

SKOLKOVO INSTITUTE OF SCIENCE AND  
TECHNOLOGY

MA06018 (TERM 3, 2018-2019)

---

Online ARIMA

---

*Submitted By :*

Bespalov Iaroslav

Vyacheslav Kozitsin

Iurii Katser

Edgar Makarov

## Contents

Introduction	2
Related Work	3
Dataset Description and preprocessing	4
ML Methods	6
Experiments	7
Conclusion	10
Appendix	12

## Introduction

The ubiquity of time series is a fact of modern-day life: from stock market data to social media information; from search-engine metadata to webpage analytics, modern-day data exists as a continuous flow of information indexed by timestamps.

ARIMA models are the main time-series long-term forecasting models in many practical applications. In time series trends or dependencies usually change. This requires frequent refitting of the model from scratch or performing additional training. However, fitting ARIMA models to large-scale data (e.g. hourly time series for 10 years) is time consuming. In this case, the online learning approach helps.

For a real-time application, online modeling is desirable — a self-generating and self-updating forecasting method is desirable for producing a timely forecast. The ARIMA method in time series is an effective means to address a forecast; it has a solid foundation in classical probability theory and mathematical statistics [1].

We propose a comparison of different methods of seasonal ARIMA online implementation for two separated datasets. Therefore, the objective of this study was to investigate time complexity and performance of different online learning approaches:

- complete refitting on data of steps  $[1, t]$  each step  $t$ ;
- rolling-window refitting on  $[t-h, t]$ ;
- additional fitting (warm start) for every new observation;
- online SARIMA based on gradient descent.

## Related Work

In literature, very few study has seriously investigated scalable algorithms for ARIMA models, although some simple methods were attempted. For example, the iterated least- squares approach was developed to consistently estimate autoregressive parameters [2]. Least squares and gradient algorithms are presented through estimating residuals, for which a convergence analysis is also given by using the martingale convergence theorem [3]. Nevertheless, none of these has been formally formulated in a standard online learning setting [4].

But in recent times there appeared several works [4–7] on developing online ARIMA algorithm for different applications.

## Dataset Description and preprocessing

The first Dataset is Atmospheric  $CO_2$  Data [8]. After dropping NaNs and splitting Dataset for train and test we get following:

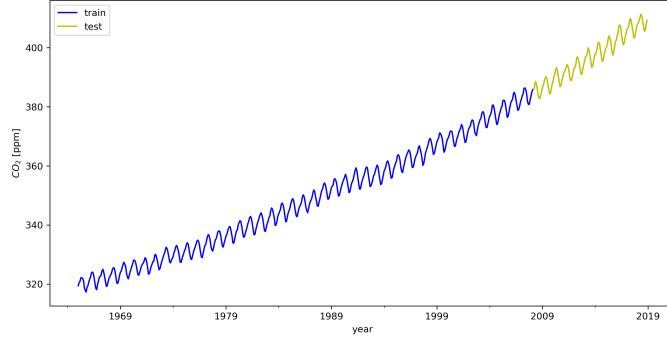


Figure 1 – Atmospheric  $CO_2$  Dataset

Further we start analyzing of our Dataset:

- Estimation spectral density using Welch's method (figure 2):

The Welch method is based on Bartlett's method and differs in two ways: the signal is split up into overlapping segments: the original data segment is split up into  $L$  data segments of length  $M$ , overlapping by  $D$  points. If  $D = M/2$ , the overlap is said to be 50%. The overlapping segments are then windowed: After the data is split up into overlapping segments, the individual  $L$  data segments have a window applied to them (in the time domain). The windowing of the segments is what makes the Welch method a "modified" periodogram.

After doing the above, the periodogram is calculated by computing the discrete Fourier transform, and then computing the squared magnitude of the result. The individual periodograms are then averaged, which reduces the variance of the individual power measurements.

- Selection of the period for seasonal differences to delete seasonal component using SARIMA.
- Providing Augmented Dickey–Fuller (ADF) test:

The Dickey–Fuller test tests the null hypothesis that a unit root is present in an autoregressive model:

$$\Delta y_t = (\rho - 1)y_{t-1} + u_t = \delta y_{t-1} + u_t,$$

testing for a unit root is equivalent to testing  $\delta = 0$ . The testing procedure for the ADF test is the same as for the Dickey–Fuller test but it is applied to the model:  $\Delta y_t = \alpha + \beta t + \gamma y_{t-1} + \delta_1 \Delta y_{t-1} + \dots + \delta_{p-1} \Delta y_{t-p+1} + \varepsilon_t$ .

- Selection of the period for first differences to delete trend component.
- Providing Augmented Dickey–Fuller (ADF) test again to approve non-stationarity of the received Dataset.
- Searching for the hyperparameters  $(p,d,q)$ ,  $(P,D,Q,t)$  of the model SARIMAX by fitting a number of models and selecting the best depending on the AIC.

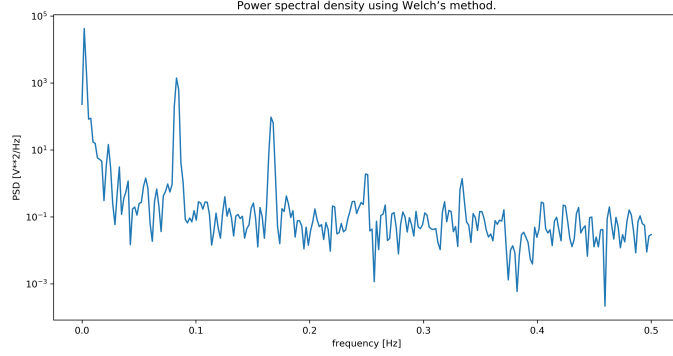


Figure 2 – Spectral density estimation using Welch's method

The second Dataset is International airline passengers: monthly totals in thousands [9]. After dropping NaNs and splitting Dataset for train and test second dataset looks like at the figure 3. Then we provide all the procedures described earlier for the first Dataset.

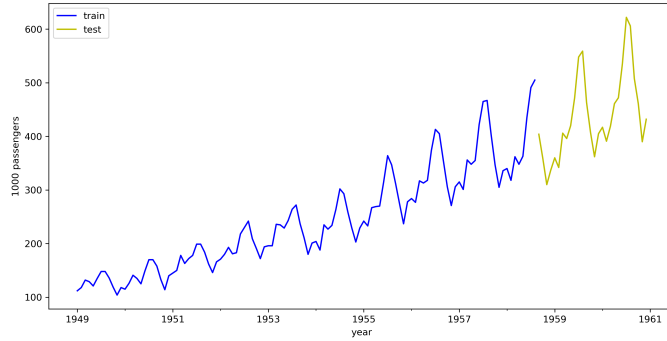


Figure 3 – International airline passengers: monthly totals in thousands

## ML Methods

Time series data are usually not realizations of a stationary process. For example, some of them may contain deterministic trends. An effective way to handle such strong serial correlations is to consider the differential method. For example, one can compute the first order differences of  $X_t$  by  $\nabla X_t = X_t - X_{t-1}$  and the second order differences of  $X_t$  by  $\nabla^2 X_t = \nabla X_t - \nabla X_{t-1}$ .

If the sequence of  $\nabla^d X_t$  satisfies an ARMA(k, q), we say that the sequence of  $X_t$  satisfies the ARIMA(k, d, q) (short for AutoRegressive Integrated Moving Average):

$$\nabla^d X_t = \sum_{i=1}^q \beta_i \epsilon_{t-i} + \sum_{i=1}^k \alpha_i \nabla^d X_{t-i} + \epsilon_t,$$

which are parameterized by three terms k, d, q and weights vector  $\alpha \in \mathbb{R}^k$  and  $\beta \in \mathbb{R}^q$ . Note that ARMA(k,q) is a special case of the ARIMA(k, d, q), where the differences order is zero. Forecasting with ARIMA(k,d,q) is a reversion of differential process. Suppose time series sequence  $X_t$  satisfies ARIMA(k, d, q), we can predict the d-th order differential of observation at time  $t + 1$  as  $\tilde{X}_t$ :

$$\tilde{X}_t = \nabla^d \tilde{X}_t + \sum_{i=0}^{d-1} \nabla^i X_{t-1}.$$

Consider an online ARIMA iteration at time  $t$ , the learner makes a prediction  $\tilde{X}_t$  and then the true  $X_t$  is disclosed to the learner. As a result, the learner suffers a loss  $\ell_t(X_t, \tilde{X}_t)$ . More formally, we can define the loss function as follows:

$$\begin{aligned} f_t(\alpha, \beta) &= \ell_t(X_t, \tilde{X}_t(\alpha, \beta)) = \ell_t\left(X_t, \left(\nabla^d \tilde{X}_t + \sum_{i=0}^{d-1} \nabla^i X_{t-1}\right)\right) \\ &= \ell_t\left(X_t, \left(\sum_{i=1}^q \beta_i \epsilon_{t-i} + \sum_{i=1}^k \alpha_i \nabla^d X_{t-i} + \sum_{i=0}^{d-1} \nabla^i X_{t-1}\right)\right) \end{aligned}$$

The goal of online ARIMA learning is to minimize the sum of losses over some number of rounds  $T$ . More formally, we can define the regret of the learner after  $T$  rounds as:

$$R_T = \sum_{t=1}^T \ell_t(X_t, \tilde{X}_t) - \min_{\alpha, \beta} \sum_{t=1}^T \ell_t(X_t, \tilde{X}_t(\alpha, \beta))$$

## Experiments

In our work we put into practice four different methods of online ARIMA implementation. The first three methods use built-in SARIMAX functions for fitting and forecasting from Statsmodel library for python 3.\*. The fourth one is an algorithm from [4] based on gradient descent. We improved this algorithm by adding opportunity of using seasonal differences to delete seasonal component from the data.

### Method 1. Complete refitting

This method produces complete refitting at each step  $t$  of our model on the data of steps  $[1, t]$ .

### Method 2. Rolling-window refitting

This method produces rolling-window refitting at each step  $t$  on the data of steps  $[t - h, t]$ .

### Method 3. Warm start

This method uses built-in functions of statsmodel.SARIMAX by using params of the previously fitted model to speed up convergence of online fitting of the model at each step  $t$ .

### Method 4. Gradient descent. Real online

We now implement a more general online convex optimization solver, Online Gradient Descent, which is applicable to any convex loss functions. Figure 4 presents the proposed ARIMA-OGD algorithm for optimizing the coefficient vector using the OGD algorithm. The projection  $\Pi_{\mathcal{K}}(y)$  refers to the Euclidean projection onto  $\mathcal{K}$ , i.e.,  $\Pi_{\mathcal{K}}(y) = \arg \min_{x \in \mathcal{K}} \|y - x\|_2$ .

**Input:** parameter  $k, d, q$ ; learning rate  $\eta$ .  
**Set**  $m = \log_{\lambda_{max}}((TLM_{max}q)^{-1})$ .  
**for**  $t = 1$  **to**  $T - 1$  **do**  
    **predict**  $\tilde{X}_t(\gamma^t) = \sum_{i=1}^{k+m} \gamma_i \nabla^d X_{t-i} + \sum_{i=0}^{d-1} \nabla^i X_{t-1}$ ;  
    **receive**  $X_t$  **and incur loss**  $\ell_t^m(\gamma^t)$ ;  
    **Let**  $\nabla_t = \nabla \ell_t^m(\gamma^t)$ ;  
    **Set**  $\gamma^{t+1} \leftarrow \Pi_{\mathcal{K}}(\gamma^t - \frac{1}{\eta} \nabla_t)$ ;  
**end for**

Figure 4 – ARIMA-OGD algorithm

Four metrics were used to compare quality of the algorithms:

$$\text{MAE} = \frac{1}{N} \sum_{i=1}^N |y_i - \hat{y}_i|,$$

$$\text{MAPE} = \frac{100\%}{N} \sum_{i=1}^N \left| \frac{y_i - \hat{y}_i}{y_i} \right|,$$



$$\text{RMSE} = \frac{1}{N} \sqrt{\sum_{i=1}^N (y_i - \hat{y}_i)^2},$$

$$\text{RMSPE} = \frac{100\%}{N} \sqrt{\sum_{i=1}^N \frac{(y_i - \hat{y}_i)^2}{y_i^2}},$$

where  $N$  — number of samples to compare,  $y_i$  - true (real) value,  $\hat{y}_i$  — predicted value.

Two experiments to compare quality of the mentioned above online algorithms for each of the datasets were carried out (here and further results for the first dataset presented):

1. Short term forecast (fig. 6 in Appendix). Algorithms were fitted on the train set, during the test stage forecast for one point was obtained, then forecasted value was compared to real (test) value and, finally, real (test) point was added to train set. In case of rolling-window it was moved forward on one point.

Quality was compared using chosen metrics on each step (fig. 5a), in other words, one point forward forecast on each step of test set was compared to the real test value ( $N = 1$  in metrics formulas on each step).

2. Long term forecast (fig. 7 in Appendix). Algorithms were fitted on the part of the train set of length  $N_{\text{train}} - 100$ , then online algorithms were fitted 100 times till the end of the train set and the forecast was obtained on the last step of fitting.

Quality was compared using chosen metrics (fig. 5b) on each step of long term forecast including all previous steps.

Computational time results are represented in Table 1, while some errors results are represented in Table 2 and on Figure 5.

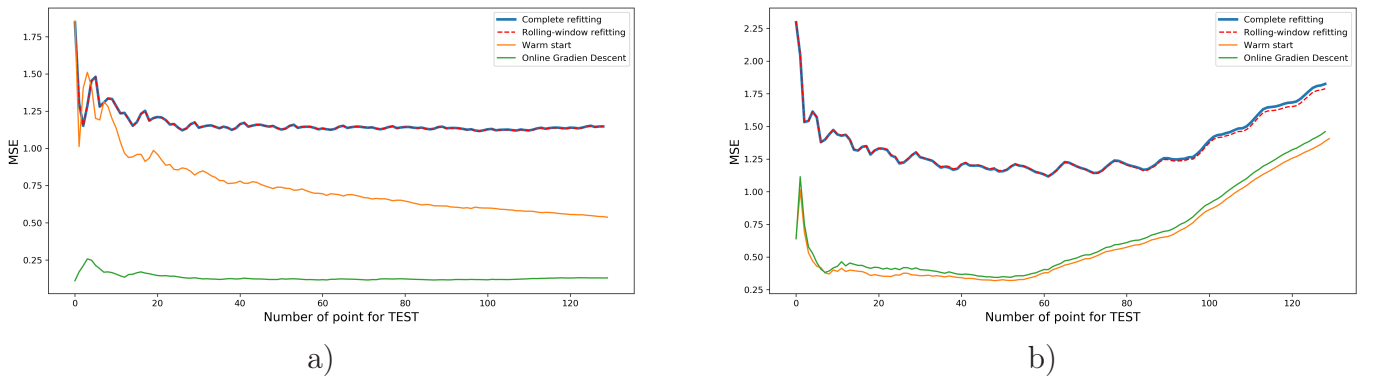


Figure 5 – Comparison of four online ARIMA algorithms using MSE: a) Short term (one point forward) forecast, b) Long term (130 points forward) forecast.

Table 1 – Computational time in msec/sample for the first dataset.

Method	short-term		long-term	
	fit	predict	fit	predict
Complete refit	180.9	14.3	134	13.2
Rolling-window	130.9	12.4	127.9	10.7
Warm start	2.15	3.61	0.094	0.197
Online GD	0.0074	0.0065	0.0079	0.0086

Table 2 – Different types of errors for the first dataset.

Method	Type of forecast	RMSE	MAE	MAPE	RMSPE
1	short	1.27	1.15	0.29	0.32
1	long	2.30	1.82	0.46	0.58
2	short	1.26	1.15	0.29	0.32
2	long	2.25	1.79	0.45	0.56
3	short	0.72	0.54	0.14	0.18
3	long	1.88	1.41	0.35	0.46
4	short	0.17	0.13	0.03	0.04
4	long	1.95	1.46	0.36	0.48

## Conclusion

We empirically compared our four online ARIMA algorithms, in which the promising results on two different datasets validate that all of the algorithms are effective and promising for time series prediction. Online-GD ARIMA algorithm shows the fastest computational time in both experiments for both datasets and lowest errors in short term forecasting while in long term forecasting warm-start algorithm has the smallest errors, but OGD algorithm shows slightly worse results which are still significantly better than others. Online-Gd ARIMA is the best way to implement online time series forecasting for datasets with high frequency or, in other words, computationally demanding problem.

Further research of this topic will include practical task analysis and improvements of implemented algorithms attempts for online: time reduction without reducing forecasting accuracy.

## Bibliography

- [1] Huitian Lu, W.J. Kolarik, and S.S. Lu. Real-time performance reliability prediction. *IEEE Transactions on Reliability*, 50(4):353–357, 2001.
- [2] Ruey S. Tsay and George C. Tiao. Consistent estimates of autoregressive parameters and extended sample autocorrelation function for stationary and nonstationary ARMA models. *Journal of the American Statistical Association*, 79(385):84–96, mar 1984.
- [3] Feng Ding, Yang Shi, and Tongwen Chen. Performance analysis of estimation algorithms of nonstationary ARMA processes. *IEEE Transactions on Signal Processing*, 54(3):1041–1053, mar 2006.
- [4] P. Zhao C. Liu, S. C. H. Hoi and J. Sun. Online arima algorithms for time series prediction. In *Proceedings of AAAI Conference on Artificial Intelligence*, ser. AAAI’16, page pp. 1867–1873. AAAI Press, 2016.
- [5] Qiuwen Chen, Tiesheng Guan, Liu Yun, Ruonan Li, and Friedrich Recknagel. Online forecasting chlorophyll a concentrations by an auto-regressive integrated moving average model: Feasibilities and potentials. *Harmful Algae*, 43:58–65, mar 2015.
- [6] Johann Leithon, Teng Joon Lim, and Sumei Sun. Renewable energy management in cellular networks: An online strategy based on ARIMA forecasting and a markov chain model. In *2016 IEEE Wireless Communications and Networking Conference*. IEEE, apr 2016.
- [7] Florian Schmidt, Florian Suri-Payer, Anton Gulenko, Marcel Wallschlager, Alexander Acker, and Odej Kao. Unsupervised anomaly event detection for cloud monitoring using online arima. In *2018 IEEE/ACM International Conference on Utility and Cloud Computing Companion (UCC Companion)*. IEEE, dec 2018.
- [8] Atmospheric data: [http : //scrippsco2.ucsd.edu/data/atmospheric\\_co2/primary\\_mlo\\_co2\\_record](http://scrippsco2.ucsd.edu/data/atmospheric_co2/primary_mlo_co2_record).
- [9] International airline passengers: monthly totals in thousands: <https://www.kaggle.com/andreazzini/international-airline-passengers/version/1>.

## Appendix

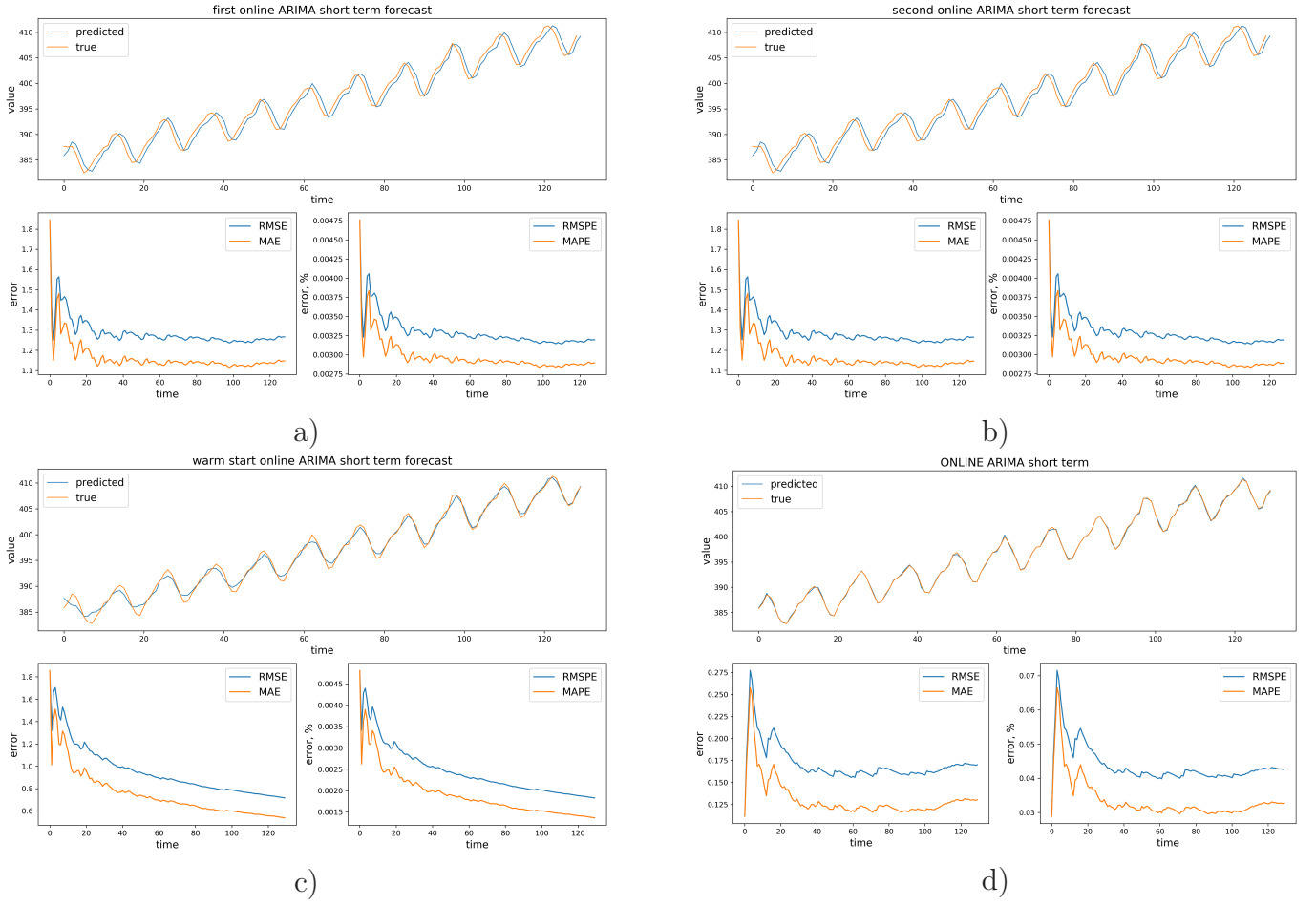


Figure 6 – Example of short term (one point forward) forecast and errors: a) Complete refitting algorithm, b) Rolling-window refitting algorithm, c) Warm start algorithm, d) Gradient descent. Real online algorithm.

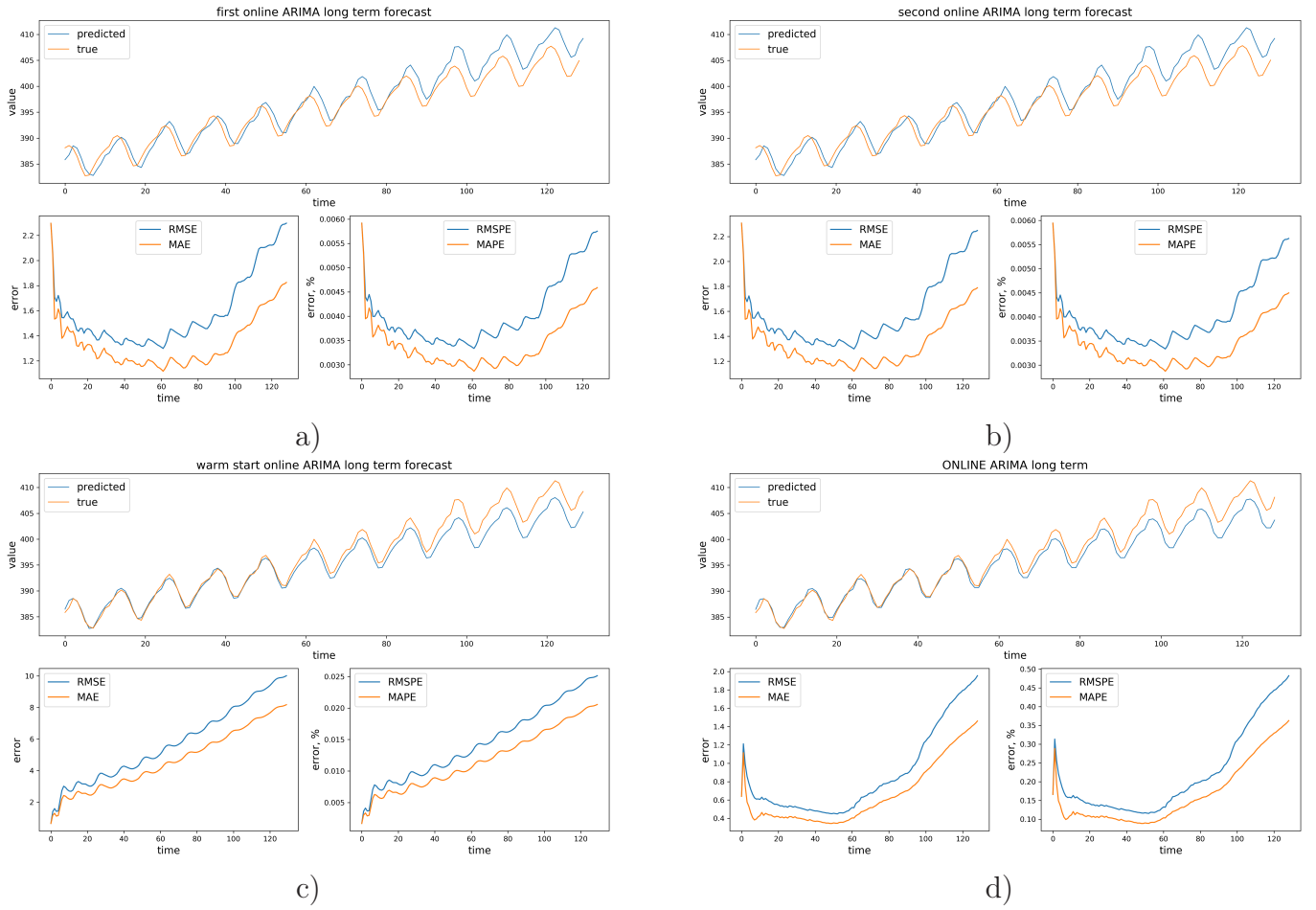


Figure 7 – Example of long term (130 point forward - length of test set) forecast and errors: a) Complete refitting algorithm, b) Rolling-window refitting algorithm, c) Warm start algorithm, d) Gradient descent. Real online algorithm.