



## Modelling, forecasting and trading with a new sliding window approach: the crack spread example

Andreas Karathanasopoulos, Christian Dunis & Samer Khalil

To cite this article: Andreas Karathanasopoulos, Christian Dunis & Samer Khalil (2016) Modelling, forecasting and trading with a new sliding window approach: the crack spread example, Quantitative Finance, 16:12, 1875-1886, DOI: [10.1080/14697688.2016.1211796](https://doi.org/10.1080/14697688.2016.1211796)

To link to this article: <https://doi.org/10.1080/14697688.2016.1211796>



Published online: 14 Sep 2016.



Submit your article to this journal [↗](#)



Article views: 402



View related articles [↗](#)



View Crossmark data [↗](#)



Citing articles: 4 View citing articles [↗](#)

# Modelling, forecasting and trading with a new sliding window approach: the crack spread example

ANDREAS KARATHANASOPOULOS\*<sup>†</sup>, CHRISTIAN DUNIS<sup>‡</sup> and SAMER KHALIL<sup>†</sup>

<sup>†</sup>Olayan Business school, American University of Beirut, Beirut, Lebanon

<sup>‡</sup>Liverpool John Moores University, Liverpool, UK

(Received 19 August 2015; accepted 21 June 2016; published online 14 September 2016)

The scope of this analysis is the modeling and the tracking of the crack spread with a sophisticated new non-linear approach. The selected trading period covers 2087 trading days starting on 09/05/2005 and ending on 21/12/2015. The proposed model is a combined particle swarm optimiser (PSO) and a radial basis function (RBF) neural network which is trained using sliding windows of 300 and 400 days. This is benchmarked against a multilayer perceptron (MLP) neural network and higher order neural network using the same data-set. Outputs from the neural networks provide forecasts for 5 days ahead trading simulations. To model the spread an expansive universe of 250 inputs across different asset classes is also used. Included in the input data-set are 20 Autoregressive Moving Average models and 10 Generalized Autoregressive Conditional Heteroscedasticity volatility models. Results reveal that the sliding window approach to modelling the crack spread is effective when using 300 and 400 days training periods. Sliding windows of less than 300 days were found to produce unsatisfactory trading performance and reduced statistical accuracy. The PSO RBF model which was trained over 300 is superior in both trading performance and statistical accuracy when compared to its peers. As each of the unfiltered models' volatility and maximum drawdown were unattractive, a threshold confirmation filter is employed. The threshold confirmation filter only trades when the forecasted returns are greater than an optimized threshold of forecasted returns. As a consequence, only forecasted returns of stronger conviction produce trading signals. This filter attempts to reduce maximum drawdowns and volatility by trading less frequently and only during times of greater predicted change. Ultimately, the confirmation filter improves risk return profiles for each model and transaction costs were also significantly reduced.

*Keywords:* Spread trading; PSO RBF neural network; MLP neural network; HONN; Sliding window training; ARMA; GARCH; Threshold confirmation filters

## 1. Introduction

Petroleum refiners are exposed to price fluctuations on both sides of the refining process which may reduce profit margins. Refiner's primary risk is that posed by an increase in input (raw materials) prices while output prices such as Reformulated Gasoline Blendstock for Oxygen Blending (RBOB) gas and heating oil remain static or simultaneously decrease. This would result in narrowing of the spread and perhaps momentarily result in a negative spread as the price of crude becomes greater than the sum of output prices. In order to hedge this risk the 'Crack Spread' is traded to safeguard profit margins. The process of converting crude oil

into 'refined' outputs which include petroleum gas, gasoline, kerosene, diesel, industrial fuel oil (heating oil), lubricating oil, paraffin wax and asphalt is known as 'cracking' because crude oil is cracked to produce each by-product. In the refining industry, there are two widely used crack ratios as the hedge traded by each refiner varies based on variables such as capacity and operational configuration. Furthermore, both the inputs (grades of crude oil) and outputs vary from region to region depending on requirements for delivery and demand for finished products. The RBOB unleaded gasoline contract traded here is relatively new to the NYMEX exchange as the grade of gasoline changed in 2005 to include ethanol in the mix.

---

\*Corresponding author. Email: [ak152@aub.edu.lb](mailto:ak152@aub.edu.lb)

The first hedge is based on the 3:2:1 ratio which means that 3 barrels of crude oil are required to ‘crack’ 2 barrels of gasoline and 1 barrel of distillate heating oil fuel. The other ratio which refiners may trade is known as the 5:3:2 ratio. In this case, 5 barrels of crude are ‘cracked’ into 3 barrels of gasoline and 2 barrels of heating oil. Refiners that crack crude with a lower yield of gasoline relative to distillate are more likely to trade using the latter of the two combinations.

The spread is positive and hence profitable when the sum of by-product returns is greater than the cost to procure crude oil. As the hedge is executed based on the output side of the spread refiners would generally purchase crude oil futures to hedge rising crude prices and sell both the gasoline and heating oil futures to hedge decreasing output prices. This would be considered ‘shorting’ or selling the spread. Furthermore, these counteracting positions allow the market participant to ‘lock into’ a predetermined margin. For the purpose of this investigation, a spread between crude oil, gasoline and heating oil is formed by trading 3 futures contracts of crude oil, 2 futures contracts of RBOB unleaded gasoline and 1 futures contract of heating oil.

Motivation for this investigation derives from the initial analysis carried out by Dunis *et al.* (2005) who model the crack spread between NYMEX West Texas Intermediate (WTI) for crude oil and NYMEX Unleaded Gasoline (GAS). The conclusions of this paper reveal that neural networks offer interesting results and the motivation here is to offer more insight into the benefits of using non-linear modelling by expanding the universe of explanatory variables, to train the network over different sliding windows using both a particle swarm optimiser (PSO) algorithm and traditional back propagation algorithms. In addition, each model is filtered using a threshold confirmation filter to enhance performance. Furthermore, the spread which is investigated here also includes heating oil as an output. In general, the crack spread is calculated using three variables and not just crude oil and gasoline as traded in Dunis *et al.* (2005). Therefore, a more in depth application of neural networks is investigated to more accurately predict next day returns for the Crack Spread.

The crack spread is calculated always using three variables. The input variable is crude oil (CL) which is denominated in US dollars per barrel while the outputs consist of gasoline (RBOB) and heating oil (HO) of which both are denominated in US cents per gallon. In order to create the spread, a conversion of units is required. As the quantity of a crude contract is 1000 barrels per contract and both the gasoline and heating oil amount to 42 000 gallons per contract then the latter two are multiplied by 0.42. This is based on the calculation that there are 42 gallons of oil per barrel. Using this conversion of units the outputs are converted into US dollars per barrel as mathematically depicted in equation (1).

$$3 : 2 : 1 \text{ CRACKSPREADS}_t = \frac{((2 \times \text{RBOB} \times 0.42) + (1 \times \text{HO} \times 0.42) - (3 \times \text{CL}))}{3} \tag{1}$$

The methodology applied throughout this investigation in order to calculate the returns of the crack spread can be

seen below as provided by Butterworth and Holmes (2002) and more recently by Dunis *et al.* (2006) and Dunis *et al.* (2015):

$$\Delta S_t = \left[ \left( \frac{(P_{\text{RBOB}(t)} - P_{\text{RBOB}(t-1)})}{(P_{\text{RBOB}(t-1)})} + \frac{(P_{\text{HO}(t)} - P_{\text{HO}(t-1)})}{(P_{\text{HO}(t-1)})} \right) - \left( \frac{(P_{\text{CL}(t)} - P_{\text{CL}(t-1)})}{(P_{\text{CL}(t-1)})} \right) \right], \tag{2}$$

where  $\Delta S_t$  = Percentage change in returns of the crack spread at time  $t$

$P_{\text{RBOB}(t)}$  = is the price of RBOB at time  $t$  (in dollar per barrel)

$P_{\text{RBOB}(t-1)}$  = is the price of RBOB at time  $t - 1$  (in dollar per barrel)

$P_{\text{HO}(t)}$  = is the price of heating oil at time  $t$  (in dollar per barrel)

$P_{\text{HO}(t-1)}$  = is the price of heating oil at time  $t - 1$  (in dollar per barrel)

$P_{\text{CL}(t)}$  = is the price of crude oil at time  $t$  (in dollar per barrel)

$P_{\text{CL}(t-1)}$  = is the price of crude oil at time  $t - 1$  (in dollar per barrel)

Instead of benchmarking the proposed PSO RBF, MLP and HONN models against linear models, which is frequently criticized, this investigation utilizes informational content from traditional models. Traditional models are included in the universe of inputs to produce a mixed model approach in attempt to improve the accuracy and trading performance of each neural network. In particular, the inclusion of a GARCH volatility time series was justified as it enhanced performance by reducing volatility and maximum drawdowns. In this research, we have used 10 different GARCH outputs and 20 ARMA outputs as inputs in our non-linear models. The combination of GARCH and ARMA models has been chosen through the optimization of the results.

Preliminary research has led to a number of unanswered questions when using neural networks as a methodology for forecasting commodity spread time series. For instance, how large should the training window should be? Should the inputs be pre-processed (i.e. normalization of inputs or the removal of outliers)? What network configuration (e.g. number of hidden neurons, number of layers, etc.) should be selected? What algorithm should be used to train the data? In attempt to answer these questions, the remaining structure of this paper is presented as follows. Section 2 provides a review of all current literature that has been produced relevant to modelling the crack spread and other gasoline spreads. Section 3 presents descriptive statistics of the data used to model the spread. Section 4 presents the methodologies and estimation parameters for the particle swarm optimiser (PSO); radial basis function (RBF) neural network (NN); and the multilayer perceptron (MLP) NN; and higher order neural network (HONN). Section 5 offers an evaluation of empirical results and trading performance. This is then followed by concluding remarks and research limitations.

## 2. Literature review

### 2.1. Modelling the crack

There are number of linear methodologies that have been applied to the task of modelling, forecasting and trading various combinations of gasoline spreads as well as the crack spread investigated here. For example, Chen *et al.* (2005) also utilize threshold cointegration models when examining price adjustments for the spread between crude oil and gasoline prices. They present evidence of asymmetry in both the short- and long-run adjustments using both futures data and spot prices. Their conclusions reveal that retail gasoline prices respond asymmetrically to crude oil price changes. Later, in a similar approach to modelling the crack spread, Dunis *et al.* (2005) use both the aforementioned Enders and Granger (1998) threshold cointegration technique and numerous neural network architectures to the task of predicting next day spread returns. A fixed training period is used to train each of the networks with the training set being divided into training and test data-sets in order to avoid 'over-fitting'. Results from Dunis *et al.* (2005) reveal that the spread does in fact exhibit asymmetric adjustment. It is also observed that movements away from fair value are almost three times larger on the downside than on the upside. Overall the fair value cointegration model produces the most profitable trading performance. Later Al-Gudhea *et al.* (2007) has used a threshold cointegration models to capture the relationships between crude, spot wholesale and retail gasoline price adjustments during the period of December 1998 to January 2004. In total, four spreads are analysed. The first is a spread between crude oil prices and retail gasoline prices, the second is between crude oil prices and spot gasoline prices, the third spot gasoline prices and wholesale gasoline prices and the fourth spread is that of wholesale gasoline prices and retail gasoline prices. Test statistics from each of these spreads confirm that they are all cointegrated with evidence of asymmetric adjustments toward long-run equilibrium. Furthermore Murat and Tokat (2009), came in the same research output mentioning that neural networks can predict more accurate the crack spread against linear models such as random walk. More recently, Wang and Wu (2012) and Zhang *et al.* (2015), after investigating in the same direction, point out the power of new methodologies in terms of forecasting more accurate the crack spread.

### 2.2. Training of neural networks

Different approaches to train neural networks have been explored by many over the years and even more so in recent years. Kaastra and Boyd (1996) discuss these various techniques used to train neural networks. In his research he has pointed out the superiority of artificial intelligence in terms of forecasting more accurate indices. The most popular and widely used approach is one where the practitioner elects fixed training and validation data-sets. For example, this training approach was adopted by Dunis *et al.* (2005) who also model the crack spread. Using a fixed training and test

data-set, they train the network using 85% of the data and then validate the neural parameters over the remaining 15% of the data-set (out of sample trading). Training data-sets usually account for 70–85% of the in sample period, while the validation data-set covers anywhere from 30% to as little as 10%. Karathanasopoulos *et al.* (2013a, 2013b, 2014, 2015a, 2015b) have continued to use the same way to train the machine learning methodologies providing tables with the split of the total data-set. Another recent approach is when the practitioner randomly selects the validation data-set which is usually within the training data-set. This, however, may bias the test and reduce the accuracy when validating the training using larger out of sample data-sets. For this reason, the first approach is usually favoured by practitioners. In addition to this the first approach of selecting simultaneous in sample and out of sample data-sets allows for practitioners to test the parameters of the 'trained' neural network on more recent data which is usually more relevant than historical data. The final approach Kaastra and Boyd (1996) propose is a 'sliding window' approach as used in this investigation when training both the PSO RBF and MLP neural networks. Kaastra and Boyd (1996) call this a 'walk-forward' testing routine which is commonly adopted by commodity trading systems to model and trade data in dynamic and changing market conditions. In order to adapt to these changing conditions a sliding window is utilized to provide a more robust and time-varying approach. This technique continuously updates the training data-set and as a result it provides a more practical and realistic approach to trading financial assets.

Furthermore related to the approach of sliding window Cortez *et al.* (2001) was first who discussed the implications that may arise when selecting the duration of sliding windows. For instance, a large sliding window may increase the complexity of the neural network which could ultimately reduce the learning capabilities of the model. On the other hand, smaller windows may not contain a sufficient amount of information for the neural network to be able to train the data and produce 'informationally' significant forecasts. Finally, Karathanasopoulos *et al.* (2013d) notice, that the best sliding window is coming from checking different lengths of windows in the out of sample period.

Chang *et al.* (2004) find that performance of neural networks is enhanced when using ensemble and hybrid techniques such as combining multiple forecasts of varying sliding windows. Thawornwong and Enke (2004) use a sliding window training technique to forecast an S&P500 monthly time series using a total of 31 inputs from 24 years of data. In particular, they use four different sliding windows to capture different trends while also registering the significance of inputs during each of these windows. Tsai and Wang (2009) use an average of different sliding windows to obtain an ensemble forecast when predicting next day returns for Taiwanese electronic stocks. They run four different sliding windows and take an average of these four training sets to produce a forecast. More recently, Karathanasopoulos *et al.* (2013d) successfully have used a sliding window approach to predict daily the FTSE100 with evolutionary support vector machines. They point out that the best sliding window came after 1000 different trials in the out of sample period.

Table 1. Total data-set.

Period	Days	Beginning	End
Total data-set	2087	09/05/2005	21/12/2015
In sample period	1387	09/05/2005	05/09/2012
Out of sample	700	06/09/2012	21/12/2015
Sliding window 300 (out of sample period)	300	05/07/2013	08/07/2014
Sliding window 400 (out of sample period)	400	05/07/2013	03/01/2015

### 3. Descriptive statistics

All data were sourced from Bloomberg for the period of 09/05/2005–21/12/2015 for WTI Crude, RBOB Unleaded Gasoline and Heating Oil futures contracts. The RBOB Unleaded Gasoline contract is fairly new to the exchange as it replaced the old Unleaded Gasoline contract when Methyl Tertiary-Butyl Ether (MTBE) was phased out in 2005. This was seen to be less environmentally friendly than its alternative ethanol. As a result, this new blend now comprises 10% ethanol. Segregation of the data-set is displayed in the first table.

As presented in table 1, the modelling and trading of the PSO, RBF, the MLP and the HONN neural networks is based on two sliding training windows using 300 for the shortest period and 400 days for the longer period. The first represents 1.3 years of working days and the second covers 1.7 years of working days. Anything less than 300 days was found to produce unsatisfactory results therefore it is assumed that the training period did not include enough data points to accurately capture patterns within the data. Obviously, the best two sliding windows came from the out of sample period during the numerous back tests.

For the proposed PSO, RBF, MLP and HONN models, over-fitting is dealt with using a two pronged approach. Firstly, each of the data-sets is separated into training/test and out of sample period. Training sets account for 85%, while the remaining 15% is allocated for validation. The second control that has been tested during the in-sample backtest and implemented for the validation period is to use a fixed and constant amount of neurons in the hidden layer. For instance, for the PSO RBF model a total of 100 neurons were found to produce adequate results during the in-sample backtest while avoiding over-fitting. In the absence of a ‘feature selection method’ all inputs are selected during the training process for the MLP and HONN model. The complexity of the network is calculated based on the number of inputs as displayed in equation (3).†

$$h = (n + 1)/2 \quad (3)$$

where  $h$  = number of hidden neurons

$n$  = number of inputs

A PSO algorithm is used to calculate the number of hidden neurons for the RBF neural network. This algorithm is programmed to adapt, search for and identify the ‘optimal’ number of neurons. Results from these experiments produced an average of 100–125 neurons in the hidden layer.

In this case, the complexity of a network with as many as 125 neurons was found to ‘over fit’ the data-set. Therefore, it was decided to use fewer (100 neurons) neurons in order to reduce the risk of over fitting with a less complex network topology.

Each of these training periods produces 5-day ahead forecasts. In a similar approach, Von Mettenheim and Breiter (2012) use a sliding window of 128 days to produce forecasts for 10 days ( $t, t+1, t+2, \dots, t+10$ ) a head when modelling various stocks and ETFs. As the training process is rolling so too are the forecasts. For example, in order to obtain the predicted  $t+2$  output the window moves forward by one day to include the forecasted  $t+1$  in the training period which is used to estimate  $t+2$ . Then  $t+1$  and  $t+2$  are used in the training window to produce  $t+3$  and so on. Therefore, the sliding window approach is where the PSO-RBF and MLP networks are trained to use the last  $k$  values of a series ( $t_{n-k}, \dots, t_n$ ) to predict the next value at  $t_{n+1}$ . In practice, this means that the model only needs to be trained every ‘ $x$ ’ day(s) depending on how far in the future one wish to forecast. In this case, the neural network is retrained every 5 days to produce a forecast as traded during the out of sample validation period. More frequent retraining of a sliding window is not problematic as it takes a matter of minutes to retrain and generate forecasts. This can be done over a weekend or outside trading hours such as in the morning before market open.

The three models used here are trained to forecast the next day change in the crack spread ( $S_{t+5}$ ) using historical returns from 250 different explanatory variables.  $S_t$  is essentially the daily change in the spread as calculated in equation (2). Simple returns are used as inputs due to the fact that they enable neural networks to converge much quicker than price series data. Furthermore, many simple return time series are found to be stationary which is the main reason for quicker convergence. This is however not always the case as some time series displays unit roots.

The selection of input variables is a modelling decision that can greatly affect a model’s performance. Dunis *et al.* (2005), who also model and trade the crack spread, only use autoregressive returns of the spread to produce non-linear forecasts however for the purpose of this application a more comprehensive and significant set of inputs are considered. The aim more accurately capture and forecast the directional change of the spread by introducing more informational content in the input series. A larger universe of inputs was initially evaluated over the duration of each training window. Following numerous backtests, a total of 250 inputs were retained for out of sample trading. Included in these 250 inputs are various moving average time series based on 2, 4, 6, 8, 10, 21, 50, 100, 150, 200 and 250 days, changes in

†For this application a total of 100 hidden neurons were used for the MLP and HONN training process.

Table 2. Most significant explanatory variables.

Explanatory variable	Lag (days)	300 days sliding window (%)	400 days sliding window (%)
Spread Return Series	2	70.06	66.92
BP PLC Stock Price Returns	1	52.46	52.78
BP PLC Stock Price Returns	2	51.61	56.28
Alon USA Energy Inc. Stock Price Returns	1	56.10	54.69
Valero Energy Corp. Stock Price Returns	1	59.43	57.32
ARMA(11,20)	1	58.81	50.08
ARMA(15,17)	1	57.31	51.28
ARMA(19,18)	4	60.38	60.18
ARMA (13,13)	8	59.99	61.21
GARCH (16,12)	13	58.63	63.47
GARCH (12,17)	14	60.61	61.50
GARCH (12,19)	12	62.08	63.20

daily implied volatility was also included using the CBOE VIX index, 20 ARMA and 10 GARCH models are also incorporated into the training process. Research conducted by Karathanasopoulos *et al.* (2011) and (2015c) find that the inclusion of the ARMA models as inputs to a ‘mixed neural network’ improves both statistical accuracy and trading performance as the training of the neural network is enhanced. Therefore, the inclusion of linear models as inputs for neural network training is justified. For the most part, in this application the inclusion of volatility models is found to effectively reduce overall volatility while also improving maximum drawdowns during the training period.

The majority of existing neural network literature uses fixed training windows during in-sample data-sets which is not realistic, especially during times when the data-set is continuous or when it experiences various regime changes. Furthermore, for the proposed RBF neural network the application of a PSO algorithm in input selection also provides more insight into the significance of each input as the percentage that each is selected during the sliding windows is also recorded as displayed in table 2. This enables practitioners to see which explanatory variables are more influential during the period of 09/05/2005–21/12/2015 (2087 trading days). The difference of results between the 300 and 400 days sliding windows may indicate that each sliding window identifies different trends in the data with different inputs becoming more significant at times than others.

The percentage of time each input is selected over all of the training periods is estimated based on:

$$\text{Input selection percentage} = N/R^* \quad (4)$$

\*with  $R = (S - X)/S_{t+n}$

where  $N$  = number of sliding window repetitions an input was selected

$R$  = repetitions

$S$  = total sample data-set

$X$  = days of sliding window

$S_{t+n}$  = spread forecast horizon

By observation, table 2 provides a summary of the most significant PSO-RBF neural inputs. In particular, the ARMA and GARCH inputs prove to be among the most valuable as explanatory variables.

By including ARMA and GARCH time series the trading performance and statistical accuracy of the models was increased substantially. In addition, autoregressive time series of spread returns were also included in the modelling of the crack spread. The most significant input of the lagged spread returns from lags of 1–15 days was the 2-day lag with this being selected as often as 70.06% of the time during the 300 days sliding window period and 66.92% during the 400 day sliding window. Other more influential inputs included the daily changes in some of the refiners’ share prices. For instance, BP Plc., Alon Energy Inc. and Western Refining Inc. were all seen as more significant relative to the other refiners.

A histogram of the spread’s return series over the entire sample period is displayed in figure 1. This is found to display a leptokurtic distribution with positively high kurtosis. This however is quiet common when observing normal

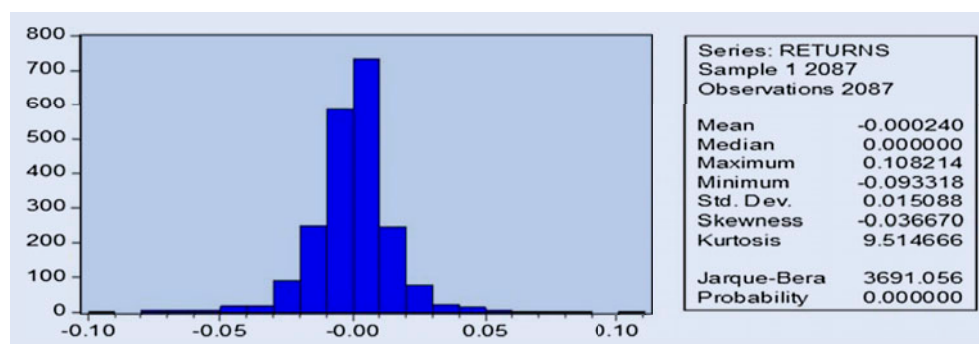


Figure 1. Spread returns (total data-set).

distributions of return series as data points tend to be highly concentrated around the mean. Furthermore, all of the spreads are confirmed to be non-normal (confirmed at a 99% confidence level by the Jarque–Bera test statistic).

All ARMA models were found to be significant at a 95% confidence level as their  $p$ -values were less than 0.05 for each of the estimated  $(p, q)$  terms. The GARCH models were deemed stable and terms for both models were also significant at 95% confidence level. Residuals were tested for serial correlation using the squared residual test revealing that serial correlation is not present in either of the models. Therefore, the estimated models are deemed adequate and have been used to estimate two of the explanatory variables which are included during the training sliding window process of the neural network.

### 4. Methodology

#### 4.1. The MLP model

The multi-layer perceptron allows the user to select a set of activation functions to explore including identity, logistic, hyperbolic tangent, negative exponential and sine.† These activation functions can be used for both hidden and output neurons. MLP also trains networks using a variety of algorithms such as gradient descent, conjugate gradient and BFGS (Broyden, Fletcher, Goldfarb and Shanno). Here, the logistic activation function and gradient descent algorithm are used.

The network architecture of a conventional MLP network can be illustrated as seen in figure 2 below:

where:  $x_t^{[n]}$  ( $n = 1, 2, \dots, k + 1$ )  
 is the model inputs (including the input bias node) at time  $t$   
 $h_t^{[m]}$  ( $m = 1, 2, \dots, j + 1$ )  
 are the hidden node outputs (including the hidden bias node)  
 $\tilde{y}_t$   
 MLP model's output  $u_{jk}$  and  $w_j$   
 are network weights

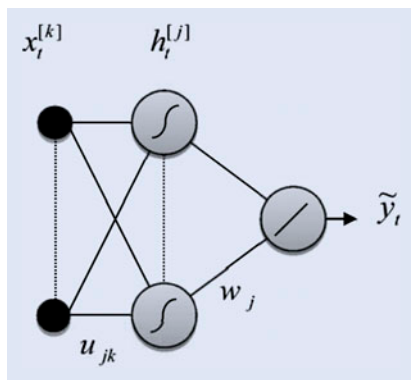




Figure 2. A single output, inter-connected MLP model.

†This activation function is considered to be non-monotonic in that it is difficult to make weights vary sufficiently from their initial position. This can result in much larger numbers of local minima in the error surface (Sopena et al. (1999).

 is the sigmoid transfer function:

$$S(x) = \frac{1}{1 + e^{-x}} \tag{5}$$

 is the linear output function:

$$F(x) = \sum_i x_i \tag{6}$$

The error function to be minimized is:  $E(u_{jk}, w_j)$

$$= \frac{1}{T} \sum_{t=1}^T (y_t - \tilde{y}_t(u_{jk}, w_j))^2 \tag{7}$$


with  $y_t$  being the target value and  $T$  the number of trading days.


Training and selection of a network is halted once profit (in the form of an annualized return) is at its greatest during the in-sample test period. This is a clear characteristic for all the neural networks.

#### 4.2. Higher order neural networks


Higher order neural networks (HONNs) were first introduced by Giles and Maxwell (1987) and were called ‘Tensor Networks’. A lot research has been produced for HONN models, some good findings came from Zhang et al. (2002), who said that significant advantage of HONN models is that these models are able to provide some rationale for the simulations they produce and thus can be regarded as ‘open box’ rather than ‘black box’ Recently, Karathanasopoulos et al. (2011) mention the superiority of HONN benchmarking with other 10 predicting models in forecasting the ASE 20 Greek index. More in depth HONN models are able to simulate higher frequency, higher order non-linear data, and consequently provide superior simulations compared to those produced by ANNs. While they have already experienced some success in the field of pattern recognition and associative recall, HONNs have not yet been widely used in finance. The architecture of a three input second order HONN is shown in figure 3 below:

where:  $x_t^{[n]}$  ( $n = 1, 2, \dots, k + 1$ )  
 are the model inputs (including the input bias node) at time  $t$   
 $\tilde{y}_t$   
 is the HONNs model output  $u_{jk}$  are the network weights

 are the model inputs

 is the transfer sigmoid function:

$$S(x) = \frac{1}{1 + e^{-x}} \tag{8}$$

 is a linear function:

$$F(x) = \sum_i x_i \tag{9}$$

The error function to be minimized is:

$$E(u_{jk}, w_j) = \frac{1}{T} \sum_{t=1}^T (y_t - \tilde{y}_t(u_{jk}, w_j))^2, \text{ with } y_t \text{ being the target value} \tag{10}$$

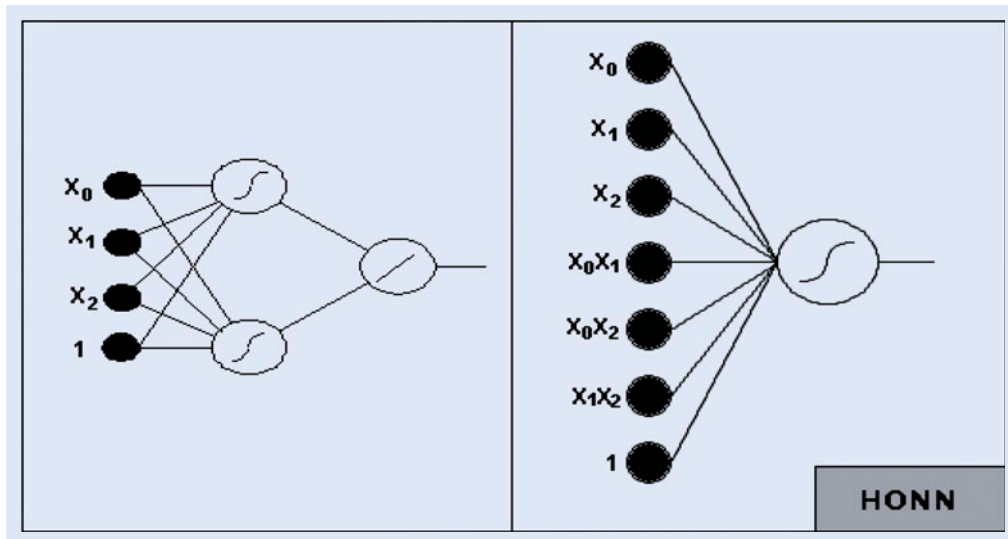


Figure 3. A single output, inter-connected HONN model.

HONNs use joint activation functions; this technique reduces the need to establish the relationship between inputs when training. Furthermore, this reduces the number of free weights and means that HONNs can be faster to train than MLPs. However, because the number of inputs can be very large for higher order architectures, orders of four and over are rarely used. That’s why the 250 inputs it’s quite unique for HONN models. Another advantage of the reduction of free weights means that the problems of overfitting and local optima affecting the results can be largely avoided, Knowles *et al.* (2009). Karathanasopoulos *et al.* (2011) and (2012b) noticed that HONN and MLP are outperforming the linear models such as Naïve strategy, moving averages and ARMA models.

4.3. The PSO radial basis function model

A radial basis function (RBF) neural network is a feed-forward neural network, where hidden layers do not implement an activation function, but instead a radial basis function. As discussed by Park *et al.* (2002), input values in an RBF network are each assigned to a node in the input layer and then passed directly through to the hidden layer without weights. On the other hand, traditional neural networks such as the MLP pass inputs through to the hidden layer as weighted computations.

The PSO aspect introduces a hybrid approach to the training of a network and hence the refinement of its forecasting accuracy has been compared to that achieved by Genetic Programming Algorithms. PSO was first introduced by Kennedy and Eberhart (1995) as a stochastic optimizer during the neural network training process. Kennedy and Eberhart (1995) developed the PSO algorithm based on observations found within nature such as the social behaviour found within a flock of birds or a school of fish. With these observations as a basis, the algorithm is developed to search a fixed space in attempt to identify optimal positions within this space to best solve a

predefined problem. In particular, PSO optimization reduces the time it takes to train neural networks by simplifying the complex calculations found within traditional Neural Networks and determining the optimal number of hidden layers. Many academics have previously researched standard RBF neural networks; however, the combination of PSO and neural networks is relatively new to time series analysis. As explained by Chen and Qian (2009), PSO optimizes parameters within a traditional RBF. In particular, this optimization helps overcome inefficiencies associated within the standard back propagation algorithm. Karathanasopoulos *et al.* (2012c) and (2013a) use the PSO to optimize neural networks with success. Their findings are quite impressive and promising.

The RBF neural network approximates a desired function by the superposition of non-orthogonal, radially symmetric functions as discussed in more detail by Karathanasopoulos *et al.* (2013a). The networks architecture is depicted below in figure 4.

Here, the Gaussian radial basis function is used in the hidden layer as this is the most common found in existing financial time series literature.  $x_t(n = 1, 2, \dots, N + 1)$

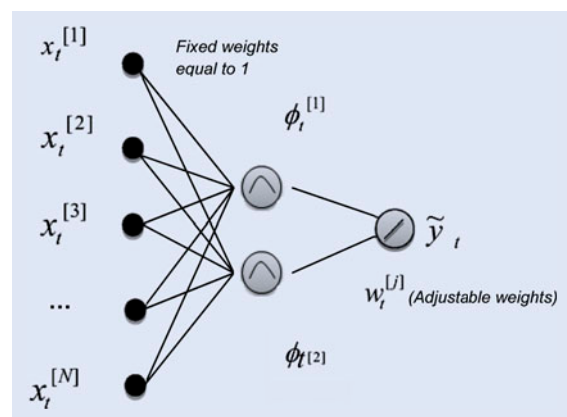


Figure 4. Radial basis function neural network (with two hidden nodes).


are the model inputs (including the input bias node) $\tilde{y}_t$   
 is the RBF model's output  $w_t^{[j]} (j = 1, 2)$   
 are the adjustable weights

 is the Gaussian function:

$$\phi_t^i(x) = e^{-\frac{\|x - C_i\|^2}{2\sigma_i^2}} \tag{11}$$

where

$C_i$  is a vector indicating the centre of the Gaussian Function and  $\sigma_i$  is a value indicating its width.  $C_i$ ,  $\sigma_i$  and the weights  $w_i$  are parameters which are optimized by the PSO algorithm during a learning phase while training the RBF neural network.

 is the linear output function:

$$U(x) = \sum_i x_i \tag{12}$$

The error function to be minimized is:  $E(C, \sigma, w_t)$

$$= \frac{1}{T} \sum_{t=1}^T (y_t - \tilde{y}_t(w_t, C, \sigma))^2 \tag{13}$$

with  $y_t$  being the target value and T the number of trading days.

In order to maximize annualize returns an additional fitness function is employed as defined below in equation (14). This approach was first introduced by Karathanasopoulos *et al.* (2012a, 2013a, 2013b, 2013c, 2013d).

The annualised return function to be maximized is:  $R^A$

$$- \text{RMSE} - (n * 10^{-2}) \tag{14}$$

where

- $R^A$  = annualized return
- RMSE = mean square error defined in equation (13).
- $n$  = number of inputs.

The  $R^A$  terms range from  $-0.4$  to  $0.5$  while experimental results indicated that the maximum value for the MSE term is  $0.01$ . These parameters are established so that the algorithm can primarily search for profitable forecasts with statistical performance becoming of secondary importance.

The hybrid methodology of combining a PSO with an RBF neural network was first inspired by Li and Xiao (2008) and is also an extension of the PSO-RBF methodology proposed by Karathanasopoulos *et al.* (2013a). The PSO methodology is used to locate the parameters  $C_i$ , of the RBF neural network, while at the same time locating the optimal feature subset which should be used as inputs to the RBF network.

The complexity of a traditional neural network is reduced by applying the PSO algorithm to refine the training process. As applied by Karathanasopoulos *et al.* (2012c) and (2013a) the PSO algorithm encodes network weights as

particle components with each particle evaluating inputs based on minimizing the error function in equation (10). PSO parameters are also 'adaptive' as depicted in equations (15)–(17). This proves beneficial to a wider range of users. Therefore, 'velocity' as described originally by Kennedy and Eberhart (1995) is adaptable with the algorithm retaining knowledge of an input's (particle) best position within the population (swarm).

With the PSO algorithm the traditional neural network weight matrix is reorganized as an array of randomly initialized particles to commence the optimization procedure. During this search, the PSO algorithm is assessing 'global' and 'local' variants. A local variant is an individual particle's best solution achieved thus far while the global variant is the best solution achieved in the entire population of particles. Furthermore, Mohaghegi *et al.* (2005) note that particles have a tendency to repeat their past behaviour (cognitive) as well as follow the behaviour of those particles deemed 'fit' (socialization). The eventuality of this behaviour is that the population of particles converges to create an optimal solution. Upon the completion of iterations the particles return to their best position which is identified during the search/training process.

$$W_{(T)} = (0.4/N^2) * (T - N)^2 + 0.4 \tag{15}$$

$$c1_{(T)} = (-2) * T/N + 2.5 \tag{16}$$

$$c2_{(t)} = (2) * T/N + 0.5 \tag{17}$$

where

- $T$  = is the current iteration
- $N$  = is the total number of iterations

Weights are decreased from  $1.0$  to  $0.4$  during the training phase in search of a candid solution to the proposed problem. In selecting the appropriate training set the termination criterion applied to the PSO algorithm is  $10^{-3}$ . Ultimately, training is stopped once the number of iterations reaches  $1000$  or the profit in the form of annualized returns is at its maximum.

### 5. Empirical results

The general trading rule is to long the spread on a positive forecast and short the spread when a negative forecast is indicated. When consecutive positive or negative signals are generated then the position is held from the previous signal. Longing the spread or buying the spread is when WTI Crude oil is sold and both Heating Oil and RBOB Gasoline are bought. Shorting the spread or selling the spread occurs when WTI Crude is bought and both Heating Oil and RBOB gasoline are sold.

Table 3. Out-of-sample sliding windows trading statistics.

Statistical performance Sliding training windows Forecast	PSO RBF model		MLP model		HONN Model	
	300	400	300	400	300	400
	5 days ahead	5 days ahead	5 days ahead	5 days ahead	5 days ahead	5 days ahead
MAE	0.0100	0.0112	0.0205	0.0203	0.0332	0.0345
MAPE	166.87%	158.46%	420.05%	442.12%	469.16%	469.18%
RMSE	0.0194	0.0196	0.0263	0.0260	0.0345	0.0444
PT-Statistics	13.23	13.10	12.78	12.62	11.25	11.14
DM	-3.25	-3.45	-4.24	-4.45	-4.87	-4.90
Correct Directional Change (CDC)	60.38%	59.87%	55.23%	54.65%	53.29%	53.40%

### 5.1. Statistical performance

As it is standard in the literature, in order to evaluate statistically the obtained forecasts, the root mean squared error (RMSE), the mean absolute error (MAE), the mean absolute percentage error (MAPE), the correct directional change (CDC), the Pesaran–Timmermann (PT) and Diebold–Mariano (DM) statistics are computed. The mathematical formula of this statistic is presented in Appendix A1. The lower the output for MAE, MAPE and RMSE the better the forecasting accuracy of the model concerned. The PT test Pesaran and Timmermann (1992) examines whether the directional movements of the real and forecast values are in step with one another. In other words, it checks how well rises and falls in the forecasted value follow the actual rises and falls of the time series. The null hypothesis is that the model under study has no power on forecasting the relevant crack spread. Statistics are computed by taking the average of 1000 executions in order to reduce the variance of each forecast. As neural networks are stochastic by nature it is in the best interest of a practitioner to use an average derived from numerous models. In order to further verify the statistical superiority of the proposed algorithm, the Diebold and Mariano (1995) DM statistic for predictive accuracy is computed, while the RMSE is considered as the loss function. The test is applied in the two consecutive out-of-sample sliding window periods. Table 3 below presents all the statistics measures for both windows.

From a statistical perspective, the PSO-RBF model which is trained over 300 days is the most accurate when predicting  $t + 5$  returns. In particular, the CDC statistic is more than 50%. A CDC of greater than 50% is more desirable

and profitable. The MLP and the HONN sliding window models are also found to be less accurate in comparison to the PSO-RBF models. For the MAE, MAPE and RMSE statistics the lower they are the more accurate a model is considered to be. As explained by Dunis (1996) the RMSE, MAE and the MAPE statistics are ‘scale-dependent’ measures. These provide a modeller with statistics to compare each of the models with actual crack spread returns. Finally, PT-statistics indicate that all models continue to forecast accurately the directional change of the two sliding window approaches for all the three non-linear models and the statistical superiority of RBF-PSO is confirmed as the realizations of the DM statistic are negative.

### 5.2. Trading performance

Numerous sliding windows were back tested and then traded for the purpose of forecasting the crack spread. As mentioned previously, any windows with less than 300 days of observations were found to produce unsatisfactory results. The best sliding windows were providing 300 and 400 observations. Table 4 presents the trading performance of the crack spread in the out of sample period without filter for 300 and 400 days sliding windows.

By observation of table 4, the PSO-RBF which was trained using a 300 days sliding window achieved the highest annualized returns and the best risk return trade off. This is challenged closely by the PSO-RBF which is trained over 500 days. The MLP and HONN models which were also trained over 300 and 400 days ranked 3rd, 4th, 5th and 6th consecutively. Another interesting observation is that

Table 4. Out-of-sample unfiltered trading performance.

Trading performance Sliding training windows Forecast horizon	PSO RBF model		MLP model		HONN model	
	300	400	300	400	300	400
	5 days ahead	5 days ahead	5 days ahead	5 days ahead	5 days ahead	5 days ahead
Gross annualized return	52.82%	48.84%	32.02%	30.86%	30.67%	29.65%
Annualized volatility	20.65%	20.46%	20.43%	20.12%	20.96%	20.75%
Maximum cumulative drawdown	-30.90%	-28.10%	-32.31%	-31.09%	-29.65%	-29.87%
Information ratio	2.06	1.90	1.01	1.01	0.97	0.98
# Transactions (annualized)	99	95	100	104	103	107
Total trading days	2087	2087	2087	2087	2087	2087
Transaction costs (annualized)	10.08%	9.89%	10.27%	10.37%	10.28%	9.24%
Net annualized return	42.74%	38.95%	21.75%	20.49%	20.39%	20.41%
Ranking	1	2	3	4	5	6

Table 5. Out-of-sample filtered trading performance.

Trading performance Sliding training windows Forecast horizon	PSO RBF model		MLP model		HONN model	
	300 5 days ahead	400 5 days ahead	300 5 days ahead	400 5 days ahead	300 5 days ahead	400 5 days ahead
Gross annualized return	50.99%	48.84%	37.02%	35.86%	32.56%	31.67%
Annualized volatility	22.19%	22.19%	22.17%	22.35%	22.23%	22.78%
Maximum cumulative drawdown	-30.67%	-29.67%	-28.785	-31.78%	-34.89%	-29.87%
Information ratio						
# Transactions (annualized)	95	98	94	112	100	102
Total trading days	2087	2087	2087	2087	2087	2087
Transaction costs (annualized)	11.25%	11.00%	12.12%	11.53%	11.26%	10.63%
Net annualized return	39.74%	37.48%	24.90%	24.33%	21.30%	21.04%
Ranking	1	2	3	4	5	6

the PSO RBF the MLP models and the HONNs which were trained using a sliding window of 300 days had much worse maximum drawdowns in comparison to their respective 500 days models. However, in order to minimize maximum drawdowns a threshold filter which was optimized during the training period is used to filter each model. Table 5 presents the trading performance of the crack spread in the out of sample period with filter for 300 and 400 days sliding windows.

Results from a filtered trading simulation are presented in table 5. With this threshold filter the model only trades when the PSO-RBF, the MLP NN and the HONN models produce forecasts greater than 'x' or less than 'x'. These 'x' parameters are optimized during the in sample period as a threshold for trading each of the models. When comparing each of the forecasted return series it is clear that the RBF-PSO in both sliding windows outperforms the other two neural networks giving the second position to MLP and third the HONN. Using this filter, only larger more significant forecasts are traded while smaller less significant changes in the spread are filtered out. This minimizes maximum drawdowns and reduces volatility while also increasing annualized returns.

## 5. Concluding remarks

Results from empirical analysis clearly show that the sliding window technique for training the proposed PSO-RBF neural network offers a mixture of positive results. The same is also true for the MLP and HONN neural network. Furthermore, the inclusion of linear models as inputs also assists in enhancing the performance of both the PSO-RBF and MLP models. This is corroborated by Makridakis (1989), Clemen (1989), Newbold and Granger (1974), Palm and Zellner (1992) and recently Karathanasopoulos *et al.* (2011), who all establish that forecasts are improved by combining different linear forecasting methodologies when compared to individual forecasts. For the PSO-RBF models a feature selection method is explored using the PSO algorithm to optimize the inputs. During both sliding windows only the more significant inputs are selected to train the PSO-RBF NN. Each time an input is selected the algorithm produces a '1' and when an input is not selected then a '0' is generated. At the end of the trading period the algorithm then

calculates a total for each input as a percentage of time each were selected. Over the 300 and 400 day sliding windows a handful of more significant explanatory variables emerged. Table 2 in the descriptive statistics section summarizes the more significant inputs over these periods. In summary, the longer term moving average inputs along with the ARMA and GARCH inputs ranked among the most significant. On the other hand, the MLP and HONN models used all of the inputs for its training as no optimization algorithms were employed during the input selection phase. In conclusion, after applying statistical and numerical measures the ranking of the models proves the superiority of the RBF-PSO in both windows giving the second position to the MLP model and finally to HONN model. The results are giving a clear picture of the market and promising ideas for further research in the area of forecasting with sliding windows and a huge range of inputs.

## Disclosure statement

No potential conflict of interest was reported by the authors.

## References

- Al-Gudhea, S., Dibooglu, S. and Kenc, T., Do retail gasoline prices rise more readily than they fall? A threshold cointegration approach *J. Econ. Bus.*, 2007, **59**, 560–574.
- Butterworth, D. and Holmes, P., Inter-market spread trading: Evidence from UK index futures markets. *Appl. Financ. Econ.*, 2002, **12**(11), 783–790.
- Chen, L.H.C., Finney, M. and Lai, K. S., A threshold cointegration analysis of asymmetric price transmission from crude oil to gasoline prices. *Econ. Lett.*, 2005, **89**, 233–239.
- Chen, Z. and Qian, P., Application of PSO-RBF neural network in network intrusion detection. *Intell. Inform. Technol. Appl.*, 2009, **75**, 362–364.
- Clemen, R.T., Combining forecasts: A review and annotated bibliography. *Int. J. Forecasting*, 1989, **5**, 559–583.
- Cortez, P., Rocha, M. and Neves, J., Evolving time series forecasting neural network models. In *Proc. of the 3rd Int. Symposium on Adaptive Systems: Evolutionary Computation and Probabilistic Graphical Models (ISAS 2001)*, Havana, Cuba, pp. 84–91, 2001.
- Chang, P.C., Wang, Y.W. and Yang, W.N., An investigation of the hybrid forecasting models for stock price variation in Taiwan. *J. Chin. Inst. Ind. Eng.*, 2004, **21**(4), 358–368.

- Diebold, F.X. and Mariano, R.S., Comparing predictive accuracy. *J. Bus. Econ. Stat.*, 1995, **13**, 253–263.
- Dunis, C., The economic value of neural network systems for exchange rate forecasting. *Neural Network World*, 1996, **1**, 43–55.
- Dunis, C., Laws, J. and Evans, B., Modelling and trading the gasoline crack spread: A non-linear story. *Derivat. Use Trad. Regul.*, 2005, **12**(1–2), 126–145.
- Dunis, C., Laws, J. and Evans, B., Modelling and trading the soybean-oil crush spread with recurrent and higher order networks: A comparative analysis. *Neural Network World*, 2006, **13**(3/6), 193–213.
- Dunis, C., Laws, J., Middleton, P.W. and Karathanasopoulos, A., Modelling and trading the corn/ethanol crush spread with neural networks. *Eur. J. Finance*, 2015, **21**(4), 352–375.
- Enders, W. and Granger, C.W.J., Unit-root tests and asymmetric adjustment with an example using the term structure of interest rates. *J. Bus. Econ. Stat.*, 1998, **16**(3), 304–311.
- Giles, L. and Maxwell, T., Learning invariance and generalization in higher order neural networks. *Appl. Optics*, 1987, **26**, 4972–4978.
- Kaastra, I. and Boyd, M., Designing a neural network for forecasting financial and economic time series. *Neurocomputing*, 1996, **10**(3), 215.
- Karathanasopoulos, A., Laws, J. and Dunis, C., Modelling and trading the Greek stock market with mixed neural network models. *Appl. Financ. Econ.*, 2011, **21**, 1793–1808.
- Karathanasopoulos, A., Laws, J., Sermpinis, G. and Dunis, C., Forecasting and trading the EUR/USD exchange rate with gene expression and psi sigma neural networks. *Expert Syst. Appl.*, 2012a, **39**(10), 8865–8877.
- Karathanasopoulos, A., Theofilatos, K., Sermpinis, G., Amorgianiotis, T., Georgopoulos, E. and Likothanassis, S., Modelling and trading the DJIA financial index using neural networks optimized with adaptive evolutionary algorithms. In *Engineering Applications of Neural Networks*, pp. 453–462, 2012b (Springer: Berlin Heidelberg).
- Karathanasopoulos, A., Sermpinis, G., Theofilatos, K., Georgopoulos, E. and Dunis, C., A hybrid radial basis function and particle swarm optimization neural network approach in forecasting the EUR/GBP exchange rates returns. *Eng. Appl. Neural Networks*, 2012c, **311**, 413–422.
- Karathanasopoulos, A., Sermpinis, G., Theofilatos, K., Georgopoulos, E. and Dunis, C., Forecasting foreign exchange rates with adaptive neural networks using radial-basis functions and particle swarm optimization. *Eur. J. Operat. Res.*, 2013a, **225**(3), 528–540.
- Karathanasopoulos, A., Dunis, C., Likothanassis, S., Sermpinis, G. and Theofilatos, K., A hybrid genetic algorithm-support vector machine approach in the task of forecasting and trading. *J. Asset Manage.*, 2013b, **14**, 52–71.
- Karathanasopoulos, A., Vasilakis, G., Theofilatos, K., Georgopoulos, E. and Likothanassis, S., A genetic programming approach for EUR/USD exchange rate forecasting and trading. *Comput. Econ.*, 2013c, **42**(4), 415–431.
- Karathanasopoulos, A., Middleton, P., Theofilatos, K., Georgopoulos, E. and Likothanassis, S., Modeling and trading FTSE100 index using a novel sliding window approach which combines adaptive differential evolution and support vector regression. *Artif. Intell. Appl. Innovation*, 2013d, **1**, 486–496.
- Karathanasopoulos, A., Sermpinis, G., Laws, J. and Dunis, C., Modelling and trading the Greek stock market with gene expression and genetic programming algorithms. *J. Forecasting*, 2014, **33**(8), 596–610.
- Karathanasopoulos, A., Stasinakis, C., Sermpinis, G. and Theofilatos, K., Modeling, forecasting and trading the EUR exchange rates with hybrid rolling genetic algorithms support vector regression forecast combinations. *Eur. J. Operat. Res.*, 2015a, **247**(3), 831–846.
- Karathanasopoulos, A., Sermpinis, G., Stasinakis, C. and Theofilatos, K., Forecasting US unemployment with radial basis neural networks, Kalman filters and support vector regressions. *Comput. Econ.*, 2015b, **23**(2), 1–19.
- Kennedy, J. and Eberhart, R., Particle swarm optimization. *Proc. IEEE Int. Conf. Neural Networks*, 1995, **4**, 1942–1948.
- Knowles, A., Hussein, A., Deredy, W., Lisboa, P. and Dunis, C., Higher-order neural networks with Bayesian confidence measure for prediction of EUR/USD exchange rate. *Artif. High. Order Neural Networks Econ. Bus.*, 2009, **1**, 48–59.
- Li, J. and Xiao, X., Multi-swarm and multi-best particle swarm optimization algorithm. In *Proc. of the 7th World Congress on Intelligent Control and Automation*, Chongqing, China, pp. 6281–6286, 2008.
- Makridakis, S., Why combining works? *Int. J. Forecasting*, 1989, **5**, 601–603.
- Mettenheim, V.H.J. and Breitner, M.H., Forecasting daily highs and lows of liquid assets with neural networks. In *Operations Research Proceedings, Selected Papers of the Annual International Conference of the German Operations Research Society*, Hannover, 2012.
- Murat, A. and Tokat, E., Forecasting oil price movements with crack spread futures. *Energy Econ.*, 2009, **31**(1), 85–90.
- Mohaghegi, S., Valle, Y., Venayagamoorthy, G. and Harley, R., A comparison of PSO and backpropagation for training RBF neural networks for identification of a power system with statcom. In *Proc. IEEE Swarm Intelligence Symposium*, Pasadena, CA, pp. 381–384, 2005.
- Newbold, P. and Granger, C.W.J., Experience with forecasting univariate time series and the combination of forecasts (with discussion). *J. Stat.*, 1974, **137**, 131–164.
- Palm, F.C. and Zellner, A., To combine or not to combine? Issues of combining forecasts. *J. Forecasting*, 1992, **11**, 687–701.
- Park, J.W., Harley, R.G. and Venayagamoorthy, G.K., Comparison of MLP and RBF neural networks using deviation signals for on-line identification of a synchronous generator. *Proc. Power Eng. Soc. Winter Meet. IEEE*, 2002, **1**(27–31), 274–279.
- Pesaran, M.H. and Timmermann, A., A simple nonparametric test of predictive performance. *J. Bus. Econ. Stat.*, 1992, **10**(4), 461–461.
- Sopena, J.M., Romero, E. and Alquezar, R., Neural networks with periodic and monotonic activation functions: A comparative study in classification problems. In *Proceedings of the 9th International Conference on Artificial Neural Networks*, pp. 104–115, 1999.
- Thawornwong, S. and Enke, D., The adaptive selection of financial and economic variables for use with artificial neural networks. *Neurocomputing*, 2004, **56**, 205–232.
- Tsai, C.F. and Wang, S.P., Stock price forecasting by hybrid machine learning techniques. *Proc. Int. Multi-Conf. Eng. Comp. Sci.*, 2009, **1**, 755–760.
- Wang, Y. and Wu, C., What can we learn from the history of gasoline crack spreads? Long memory, structural breaks and modelling implications. *Econ. Model.*, 2012, **29**(2), 349–360.
- Zhang, M., Xu, S.X. and Fulcher, J., Neuron-adaptive higher order neural-network models for automated financial data modelling. *IEEE T. Neural Networ.*, 2002, **13**(1), 188–204.
- Zhang, J.L., Zhang, Y.J. and Zhang, L., A novel hybrid method for crude oil price forecasting. *Energy Econ.*, 2015, **49**, 649–659.

**Appendix 1.**

**A.1. Performance measures**

Root mean squared error (RMSE)	$RMSE = \sqrt{(1/N) * \sum_{t=1}^{t+N} (\bar{\sigma}_t - \sigma_t)^2}$
Mean absolute error (MAE)	$MAE = (1/N) * \sum_{t=1}^{t+N}  \bar{\sigma}_t - \sigma_t $
Mean absolute percentage error (MAPE)	$MAPE = (1/N) * \sum_{t=1}^{t+N} \left  \frac{\bar{\sigma}_t - \sigma_t}{\sigma_t} \right $
Correct directional change (CDC)	$CDC = (100/N) * \sum_{t=1}^{t+N} Dt$ <p>where <math>Dt = 1</math> if <math>(\sigma_t - \sigma_{t-1}) * (\bar{\sigma}_t - \bar{\sigma}_{t-1}) &gt; 0</math>, Else <math>Dt = 0</math></p>
Annualized return	$R^A = 252 * \frac{1}{N} \sum_{t=1}^N R_t$ with $R_t$ being the daily return
Cumulative return	$R^C = \sum_{t=1}^N R_t$ with $R_t$ being the daily return
Annualized volatility	$\sigma^A = \sqrt{252} * \sqrt{\frac{1}{N-1} * \sum_{t=1}^N (R_t - \bar{R})^2}$
Information ratio	$IR = \frac{R^A}{\sigma^A}$
Maximum drawdown	<p>Maximum negative value of <math>\sum (R_t^c)</math> over the period</p> $MaxDD = \text{Min} \left[ R_t - \text{Max} \left( \sum_{t=1}^N R_t \right) \right]$