

IIT. Madras



Fractional differentiation and its use in machine learning

Rafał Walasek¹ D · Janusz Gajda¹

Accepted: 1 August 2021/Published online: 25 August 2021 © The Author(s) 2021

Abstract This article covers the implementation of fractional (non-integer order) differentiation on real data of four datasets based on stock prices of main international stock indexes: WIG 20, S&P 500, DAX and Nikkei 225. This concept has been proposed by Lopez de Prado [5] to find the most appropriate balance between zero differentiation and fully differentiated time series. The aim is making time series stationary while keeping its memory and predictive power. In addition, this paper compares fractional and classical differentiation in terms of the effectiveness of artificial neural networks. Root mean square error (RMSE) and mean absolute error (MAE) are employed in this comparison. Our investigations have determined the conclusion that fractional differentiation plays an important role and leads to more accurate predictions in case of ANN.

Keywords Fractional differentiation · Financial time series · Stock market · Artificial neural networks · Machine learning

Mathematics Subject Classification 62M10 · 91B84

- □ Rafał Walasek rwalasek@wne.uw.edu.pl
- Faculty of Economic Sciences, University of Warsaw, 44/40 Dluga St, 00-241 Warsaw, Poland



1 Introduction

Many supervised machine learning algorithms are applied on financial time series data require stationarity [1]. In the process of creating a predictive model, it is assumed that a given time series is generated by a stochastic process. To make an accurate inference from such model, it is crucial that mentioned data generation process remains consistent. In statistical terms, the mean, variance and covariance should not change with time [1]. As a result in analyzed time series, a trend over time is not revealed. When this assumption is not fulfilled, the algorithm may assign a wrong prediction to a new observation.

In contrary to cross-sectional data, time series is a specific kind of data in the sense that any observation reflects a history of observations occurred in the past. The literature named it as a distinctive memory of past track record [2]. Due to the stationarity condition, this series memory is often excluded from the dataset.

The most commonly used method for non-stationarity removal is differencing up to some integer order [2]. Subtracting from each observation its predecessor, one gets the first-order differentiation. The second-order differencing is accomplished by repeating this process on obtained time series. It is similar for higher orders. Admittedly, these transformations can lead to the stationarity, but as a consequence all memory from the original series is erased [2]. On the other hand, the predictive power of machine learning algorithm is based on this memory. Lopez de Prado [3] calls it as the stationarity versus memory dilemma by asking a question whether there is a trade-off between these two concepts, in other words—does exist a solution of making the time series stationary concurrently with keeping its predictive power. One way to resolve this dilemma—fractional differentiation—has been proposed

by Hosking [4]. Lopez de Prado upgraded this idea to find the optimal balance between zero differentiation and fully differentiated time series.

The remainder of the paper is organized as follows. The next section gives an overview of fractional differentiation introduced above. The data used in this research are presented in Sect. 3, and the method applied on this data is described in Sect. 4. Section 5 discusses the results, and chapter 6 concludes the paper.

2 Fractional differentiation—an overview

In this section, the concept of fractional differentiation is elaborated in more detail. Assume that a time series X runs throughout time t and it is not stationary:

$$X = \{X_t, X_{t-1}, X_{t-2}, \dots, X_{t-k}, \dots\}.$$

As previously explained, computing the differences between consecutive observations is an approach to achieve a stationary time series [2]:

$$\nabla X_t = X_t - X_{t-1}$$

By defining a backshift operator B as $B^k X_t = X_{t-k}$ for $k \ge 0$ and t > 1, the above formula, first-order differentiation, can be expressed as:

$$\nabla X_t = X_t - X_{t-1} = X_t - BX_t = (1 - B)X_t.$$

Concerning the situation when differenced data will not appear to be stationary, it may be demanded to difference the data a second time to obtain a stationary series:

$$\nabla^2 X_t = \nabla(\nabla X_t) = \nabla(X_t - X_{t-1})$$

= $(X_t - X_{t-1}) - (X_{t-1} - X_{t-2}) = X_t - 2X_{t-1} + X_{t-2}$

which using the backshift operator may be represented as:

$$(1-B)^2 = 1 - 2B + B^2$$

$$B^2X_t = X_{t-2}$$

$$(1-B)^2 X_t = X_t - 2X_{t-1} + X_{t-2}$$

More generally, for order of differentiation d we have:

$$\nabla^d X_t = (1 - B)^d X_t$$

For a real number d using a binomial formula [3]:

$$(1+x)^d = \sum_{k=0}^{\infty} \binom{d}{k} x^k$$

the series can be expanded to [3]:

$$(1-B)^{d} = \sum_{k=0}^{\infty} (-B)^{k} \prod_{i=0}^{k-1} \frac{d-i}{k-i}$$

$$= 1 - dB + \frac{d(d-1)}{2!} B^{2} - \frac{d(d-1)(d-2)}{3!} B^{3}$$

$$+ \dots$$

The current value in time series is the function of all the past values occurred before this time point. To each past value, a weight ω_k is assigned [3]:

$$X_t = \sum_{k=0}^{\infty} \omega_k X_{t-k}$$

The application of fractional differentiation on time series allows to decide each weight for each corresponding value. All weights calculated by fractional derivative can be expressed as [3]:

$$\omega = \left\{ 1, -d, \frac{d(d-1)}{2!}, -\frac{d(d-1)(d-2)}{3!}, \dots, (-1)^k \prod_{i=0}^{k-1} \frac{d-i}{k!}, \dots \right\}$$

When we consider d as a positive integer number, there is a point when d is equal to k, then d - k = 0 and:

$$\prod_{i=0}^{k-1} \frac{d-i}{k!} = 0$$

which leads to conclusion that memory beyond that point is removed. In first-order differencing (d = 1), weights follow as (see Fig. 1 as a confirmation):

$$\omega = \{1, -1, 0, 0, \ldots\}$$

General-purpose approach of coefficient for various orders of differencing is depicted in Fig. 1. For example, if d = 0.25 and k is always an integer number, all weights

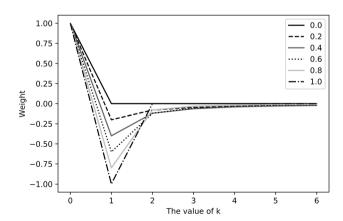


Fig. 1 Weights of the lag coefficients for various values of k. Each line is related to particular order of differencing $(d \in [0, 1])$



achieved values other than 0 which means that the memory is going to be preserved.

From the above derivation, the iterative formula for the weights of the lags can be deduced [3]:

$$\omega_k = -\omega_{k-1} \cdot \frac{(d-k+1)}{k}$$

where ω_k is the coefficient of backshift operator B^k . For the first-order differentiation, we have: $\omega_0 = 1$, $\omega_1 = -1$. $\omega_k = 0$ for k > 1.

In conclusion, the main intention to use the fractional differentiation is finding the fraction d, which is considered as minimum number needed to achieve stationarity, meanwhile keeping the maximum volume of memory in analyzed time series [3].

3 Data

This study uses four datasets with main stock indexes from different countries: WIG20 (Poland), S&P 500 (USA), DAX (Germany) and Nikkei 225 (Japan). The stock indexes were recorded from 1st June of 2010 to 30th June of 2020. The empirical distributions of mentioned indexes observed in each of the datasets are given in Figs. 2, 3, 4, 5.

Table 1 shows the results of the unit root test for all the analyzed stocks, including the augmented Dickey–Fuller (ADF) test and the Kwiatkowski–Phillips–Schmidt–Shin (KPSS) test (described in Appendix). These tests are the most widely used to determine whether a given time series is stationary [5]. The null hypothesis of the ADF test essentially assumes non-stationarity, and the null hypothesis of the KPSS test is stationarity. It can be found out that all the stock prices are non-stationary.

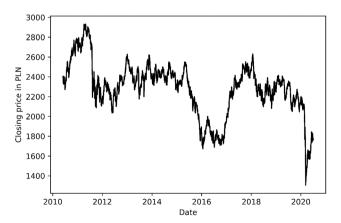


Fig. 2 Closing price time series for WIG20 index

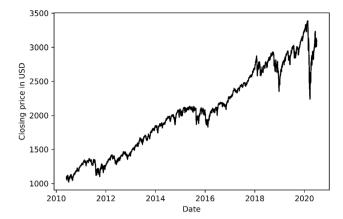


Fig. 3 Closing price time series for S&P 500 index

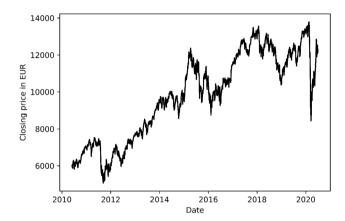


Fig. 4 Closing price time series for DAX index

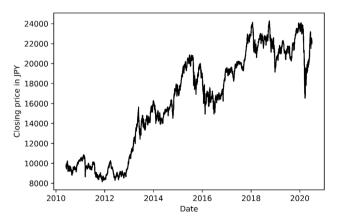


Fig. 5 Closing price time series for Nikkei 225 index

4 Method

This section describes the selected approach used to compare statistical properties of fractional differentiation with differencing of first order.



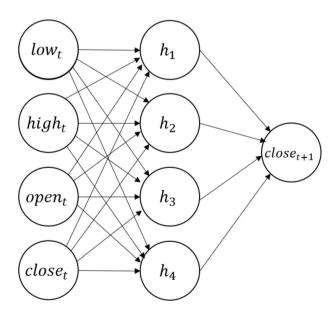
Table 1 The results of ADF and KPSS tests

Stock index	ADF	KPSS
WIG 20	- 2.22	2.64
	(0.19)	(0.01)
S&P 500	- 0.71	8.88
	(0.85)	(0.01)
DAX	- 1.61	8.12
	(0.48)	(0.01)
Nikkei 225	- 1.21	8.18
	(0.69)	(0.01)

In parentheses, the p value of tests is provided

In [6] (see also references therein) authors claim that artificial neural networks (ANN) are commonly used for forecasting stock prices indexes. In general, ANN outperforms other statistical models applied on time series due to their good nonlinear approximation ability [7]. They were inspired by the strategy human brain processes given information. One of the most frequently implemented neural networks topology is multilayer perceptron.

Like the human brain the neural network has single processing elements which are called neurons. They are connected to each other by weighted and directed edges (see Fig. 6). Commonly, neurons are aggregated into layers. Typical multilayer perceptron has three layers consisting of neurons: input layer, output layer and hidden layer. In the most simple case of artificial neural network, the edges between layers are limited to being forward edges (feed forward artificial neural networks). It means



 $\begin{tabular}{ll} Fig. \ 6 \\ An example of simple ANN with input, hidden and output layers \\ \end{tabular}$

that any element of a current layer feeds all the elements of the succeeding layer.

The goal of the neural network is mapping values from input layer to values from output layer using hidden neurons in some way. This mapping is based on modifying the weights of the connections to receive a result closer to the output. To determine the value of the output applying the activation function to a weighted sum of incoming values is needed as well. The most widely used activation functions are: the logistic function and the hyperbolic tangent (used in our study) [6].

In the first step, the system takes the values of neurons in the input layer and sums them up by the assigned weights. In this first iteration, all weights are randomized. Then, for each iteration an error is calculated, which is a difference between the achieved value and the output value. This divergence between estimated and expected value is called a loss function. Calculated loss information is propagated backward then from the output layer to all the neurons in the hidden layer that contribute directly to the output.

The learning process, where the total loss should be minimized, uses the propagated information for the adjustment of the weights of connections between neurons. The search of minimum of the loss function was performed in this study by the gradient descent method. This technique calculates the derivative of the loss function to find direction of descending toward the global minimum [8]. In practice, this calculation begins from defining the initial parameter's values of loss function and uses calculus to iteratively adjust the values to minimize the given function. There are more advanced learning techniques (based on gradient descent method) used to train neural network models such as scaled conjugate gradient, one-step secant, gradient descent with adaptive learning rate and gradient descent with momentum [9].

In this study, we are focused on predicting the closing price of a stock tomorrow $\{close_{t+1}\}$ which is the output layer using the input layer consisting prices measured a day before $\{low_t, high_t, open_t, close_t\}$. The structure of this ANN is presented in Fig. 6.

The performances of the resulting neural network are measured on the test set according to following metrics: root mean square error (RMSE):

$$RMSE = \sqrt{\frac{\sum_{i=1}^{N} (\hat{y_i} - y_i)}{N}}$$

mean absolute error (MAE):

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

where N denotes the number of observations, \hat{y} is the model prediction value and y_i is the observed value.



5 Results

In this section, the fractional differentiation method will be applied on stock indexes described in previous part. For every stock index, we are going to compute the minimum coefficient d to get stationary fractionally differentiated series.

To find the minimum coefficient d, the combination of the augmented Dickey–Fuller test statistics and Pearson correlation coefficient was used. This concept is illustrated in Fig. 7, 8, 9, 10 and Table 333 (in Appendix). The ADF statistic is on the left y-axis, with the correlation between the original series and the fractionally differenced series on the right y-axis.

The original series of WIG20 index has the ADF statistics of -2.22, while the differentiated equivalent has this statistic equal to -36.02. At a 95% confidence level, the critical value of the DF t-distribution is -2.8623. This value is presented as a dotted line in Fig. 7. The ADF statistic crosses this threshold in the area close to d=0.1. At this point, the correlation has the high value of 0.9975. This proves that fractionally differenced series is not only stationary but holds considerable memory of the original series as well.

Similarly, the ADF statistic S&P 500 index reaches 95% critical value when the differencing is approximately 0.4 (for DAX series $d \approx 0.3$ and Nikkei 225 $d \approx 0.4$) and the correlation between the original series and the new fractionally differenced series is over 99% (the same for DAX and Nikkei 225).

Figures 11, 12, 13, 14 contain original series with results of implementing the minimum coefficient d indicated above. The high correlation indicates that the fractionally differenced time series retains meaningful memory of the original series.

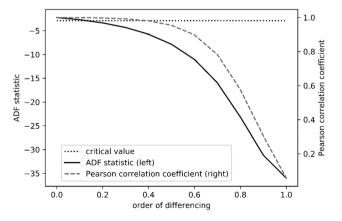


Fig. 7 ADF test statistics and Pearson correlation coefficients with the original series for various fractional orders of differencing, applied to WIG20 index

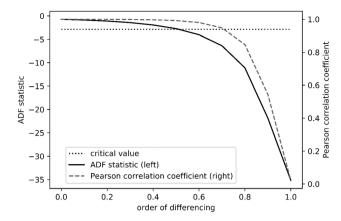


Fig. 8 ADF test statistics and Pearson correlation coefficients with the original series for various fractional orders of differencing, applied to S&P 500 index

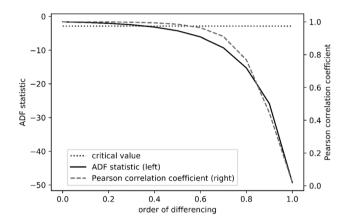


Fig. 9 ADF test statistics and Pearson correlation coefficients with the original series for various fractional orders of differencing, applied to DAX index

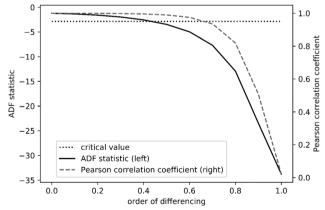


Fig. 10 ADF test statistics and Pearson correlation coefficients with the original series for various fractional orders of differencing, applied to Nikkei 225 index

The above-obtained time series are going to be implemented in creating multiplayer perceptrons for proposed



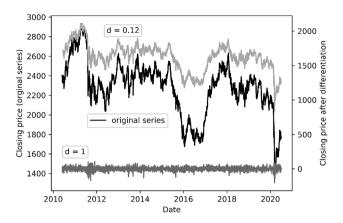


Fig. 11 WIG20 index (in black, left axis) along with fractional derivatives (shades of gray, right axis)

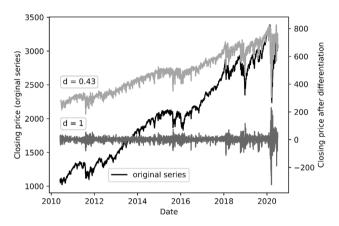


Fig. 12 S&P 500 index (in black, left axis) along with fractional derivatives (shades of gray, right axis)

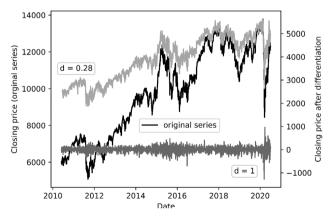


Fig. 13 DAX index (in black, left axis) along with fractional derivatives (shades of gray, right axis)

stock indexes. To begin with, the data for all stock indexes have been normalized using the following equation:

$$price_{norm} = \frac{price - \min(price)}{\max(price) - \min(price)}$$

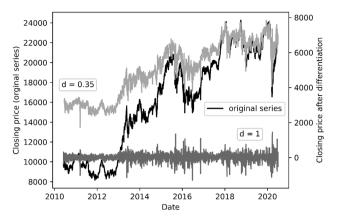


Fig. 14 Nikkei 225 index (in black, left axis) along with fractional derivatives (shades of gray, right axis)

and divided into training and testing datasets. The first 1681 days are used for training, and the last 813 used for testing process.

Feedforward neural networks were created using Keras, open-source neural network library in Python. Every network was inputted with the low, high, opening and closing price for each day t. The result layer consists of closing price on next day t+1:

$$\begin{bmatrix} low_t \\ high_t \\ open_t \\ close_t \end{bmatrix} \rightarrow [close_{t+1}]$$

It means that artificial neural network predicts the closing price of the next days using historical data from the day before.

As it is observed in Table 2, the analysis shows that for all stock indexes fractional differentiation gives better RMSE and MAE statistics obtained on test data.

The purpose of this research is not to evaluate the predictive performance of artificial neural networks, but rather to evaluate how much better a fractional differentiation is, compared to full differentiation.

6 Conclusions

In this study, the concept of fractional differentiation was evaluated on four time series datasets obtained from the known stock exchanges (from Poland, UK, Germany and Japan). In these fractionally differenced time series, selected orders of differencing vary from 0.12 to 0.43, which is far from integer differencing. For all of them, we have received a high level of linear correlation coefficients (above 0.99%), which means immense association with original series. Nonetheless, these fractional time series are



Table 2 Results of ANN on test datasets

Stock index		RMSE	MAE
WIG 20	d = 0.12	18.10	13.68
	d = 1	27.31	20.37
S&P 500	d = 0.43	36.32	20.64
DAX	d = 1	41,10	29.38
	d = 0.28	124.07	84.43
	d = 1	142.45	94.79
Nikkei 225	d = 0.35	286.86	211.96
	d = 1	317.57	249.80

stationary (indicated by the results of augmented Dickey–Fuller test), which proves that their means, variances and covariances are time-invariant.

Using fractional differentiation, we have made analyzed time series stationary while keeping its memory and predictive power. Therefore, this study has clearly demonstrated the potential of applying fractional differentiation on time series.

The previously discussed results clearly show the benefit of fractional differentiation compared to classical differentiation in terms of applied performance measurements on created artificial neural networks. Consequently, fractional differentiation used in preliminary data analysis has broad applications prospects in machine learning area, which was confirmed by predictive performance metrics.

Open Access This article is licensed under a Creative Commons Attribution 4.0 International License, which permits use, sharing, adaptation, distribution and reproduction in any medium or format, as long as you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons licence, and indicate if changes were made. The images or other third party material in this article are included in the article's Creative Commons licence, unless indicated otherwise in a credit line to the material. If material is not

included in the article's Creative Commons licence and your intended use is not permitted by statutory regulation or exceeds the permitted use, you will need to obtain permission directly from the copyright holder. To view a copy of this licence, visit http://creativecommons.org/licenses/by/4.0/.

Appendix

See Table 3.

Tests

We assume that a time series is generated by an AR(1) process:

$$y_t = \alpha y_{t-1} + \varepsilon_t$$

where ε_t is a stationary disturbance term. If $\alpha = 1$, then the analyzed process is non-stationary. The Dickey–Fuller test takes the following model:

$$\Delta y_t = \delta y_{t-1} + \varepsilon_t$$

where $\delta = \alpha - 1$ and verifies a null hypothesis that $\delta = 0$, which means that y_t is generated by AR(1). Alternative hypothesis assumes that the time series is stationary and $\delta < 0$. The estimation of $\hat{\delta}$ is calculated then using OLS regression and divided by its standard error in computing the testing statistic. The test is a one-sided left tail test. Augmented Dickey–Fuller (ADF) test adds p > 0 lags of the dependent variable Δy_t to make model dynamically complete.

In KPSS test, the following model is analyzed:

$$y_t = \beta t + r_t + \phi y_{t-1} + \varepsilon_t$$

$$r_t = r_{t-1} + u_t$$

Table 3 ADF test statistics and Pearson correlation coefficients with the original series for various fractional orders of differencing, applied to selected stock indexes

Order of differencing	WIG 20		S&P 500		DAX		Nikkei 225	
	ADF	CORR	ADF	CORR	ADF	CORR	ADF	CORR
0.0	- 2.22	1.0000	- 0.71	1.0000	- 1.61	1.0000	- 1.21	1.0000
0.1	-2.69	0.9995	-0.87	0.9999	- 1.78	0.9999	- 1.34	0.9999
0.2	-3.36	0.9975	- 1.09	0.9996	-2.05	0.9992	- 1.59	0.9995
0.3	- 4.33	0.9922	- 1.43	0.9988	-2.49	0.9976	- 1.97	0.9984
0.4	- 5.74	0.9803	- 1.93	0.9968	-3.17	0.9939	- 2.55	0.9958
0.5	- 7.84	0.9542	- 2.71	0.9921	- 4.27	0.9852	- 3.47	0.9897
0.6	- 11.04	0.8978	-4.02	0.9804	-6.09	0.9643	- 4.98	0.9748
0.7	- 15.99	0.7817	- 6.36	0.9477	- 9.29	0.9107	- 7.67	0.9351
0.8	-23.17	0.5755	- 11.10	0.8468	- 15.28	0.7681	- 12.92	0.8194
0.9	- 31.17	0.3040	-21.84	0.5441	-25.85	0.4454	-23.46	0.5074
1.0	- 36.02	0.0560	- 35.16	0.0220	- 49.28	0.0199	- 33.79	0.0202



where $t=1,2,\ldots,T$ is a deterministic trend, r_t = random walk process, ε_t are iid $(0,\sigma_\varepsilon^2)$ and $\sigma_\varepsilon^2=1$, u_t are $(0,\sigma_u^2)$ and $|\phi|<1$. If $\sigma_u^2=0$ and the initial value of r_0 is fixed, then y_t is a trend stationary process. If $\beta=0$, the process is stationary around its mean (r_0) rather than around a trend. For σ_u^2 the process is non-stationary.

References

- Granger, C.W.J., Newbold, P.: Spurious regressions in econometrics. J. Econom. 2, 111–120 (1974)
- 2. Tsay, R.: Analysis of Financial Time Series. Wiley, New Jersey (2010)
- Lopez de Prado, M.: Advances in Financial Machine Learning. Wiley, New Jersey (2018)

- Hosking, J.: Fractional differencing. Biometrika 68(1), 165–175 (1981)
- 5. Schlitzer, G.: Testing the stationarity of economic time series: further Monte Carlo evidence. Ric. Econ. 49, 125–144 (1995)
- Guresen, E., Kayakutlu, G., Daim, T.U.: Using artificial neural network models in stock market index prediction. Expert Syst. Appl. 38, 10389–10397 (2011)
- Qiu, J., Wang, B., Zhou, C.: Forecasting stock prices with longshort term memory neural network based on attention mechanism. PLoS ONE 15, 1–15 (2020)
- LeCun, Y., Bottou, L., Orr, G., Muller, K.: Efficient backprop. In: Neural Networks: Tricks of the Trade, Springer, Berlin, 9–50 (2012).
- Moghaddama, A.H., Moghaddamb, M.H., Esfandyari, M.: Stock market index prediction using artificial neural network. J. Econ. Financ. Adm. Sci. 21, 89–93 (2016)

Publisher's Note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.

