wants to perform on the input time series. Code Example 5 provides an example of specifying a first order differencing operation:

```
diffOrder = 1;
```

Code Example 5 Specify the order of differencing operations

The differenced time series will be eventually passed to the “ARIMA” function, to which the ARIMA model will be applied. When the order of differencing is set to be zero, it means no differencing operations performed and the input time series will be directly passed to the “ARIMA” function.

3.5 Performing Time Series Forecasting

After setting up the relevant information following the above steps, the user can invoke the “ARIMA_APP” function to perform time series forecasting for the specified forecasting window. The function takes the inputs introduced in Sections 3.1-3.4 (i.e., inputTimeSeries, fw_start, fw_end, params, diffOrder) and another input "verbose". The "verbose" input is a binary value: when it is set to be 1, the package will print some intermediate forecasting progress information in the Matlab console; when it is set to be 0, no progress information will be outputted.

The function generates 4 outputs:

- "predictionArray": a column vector, in which each item corresponds to the forecasted data value at a time interval within the forecasting window.
- "errorArray": a column vector, in which each item represents the forecasting error at a time interval within the forecasting window. For a forecasting time interval *t*, the forecasting error e_t is calculated as:
$$e_t = \begin{cases} 0, & if\ t > T \\ y_t - \hat{y}_t, & if\ t \le T \end{cases}$$
where $\hat{y}_t$ represents the fitted data value generate by the ARIMA model for the time interval *t*.
- "MSE": a real number, representing the Mean Squared Error (MSE) of the forecasting values over the forecasting window. MSE is calculated as follows, where *N*= fw_end - fw_start + 1 is the number of forecasting values:
$$MSE = \frac{\sum_{n=1}^{N} e_n^2}{N}$$
- "RMSE": a real number, representing the Root Mean Squared Error (RMSE). It is calculated as the square root of MSE.

The MSE and RMSE values indicate the forecasting accuracy. The smaller the MSE or RMSE is, the higher accuracy. The following code example shows a full procedure to use the package. It combines Code Examples 1, 2, 4 and 5, and then invokes the "ARIMA_APP" function to perform forecasting for 3 future time intervals, i.e., time intervals 11-13.

```
inputTimeSeries = [1.2, 2.4, 3.0, 4.8, 3.3, 2.7, 2.5, 2.4, 3.2, 5.0];
forecast_t_start = 11;
forecast_t_end = 13;
 params.c = 'default';
 params.p = 2;
 params.q = 2;
 params.faiArray = [-0.8, -0.2];
 params.thetaArray = [0.4, -0.3];
 diffOrder = 1;
 [predictionArray, errorArray, MSE, RMSE] = ARIMA_APP(inputTimeSeries, forecast_t_start,
forecast_t_end, params, diffOrder, 1);
```

Code Example 6 An example of the full procedure of using the package to forecast future data values

By running the above codes, the package will generate the following outputs: (1) predictionArray = [4.58, -0.84, 0.20]; (2) errorArray = [0, 0, 0]; (3) MSE = 0; and (4) RMSE = 0. The forecasting errors, the MSE and the RMSE are all zero, because in this application, the forecasting time intervals are in the future when the actual observations are unavailable.

Code Example 7 below shows how to use the package to fit all the historical data value in the input time series. All the codes are same with those in Code Example 6, except for the forecasting window. In Code Example 7, the forecasting window is set to be [1, 10], coving all the historical time intervals. In the last part of Code Example 7, a Matlab figure is created, in which both the actual data values in the input time series and the fitted data values generated by the ARIMA model are plotted. This can help visually evaluate how close between the actual and fitted data. The codes in Code Example 7 are included in the "demo.m" file.

```
inputTimeSeries = [1.2, 2.4, 3.0, 4.8, 3.3, 2.7, 2.5, 2.4, 3.2, 5.0];
forecast_t_start = 11;
forecast_t_end = 13;
params.c = 'default';
params.p = 3;
params.q = 2;
params.faiArray = [-0.8, -0.1, -0.1];
params.thetaArray = [0.2, -0.3];
diffOrder = 1;
[predictionArray, errorArray, MSE, RMSE] = ARIMA_APP(inputTimeSeries, forecast_t_start,
forecast_t_end, params, diffOrder, 1);

t=1:1:10;
plot(t, inputTimeSeries, t, predictionArray);
xlabel('Time interval index');
ylabel('Data value');
legend('Actual data', 'Fitted data')'
```

Code Example 7 An example of the full procedure of using the package to fit historical data values

By running the above codes, the following will be generated, and the outputs of the function will be:

(1) predictionArray = [1.20, 1.62, 5.45, 4.19, 5.89, 2.85, 2.54, 2.75, 2.95, 3.89];

(2) errorArray = [0, 0.78, -2.45, 0.62, -2.59, -0.15, -0.04, -0.35, 0.25, 1.11];

(3) MSE = 1.51.

(4) RMSE = 1.23.

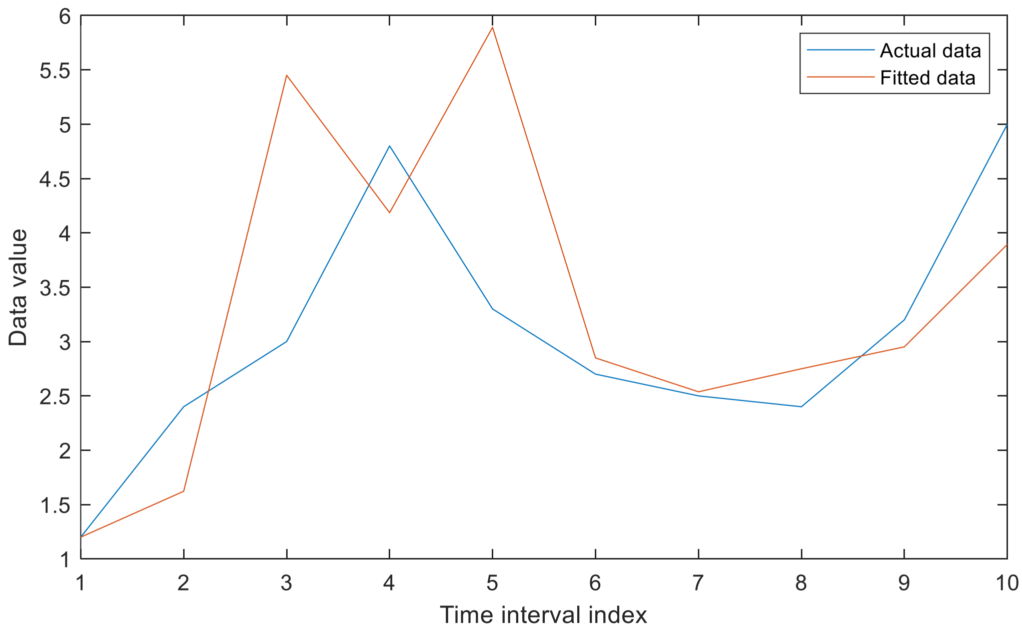


Figure 3 Plotting of the historical data fitting result

References

[1] F. Luo, G. Ranzi, and Z.Y. Dong, *Building Energy Management Systems and Techniques*, Elsevier, 2024.

[2] G. Athanasopoulos and R.J. Hyndman, *Forecasting: Principles and Practice*, 3rd edition, OTexts, 2021.