Helsinki University of Technology
Mat-2.4177 Seminar on Case Studies in Operations Research
Spring 2008

EXPLORING ALGORITHMS FOR AUTOMATED FX TRADING – CONSTRUCTING A HYBRID MODEL

Final Report
15 May, 2008

Jenni Koskinen (Head of Project)
Jussi Airas
Tuomas Nummelin
Timo Pekkala
Jani Starczewski

Table of Contents

Executive Summary

This work originated from a (successful) attempt to construct mathematical algorithms to determine when to buy and sell currency. It was requested by Nordea Markets in January 2008 in the framework of the Seminar on Case Studies in Operations Research organised at the Helsinki University of Technology.

In this report, we examine fully automated foreign exchange trading system applications and simulate them to take stock of their applicability. Based on a literature study and careful consideration, we implement three usable and intelligent trading algorithms as the basis for the automated trading systems. Our systems take as input past market prices and return a trading signal indicating whether to buy or sell. The underlying algorithms include a genetic algorithm, a recurrent reinforcement learning algorithm and a specifically designed artificial neural network. What's more – based on these basic trading algorithms, we construct two hybrid models the aim of which is to combine the signals indicating the preferred positions of the underlying submodels in order to manage currency positions even better.

Our approach is driven by the project assignment and the guidance we were given during the project. As for trading strategy, all our models assume that agents trade fixed position sizes in a single exchange rate and are able to draw on fixed credit line from which they can borrow in home or foreign currency during the trading period. That is to say, we decided to implement trading systems which project down onto the discrete space for short, neutral and long positions. Therefore, our systems cannot distinguish between strong and weak training data signals resulting in the actual decision to transact and determining the actual positions. With this approach, the number of trades generated per day and the resulting transaction costs are higher than would probably be realistic in some scenarios. We did not allow the systems to wait and hold.

We report results that are obtained by applying the approaches described above to three currency markets. We trade the highly liquid EUR/USD market, the EUR/RUB market and the EUR/SEK market. For all the currency crosses, the data was retrieved on a minute basis and spanned a period from 10 January 2008 until 12 March 2008. Bid and ask quotes were separated in the data. But since this type of data is not without difficulties, we used filtering of the data – for instance, we used averages and removed outliers to avoid any major errors resulting from potential errors in the data.[1] The intraday test set contains 420 data points[2] and a total of 33 trading days are simulated in each currency market. What is even more important is the fact commonly acknowledged by professional traders that trading in these markets requires different trading approaches due to different market characteristics. We, therefore, expect our algorithms to perform differently in the selected three currency markets.

Table 1 reveals the potentially large draw-downs in profits. This is an important practical disadvantage of the implemented models and can be partly explained by the input data set. The RRL model is the only model that shows positive return for the Russian ruble and the Swedish krona. Both the hybrid models are able to generate profits when trading the US dollar. No other algorithm can achieve this, although the RRL comes close.

[1] Since Bloomberg is the source of the data, we have no certainty whether the original data is screened for irregular quotes or for any other anomalies, what is the exact relation between ticks and minute-based quotes, and whether there are also representative quotes in the data.
[2] Trading hours are 10:00 – 17:00 Finnish time.

Table 1 provides a summary of the profits that the models were able to generate during our 33-day testing period. We have given separate figures for each of the tested currencies.

Table 1 Average daily profits generated by the models during the 33 testing period (10^3 €).

	RUB	SEK	USD
Genetic Algorithm	-44	-26	-10
RRL	3	2	-1
Neural Network	-70	-38	-10
Hybrid 1	-12	-5	4
Hybrid 2	-2	-9	14

Table 2 provides an alternative approach to comparing the models' competitiveness. We have counted the number of cases in which a particular model has made the best profit, the second best profit, etc. Here we have looked at the whole data set of 99 days. That is, we have not differentiated on the currencies in this table.

Table 2 Distribution of ranks of the models in 99 simulation days

	First	Second	Third	Fourth	Fifth
GA	10	9	13	**49**	18
RRL	**43**	22	11	11	12
Neural	5	8	9	20	**57**
Hybrid 1	17	31	**33**	10	8
Hybrid 2	24	29	**33**	9	4

The RRL stands out also in this comparison. The GA and the Neural Network show poor performance, while the hybrid models show varying levels of success. It can also be noticed that the hybrid models have been ranked relatively rarely among the most poorly performing algorithms.

Closer examination of the models reveals some key features that are noteworthy and are to be considered if the automated systems' risk-return profile is compared with professional traders' estimated deliverables under certain markets and or market conditions. **First**, the models are generally unable to adapt to sudden changes in the market prices. Even the best-performing RRL is prone to overfitting (wrong direction), which makes it slow to adapt to changes in the market.
Second, the choice of the model parameters affects the model's performance greatly. Fine-tuning the parameters to some degree to suit the data would give better results. **Third**, using a combination of the simple sub-models is indeed a good strategy in some markets. The hybrid models are able to pick up successful forecasts and to take advantage of them. **Fourth**, trading possibilities are more limited in the case of the less liquid currencies. Trading between US dollars and Euros is much cheaper, thus allowing for more possibilities.

In conclusion, the main reason for the observed model behavior, particularly the hardly acceptable risk-return profile observed in the EUR/USD market, is coupled with the very exceptional data set. The data reflects the current market turmoil, the co-ordinated central bank intervention and the Fed's rate cut resulting in the US dollar being volatile and touching its weakest level per euro since the 15-nation currency debuted. The most promising of our models might be further improved by feeding less noisy price data or other information than price data into the trading system.

1. Introduction and Motivation

The increased use of algorithms to make complex decisions and place orders in milliseconds has grown in popularity among currency dealers. The demand for algorithmic trading is linked to the growth in the adoption of quantitative research tools and the need to obtain competitive advantage. Europe and Asia progress in catching up with the US on algorithmic trading. The current study was proposed by Nordea Markets.

In this study–consistent with the assignment we were given by Nordea Markets–we examine different formalized strategies of *foreign exchange* (forex, FX) trading with an objective to make successful forecasting of future FX rate changes and to enhance trading profitability. The strategies are algorithms that manage an agent's currency position based on information about specific market indicators and time series data.

We choose the most promising algorithms and simulate them to differentiate on their performance. The choice is made based on an extensive literature review. In addition, we introduce a new hybrid algorithm or model that combines the best virtues of the selected algorithms and provides a comparison of all the simulated algorithms. We intend to demonstrate that weighting of different algorithms' predictive power may be the best solution to obtain competitive advantage among currency dealers.[3] In this work, we use intraday currency data of EURUSD, EURSEK and EURRUB and couple that with technical indicators.

Our paper is organized as follows: Section 2 reviews the main ideas and findings of decision models applied to algorithmic trading and concludes by presenting the algorithm choices made. Section 3 explains the simulation framework and performance measures. Section 4 reports the results of the experiments we carried out on real-life high-frequency FX rate data. We describe results on general and on day-specific basis. Section 5 discusses and summarizes the results obtained. Main ideas and formalism of the implemented algorithms are presented in the Annexes.

[3] Our objectives can be divided into three research questions: 1. What are the relevant algorithms for forecasting FX rate dynamics for the given currencies and thus for best enhancing a speculative agent's profitability or risk-adjusted return (e.g. Sharpe ratio); which data inputs are most relevant? 2. How to implement selected algorithms and differentiate on their performance using appropriate performance measures? 3. How to incorporate the best techniques into a hybrid model that takes advantage of the predictive power of the chosen sub-algorithms. How would the hybrid model perform?

2. Literature Review on Algorithmic Trading

This section describes models and algorithms, or algos, that are applied to algorithmic currency trading. Algos aim at finding optimal decisions about establishing new currency positions. The focus in this section is on identifying the most promising techniques for implementation instead of describing more detailed results.

Our focus is on microstructure-based approaches attempting to infer FX trading strategies on the underlying time series data, which typically contain plenty of noise. Macroeconomic fundamentals based approaches have been demonstrated not to fit well at horizons of less than a year and would, therefore, be even less justified in the intraday context. The readers that are not primarily interested in the theory of the implemented models are advised to move on to the section 2.9 that concludes this section.

2.2 Technical Analysis

Technical analysis refers to examining underlying time series data on FX past prices with aim to predict future price changes and to enhance profitability of trading. In spite of active research on artificial intelligence algorithms over the last two decades, technical analysis has been the most commonly used method in the high-frequency intraday FX trading. Accordingly, a survey by Taylor and Allen (1992) concluded that 90 % of their respondent traders used technical analysis in intraday trading at least to some extent. 60 % of the respondents stated also that they regard the role of technical analysis at least as important as economic fundamentals. Dempster et al. (2001) demonstrate that there is also some reliable evidence that technical analysis has predictive value when forecasting short-term changes in exchange rates. This is supported by growing sentiment of researchers that foreign exchange markets are actually less efficient than has been believed heretofore. Thus, the presumption of exchange rates nature of perfect random walk could be loosened.

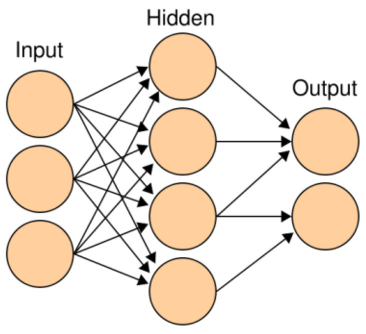
Dempster et al. (2001) examined the relative performance of selected techniques in intraday FX trading. They compared two popular computational learning algorithms, a genetic algorithm (GA) and reinforcement learning (RL), to a pair of simpler techniques of Markov decision process (MDP) and simple heuristic. According to their findings, all the models are capable of producing significant profits with zero transaction costs. With realistic transaction costs, however, none of the models is able to generate significant profits. In another study, Dempster and Jones (2001) studied the systematic use of the popular technical indicators of channel and head-and-shoulders patterns. Dempster and Jones (2000) have also done research with similar data by applying various technical trading rules. These studies do not suggest that any significant profits could be generated with these approaches. Because they focused only on relatively narrow groups of rules, anyhow, the results do not actually indicate that combining technical indicators over broader class could not create more profitable trading strategies.

Trading rules of technical indicators consist of some more complex methods as well as very simple rules, for instance buy and sell signals based on trend-detecting with moving averages. Dempster and Jones (2001) studied trading rules in terms of eight widely used technical indicators, including the price channel breakout, adaptive moving average, relative strength index, stochastics, moving average convergence/divergence, moving average crossover, momentum oscillator, and commodity channel index.

Dempster and Jones (2001) deriveda few interesting conclusions: Firstly, the results indicate that there is useful information in technical indicators that can be used. However, this is somewhat contradictory with the hypothesis of perfectly efficient currency exchange markets. Secondly, using a combination of technical indicators seems to produce better performance than using only individual indicators. Thirdly, the level of transaction costs and the specific market states affect differently the performance of technical indicators and how much they convey useful information. Finally, the authors noticed that more unsophisticated models using technical indicators tend to overly exploit noise in sample data sets. This risk was more visible especially with heuristic rules based on certain trading thresholds.

2.3 Artificial Neural Networks and RelatedAlgorithms

Artificial neural networks (ANNs) have been a popular topic amongst financial engineers as well as other engineers working with control problems from the 1990's onward. Artificial neural networks rely on the analogy to biological neural networks that act as control systems to biological organisms. ANNs consist of interconnected neurons through which information is passed. Typically an ANN consists of two or more layers of neurons, which process the inputs from the input layer all the way to the output layer. A two-layer network is schematized in Figure 1.



Source: en.wikipedia.org/wiki/Artificial_neural_network

Figure 1Artificial neural network

The connections between the neurons are assigned specific weights. Adjusting these weights according to the specific data on hand will allow for the network to produce the right output in different kinds of situations. In fact, an ANN complex enough has the ability to function as an arbitrary function approximation mechanism. Typically the process of applying an ANN is a case of black box modeling, where the weights are adjusted to satisfy a certain quality measure, and the inner workings of the construct are irrelevant for the use of the model. The adjustment can be done using a separate data set as in-sample training, or it can be done online simultaneously with the forecasting or control task.

For the adjustment of neural networks many different approaches are used. A typical division is as follows: First, in supervised learning the network is trained to model the interdependence of known pairs of data in an optimal manner. Second, in unsupervised learning the outputs of the process are not known and the performance of the network is optimized according to a specific cost function, which uses the inputs and outputs as function values. The third category is reinforcement learning (see Section 2.4 on RRL), in which the network acquires its inputs from the environment on the fly and tries to find a policy for its actions that would optimize a certain long-term cost function. Reinforcement learning is often applied to control problems as well as different kinds of sequential decision making problems.

Generally the performance of ANNs depends on some key factors. First of all, the choice of model will depend on the data and the application. As excessively complex models tend to be hard to train, model selection is important. The learning algorithm is an essential part of applying the network. It is easy to come up with an algorithm that can adjust the network to work well with a specific well known data set but getting a network to perform with unseen data will require lots of effort.

Huang et al. (2004) reviewed a wide range of ANN techniques that have been applied to trading in foreign exchange markets, motivated by the ability of these networks to capture the nonlinear nature of FX rates. They claim that "ANNs are particularly well suited to finding accurate solutions in an environment characterized by complex, noisy, irrelevant or partial information".The article represents the multi-layer perceptron as the most widely used network architecture. This is a typical network with several layers where information is only fed in one direction, directly from the input layer to the output layer. Such a network is called a feed-forward ANN.

Another class of ANNs used in related applications is the feedback ANN. <u>Feedback ANNs</u> incorporate cycles, that is, information is passed backwards at some point in the network. This structural feature may <u>potentially make the network slow</u>, as it needs several iterations to come to a solution and information is passed back and forth. These networks are also more difficult to train than feed-forward networks. The article also represents a third class of models, namely competitive ANNs. These models incorporate a mechanism that benchmarks the networks performance with a reference signal.

Huang et al. (2004) also reviewed techniques that can be combined with ANNs to enhance their performance. One of these methods is the genetic algorithm (GA), which is a suitable technique for fine-tuning the parameters of the chosen network: "the flexibility, robustness and simplicity that GA offers render them very attractive". Further methods suggested are wavelet analysis and fuzzy logic.

In appendix 5 a comparison table is presented from Huang et al. (2004). This table represents conclusions from several comparative studies, in which different kinds of ANNs have been compared with other forecasting methods. All studies have looked at the different currencies in the forex market and have used <u>different performance measures</u> to compare the models. In all cases the ANNs have performed better than the random walk approach or the study has lead to mixed results. The comparison with autoregressive methods has yielded varying results: simple ARIMA models seem to be inferior to ANNs, but more complex models are on a par with ANNs. The article summarizes the most important steps in applying an ANN: selection of input parameters, preparing data and the ANN's architecture. It is noted, however, that "there is no formal systematic model building approach". The article presents the supporting techniques described above, and also speculates on neural network ensemble models, which consist of several different networks that bring in each network's good traits. The authors suggest that further research in this area could prove successful.

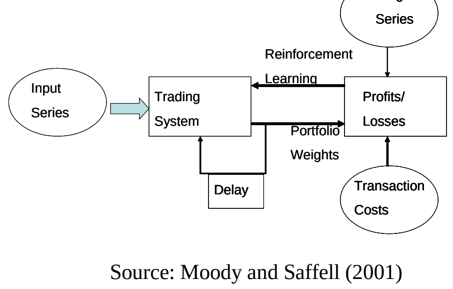
Kamruzzaman and Sarker (2004) have studied a set of learning algorithms intended to train a feed-forward ANN. They tried out the Standard Backpropagation (SBP), Scaled Conjugate Gradient (SCG) and Backpropagation with Bayesian Regularization (BPR). The task was to predict the rate of six currencies against the Australian dollar one step ahead using a set of moving averages of different lengths. They used weekly data ranging over a decade. The performance of the algorithms was measured using five statistical metrics, including normalized mean square error and mean absolute error.

The results show a significant difference in predictive performance in favor of the SCG and BPR techniques over the standard SBP method in all currencies:"the reason of better performance by SCG and BPR algorithm is the improved learning technique which allows them to search efficiently in weight space for solution." Out of the two accurate methods the scaled conjugate gradient based model performs better in terms of the majority of currencies and performance metrics.

2.4 Systems Based on Direct, Evolutionary and Recurrent Reinforcement Learning

This section reviews some important literature on recurrent reinforcement learning (RRL)[4], which is a policy search algorithm that can learn an investment strategy on-line and can be considered as an efficient algorithm. The approach has demonstrated promise and is of key importance for the current project and its implementation and testing task.It can be noted also that the RRL technique typically suffers from problems related to choosing fixed parameters due to complex interdependencies among many of the parameters. Another weakness or lack of robustness as regards results of RRL follows from the fact that understanding of FX market characteristics is still weak. Formalism of RRL is presented in Appendix 3.

The RRL technique was first introduced for training neural network trading systems in 1996 and was soon extended to trading in a FX market (Moody and Saffell 2001). The constituted machine learning technique, called RRL for direct reinforcement (DR), enables simple and elegant problem representation (Figure 2). With RRL, as with some other techniques reviewed in Section 2, trading systems can be optimized by maximizing performance functions, *U( ),* such as profit (return after transaction costs), wealth, utility functions of wealth or risk-adjusted performance ratios like the Sharpe ratio (see Section 3 for the definition of Sharpe Ratio). Investment decision making is considered as a stochastic (or adaptive) control problem. Updating is achieved via gradient ascent in the performance function. The entire set of equations necessary for estimation procedure and for updating the model weights are presented in Moody and Saffel (2001, equations (25)-(30)). The trading system takes historical time series FX price changes as inputs and outputs the preferred position (long or short). The RRL approach differs clearly from dynamic programming and reinforcement algorithms such as TD-learning and Q-learning that aim to estimate a value function for the control problem. RRL tend to produce better trading strategies than Q-learning. The paper provides empirical evidence that by feeding intraday FX rate data into the system, the performance can be very good. In addition, performance criteria such as differential Sharpe ratio and differential downside deviation ratio can be optimized.



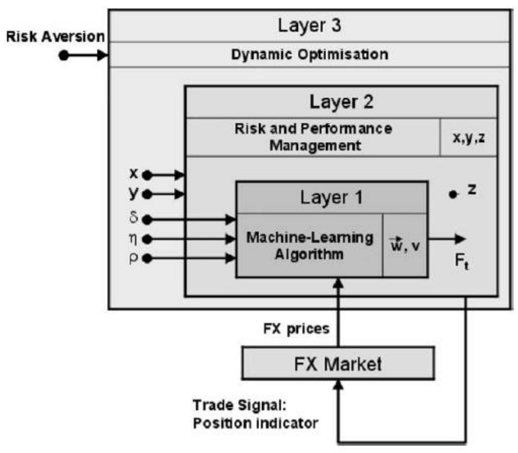
Source: Moody and Saffell (2001)

Figure 2 Trading system based on DR (RRL)

[4] Recurrent means that previous output is fed into the model as part of input. The agent needs to be able to incorporate all relevant elements (transaction costs and market impact) in his/her decision making.

Gold (2003) extends the results of Moody and Saffel (2001)–with additional details where appropriate)–with neural networks trained with evolutionary reinforcement learning (ERL). Gold tests also weight decay variant of the gradient ascent, which updates the weights in the network.[5] Here the model takes a rolling-window approach: The model is initially trained for a certain period and is then applied to a test set of a certain length. After trading period, the training window is advanced by a length of certain points and the whole procedure is repeated. The paper compares single layer network with network having a hidden layer (so-called perceptron). Returns are modeled as a function of current and previous positions and are calculated in terms of midprice of the bid and ask price. Equivalent transaction cost is applied to reflect the loss from position changes. Sharpe ratio is used as a performance measure. Gold (2003) finds, firstly, that trading systems can be effective but that their performance varies for different FX markets (a result that cannot be explained by statistics of those markets) and secondly, that single layer network outperforms the two-layer model, which traditionally can be considered as a more powerful learner. Here the outcome can be explained by the high level of noise. Gold considers these results as too optimistic due to the limitations of the model and because not optimal fixed parameters were found.

The concept of adaptive reinforcement (ARL) is introduced by Dempster et al (2006) who add value to this body of literature with a fully automated trading system application. Here the known machine learning algorithm (by Moody and Saffel 2001) in enriched by a risk management layer and a dynamic optimization layer. The former enables control of the system's strategy risk vs. return trade-off and the latter makes it possible to overcome the problems related to model calibration that are common to RRL. The authors demonstrate that the dynamical optimization layer increases performance of the system. The following schematic illustration (Figure 3) of the automated trading system captures the ideas of LLR by adding risk-management layer and dynamic optimization layer to a known algorithm (Dempster et al 2006). The middle (risk management, 2) layer stops trading if deemed necessary. The top layer (global trading performance optimization, 3) takes account of trade's appetite for risk and tunes system's hyperparameters.



Source: Dempster et al (2006)

Figure 3 Layers of an automated trading system based on ARL

[5]Performance of the network is improved because smaller weights have less tendency to overfit the noise in the data.

Dempster et al (2006) report the use of ARL technique to trade the FX markets based on high-frequency historical data set performs very well; the ARL based system was able to earn over 26 % per annum while avoiding large draw-downs. Order book and order flow data in this context may have potential for enhancing performance of ARL even further. Since the markets are becoming more efficient, it is getting more difficult to earn profits with RRL related techniques using price data. Therefore, trading system performance could be improved if order book and order flow data were fed into the ARL type trading systems (Bates et al 2003 demonstrate promise with ERL).

2.5 Genetic Algorithms and Genetic Programming

Genetic or evolutionary algorithms (GAs), based on the Darwinian theories of natural selection and survival,are search procedures that aim to find near-optimal solutions to multi-extremal problems. Figure 4 illustrates the main idea of GA.

The GA approach typically

- Uses a population of structures rather than a direction or single structure.
- Works with a high level of global sampling in order to converge to a global optimum.
- Represents the solution space as a finite number of strings of binary digits.
- Needs a means of evaluating the fitness of the constituents of the solution space and an initial population of suitable solutions.
- Can provide the functional solution in analytical form.
- Conducts an efficient exploration of the search space and is suitable for supporting on-line trading.

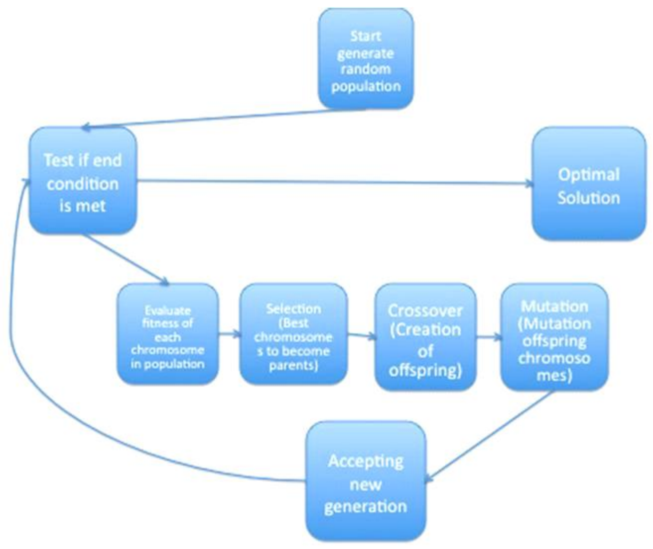

Start generate random population
Test if end condition is met
Optimal Solution
Evaluate fitness of each chromosome in population
Selection (Best chromosomes to become parents)
Crossover (Creation of offspring)
Mutation (Mutation offspring chromosomes)
Accepting new generation

Figure 4Representation of a Genetic Algorithm

The GA (GP) approach can select the relevant variables that need to be entered into the model, which is a feature that may make this approach more attractive than common autoregressive models such as ARCH and GARCH. Iterative systems based on GA can be considered as a relevant approach to nonlinear modeling of time series. Since this technique has recently shown promising results in FX time-series prediction, it is highly relevant in view of the forecasting and strategy search task of the current project (see also Appendix 2 on implementation).

Alvarez-Diaz and Alvarez (2003) evaluate with weekly FX rate time series data (six currencies, all against US dollar), whether nonlinear modeling with GAs outperforms the prediction power of the random walk model. They find that in all cases, the mathematical models found by this functional search procedure forecast somewhat better than the random walk hypothesis. A linear relationship with the most recent past value dominates these models heavily but contributions to the total forecasting performance from nonlinear terms are small. As consistent with earlier results, nonlinearities of FX rates cannot be exploited to considerably improve forecasting.

Santini and Tettamanzi (2000) report on application of a “pure” GA to forecasting daily closing values of the Dow Jones in a given period of time. For building of the initial population, they use a technique typical to GAs which generates each expression and individual independently. A probability is then applied to each element of the function set and to the constants of the terminal set, and each expression is then build recursively. What is important, however, is the fitness function for evaluating the performance of the potential predictors; a function measuring the distance between predicted and actual prices (see Santini and Tettamanzi, 3.2 for formalism). The same data set is used for training as well as calculating the fitness and for obtaining the prediction itself. The adopted algorithm was robust regarding the parameter settings and very good results were obtained by running the algorithm an adequate number of rounds, and by taking the highest fitness individual.

Dempster and Jones (2000) proposed a framework for systematic construction of a trading system that trades profitably. They rely on on-going adaptation of the system based on GAs (Genetic Programming). The aim is to develop a system and to imitate the behavior of a technical trader who adapts to the market. Such adaptation is required when the applied trading rules become loss-making. Then the question is that can such a technical trader be consistently profitable and is it better to adapt to the market or remain static. The starting point of the paper is the rule portfolio based on GA. The rules are initially based on combinations of different technical indicators at different frequencies and lags. The GA technique is used regularly (on out-of-sample data) to provide new rules and the rule portfolio is rebalanced by means of a feedback system so that there are two levels of adaptivity. The paper finds that individual indicators are generally loss-making but that the best rule found for the developed system is slightly but significantly profit making. The six technical indicators, simple moving average cross-over (SMA), adaptive moving averages (AMA), price channel breakout (PCB), the stochastic, the relative strength index (RSI) and the commodity channel index (CCI), are calculated on a set of USDGBP data aggregated to various intraday frequencies.

2.6 Support Vector Machines

Using Support Vector Machines (SVM) in predicting FX-markets has been recently researched as an alternative to the ARIMA and ANN approaches. SVMs were suggested by Vapnik (1998) for efficient classification of data: “SVMs perform by nonlinearly mapping the input data into a high dimensional feature space by means of a kernel function and then do the linear regression in the transformed space.” (Kamruzzaman, Sarker and Ahmad, 2003). A SVM is defined by a regularization parameter C, a ε-insensitive loss function and the type of the kernel function. The regularization parameter C is used to balance the trade-off between model fit and generalization ability, the loss function defines the minimum scale of errors that should be penalized and the kernel function – possibly non-linear–performs the transformation of the data into a high dimensional feature space, in which a linear classification can be performed.

Kamruzzaman, Sarker and Ahmad (2003) studied SVM-based models in forecasting forex rates and were expecting to find results that outperformed the ARIMA model. They tried out different kernel functions and model parameters on the Australian forex market and measured the methods on different performance metrics. The polynomial kernel was found to do well in terms of the prediction error (MNSE, AME), and the radial basis and polynomial kernels were successful at predicting trends. “In general radial basis and polynomial kernel appeared to be a better choice…”. Linear and spline kernel functions were found to be less effective. The choice of the kernel function should, however, always be done with the individual time series characteristics in mind. Combining and mixing different kernel functions is presented as a promising study subject.

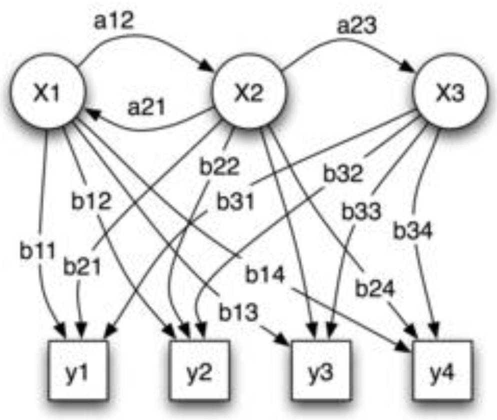
Pai and Lin (2005) combined the ARIMA and the SVM methods to predict daily closing stock prices of certain companies. They forecasted one-step ahead and used several measures of forecasting accuracy (Mean Absolute Error, Mean Square Error, Mean Absolute Percent Error, Root Mean Square Error). The forecasting models were combined in two ways in order to reduce forecasting errors, which a hybrid model should theoretically achieve. First, the combination model simply used the predictions of the separate ARIMA and SVM predictions and used the sum as the new prediction. In the actual hybrid model ARIMA was used to prefilter the linearities of the time series and the SVM dealt with the error terms of the linear model. They observed that the hybrid model clearly outperformed the predictions of the individual ARIMA and SVM models as well as that of the combined model.

Ince and Trafalis (2006) studied hybrid models in forecasting several important exchange rates. The problem was twofold: They tested several methods – including ARIMA – for classifying and selecting model inputs. The selected inputs were then fed into selected SVM and ANN models, respectively. They found that the SVM method outperformed (MSE, MAE) the estimated ANN (specifically Multilayer Perceptron) for each set of selected inputs. The writers suppose this is due to the convexity of the optimization problem of the SVM, for which it is easy to find a global optimum. In contrast, the optimization problem of the ANN is non-convex. Additionally, they find that the ARIMA input selection method beats the alternatives when used in conjunction with the SVM, whereas the vector autoregressive input selection method is most suitable alongside the ANN forecasting method.

2.7 Other Related Methods

2.7.1 Hidden Markov Experts

Financial time series can be thought of as consisting of a Markov process, where the next value of the data series depends on the previous value. These transitions between stages are governed by specific probabilities, which determine the stochastic process. In a Hidden Markov Model (HMM) the underlying Markov model remains disguised for the observer, and the observer must rely on secondary observations that are dependent on the actual states through certain probabilities. The model is schematized in Figure 5.



Source: en.wikipedia.org/wiki/Hidden_Markov_model

Figure 5 Hidden Markov Model, where X represents states, y possible observations, a state transition probabilities and b output probabilities

Shi and Weigend (1997) propose a further step in refining the HMM. They introduce the Hidden Markov Experts model, which adds a mechanism to the HMM that allows for adaptation to the changing time series characteristics of the underlying data – the so-called regime switch. The model can address the common problem of non-stationarity and time varying variance. The specific experts of the model can each react to different conditions in the target market, each condition being characterized by current volatility. The model has then one expert for trading in a low volatility environment, and another expert for trading in an extremely volatile period.

Shi and Weigend tested their model on two real life data sets, the DEM/USD forex rate and daily S&P500 stock market data. For the forex rate case they used five lagged values as inputs to predict the next value. They compared the performance of the model with the performances of an ANN and a linear autoregressive model, and found the Hidden Markov Experts model to outperform both competitors. They came to similar results with the stock market data.

2.7.2 ARMA-Model and its Derivatives

Autoregressive Moving Average –models establish a linear dependence between current and past points of time-series data, and they can be used to explore and predict the temporal behavior of a process. The general ARMA-model takes the form

$$y_t = \delta + \phi_1 y_{t-1} + \phi_2 y_{t-2} + \ldots + \phi_p y_{t-p} + \varepsilon_t + \theta_1 \varepsilon_{t-1} + \theta_2 \varepsilon_{2-1} + \ldots + \theta_q \varepsilon_{t-q},$$

where y_i and ε_i represent the lagged state and error terms, p and q stand for the number of lags used and ϕ_i and θ_i are the weights – model parameters – that are estimated.

The model can be expanded by incorporating an integrative component to the model. The resulting ARIMA model can in some cases better fit the data on hand. A further modification is the SARIMA model, where S stands for seasonal effects that occur sequentially rather than only in the immediate time steps before the present. These models can adjust to weekly or monthly cycles, for instance. Perhaps the most interesting sub-class of models is the ARMAX model family, which allows the use of exogenous time-series data to bring in additional inputs along with the data that is being predicted. In the case of FX-trading this data can be interest rate data or time-series data of some other market indicator.

These parametric methods try to capture the linear structures of time series data. However, foreign exchange rates behave inherently in a non-linear fashion, and autoregressive moving average models have serious problems in modeling this behavior. These methods have been scrutinized for these inefficiencies. Diabold et al. (1994), Kilian and Taylor (2001) and Trapletti et al. (2002) have found that even the random walk model has the potential to outperform these techniques. To overcome these problems the basic models have been extended to take into account the non-linear structures and the changing characteristics of the time series. The Generalized Autoregressive Conditional Heteroscedasticity (GARCH) model is an example of such a model.

Meade (2002) conducted a comprehensive study of the performance of a linear model against a set of non-linear methods in predicting FX rates. The research was conducted using data of different frequencies ranging from half-hourly to daily closing quotes on different currencies. Meade compared a linear GARCH and four non-linear methods and used several metrics to evaluate their performance (RMSE, Peseran and Timmerman test etc.). He found that the different accuracy measures showed varying capabilities in judging the forecasting methods. In general, the RMSE and Peseran&Timmerman showed good accuracy in forecasting some of the high frequency rates. He found no performance difference between linear and non-linear models, although the different approaches to measuring predictive accuracy gave inconsistent results.

Neely and Weller (2003) used half-hourly bid and ask quotes of foreign exchange markets to make predictions. The applied algorithms were a genetic program with three data inputs and a linear model with a maximum of ten lagged variables. They found no significant predictive power in either model, although they were able to see some structure in the data. They applied a filter in the decision making process to accommodate for the transaction cost and for insecure predictions. Hence this study suggests that linear models have a poor chance at producing profitable forecasts about the forex market.

2.8 Input Data Types

In this section, the relevant literature on input data selection for different models is reviewed. This is a particularly relevant topic due to the specific characteristics of FX rate time series and the challenges to predict them. The rates are considered to be inherently noisy, non-stationary and deterministically chaotic (Deboeck, 1994; Yaser & Atiya, 1996). As a consequence, complete information does not exist that could be accurately used to model the dependency between the future values and the past behavior of time series. Kamruzzaman and Sarker (2003) present the general assumption that the historical data incorporates all those behavior and, thus, the historical data itself is often the major input when predicting the future values.

2.8.1 Classification of Input Data Variables

Wei et al. (2004) have reviewed the use of artificial neural networks (ANN) in forecasting exchange rates. With that model group, the selection of input data variables and the preparation of data are especially underlined. Thus, the output performance can be maximized only by assuring that the input variables represent adequately the domain knowledge.

Inputs can be divided in fundamental inputs and technical inputs. Some fundamental inputs are, for example, price index, foreign reserve, GDP, export and import volume, interest rate, etc. Technical inputs are, among others, the delayed time series data, moving average, relative strength index, etc. In addition to fundamental and technical inputs, it is also possible to use individual forecast results as inputs when ANNs are used as combined forecasting tool. (Wei et al., 2004)

Another classification for input data is the division to multivariate and univariate inputs: Univariate inputs use data directly from the time series that is being forecast. Thus, their forecasting potential is based on the predictive capabilities of the time series itself and is related to technical analysis. Multivariate inputs use also data that is not based on the time series itself and they are used with fundamental analysis. Multivariate inputs in the context of foreign exchange rates are based on economics and finance theory. Walczak et al. state that it is necessary to use multivariate inputs to forecast exchange rates. However, according to Wei et al., most inputs in neural networks models for exchange rate prediction are univariate.

2.8.2 Selected Input Data Variables in Literature

In the literature, numerous types of data are used as input variables when forecasting exchange rate development. In univariate time series forecasting, the inputs are the lagged observations of the time series itself. Input patterns are composed of moving windows along the time series. The number of lag periods and the length of lag windows are to be decided, and there are mixed methods available how to determine those optimally. Wei et al. (2004) propose that a small number of lag periods can best disclose the specific features that are embedded in the data in ANN modeling. They present Autocorrelation Criterion method as a practical method to be employed in univariate input selection.

Kamruzzaman and Sarker (2003) state that exchange rates exhibit their own trend, cycle, season and irregularity. In their own study, they used time delay moving averages of one week, two weeks, one month, one quarter, half year, and the closing rate of the previous week to predict future values. The advantage of using moving averages is that it helps to smooth out some irregular characteristics that are present between different market days. In the study, they time series of six different currencies against the target currency, Australian dollar. Other currencies were US Dollar (USD), Great British Pound (GBP), Singapore Dollar (SGD), New Zealand Dollar (NZD) and Swiss Franc (CHF).

El Shazly et al. (1997) have used in their study following multivariate inputs: the one month Euro rate on US dollar deposit, the one month Euro rate on the foreign currency deposit, the spot exchange rate, and the one month forward premium on the foreign currency. In another study, Shazly et al. (1999) use additional inputs: the 90-day Euro deposit rate on the US dollar (INT US); the 90-day Euro deposit rate on the British pound (INTBP), German mark (INTDM), Japanese yen (INTJY), and the Swiss franc (INTSF); the spot exchange rate of the foreign currency: SBP, SDM, SJY, and SSF expressed in direct form; the 90-day forward exchange rate on the foreign currency: FDBP, FDDM, FDJY, FDSF; and the 90-day future exchange rate on the foreign currency: FTBP, FTDM, FTJY, FTSF. The reason behind the selection of input variables mentioned above is based on the Interest Rate Parity. According to this theory, "forward exchange rates reflect relative interest rates on default risk-free instruments denominated in alternative currencies". Consequently, markets expect that the currencies of countries with high interest rates are expected to depreciate in the longer run, and vice versa. (Wei et al., 2004)

Leung et al. (2000) have used MUIP relationship to specify multivariate inputs. These include the natural logarithm of the exchange rate, the logarithm of the nominal short-term interest rate, expected price inflation rate, the logarithm of the price level, and the ratio of current account to nominal GDP for the domestic economy. Tenti (1996) make use of the following input variables: the compound returns of the last selected n periods, the running standard deviation of the k last selected periods, and technical indicators such as the average directional movement index (ADX), trend movement index TMI), rate of change (ROC), and Ehlers leading indicator (ELI). Lisi and Schiavo (1999) rely both on the variate and multivariate information as besides using the past observation of the series itself, they have also used an “auxiliary” variable chosen among other possible pairs of currency time series.

Bates et al. (2003) have studied indicators based on order flow and order book data. They explored whether pattern recognition techniques created originally for computational learning could be useful in predicting exchange rates. The preliminary results of their study indicate that using order flow and order book data usually overperforms trading based only on technical signals. Evans and Lyons (2001) have also found that order flow is an important factor to explain changes in nominal exchange rates over periods of about one day.

As macro theories of economics and finance suggest, there are several factors affecting the development of different currency pairs and their exchange rates. Accordingly, the set of possible input variables is vast and their explanatory power is critical for the performance of the prediction output. Walczak and Cerpa (1999) present a stepwise method to determine relevant multivariate inputs especially for ANNs: First, standard knowledge should be acquired in terms of as many explanatory variables as possible. This is to guarantee that the information provides all relevant domain criteria to be used in modeling. Second, the variables increasing noise to models should be eliminated. According to Smith (1993), the best set of input variables to neural networks is predictive but the variables are not correlated. The correlation of variables decreases the prediction performance of model by creating bias to results. (Wei et al., 2004)

Ince and Trafalis (2006) suggest alternative method for input selection process. They illustrate how ARIMA models co-integration analysis with vector autoregressive (VAR) model can be used to determine the number of inputs for ANN and support vector machine (SVR) models.

Wei et al. (2004) point out that the advantages of multivariate inputs are present especially in long term forecasting because they could partly validate the development of the trend of foreign exchange rates. Other studies (Meese & Rogoff, 1983) have proved that macroeconomic variables, such as balance of payments, interest rate differentials, inflation differences etc., have little predictive power even on timescales of months. Using several multivariate input variables requires naturally more data and time for model construction and they strengthen the credibility of forecasting by increasing economic explanatory power mainly on the long term. Bates et al. (2003) continue that the interest in predicting future exchange rates has, accordingly, moved from macro economic factors towards more microstructure-based approaches. Hence, it is also relevant to consider carefully what kind of need there is for different multivariate input variables in our prediction modeling project.

2.8.3 Preparation of Input Data Series

Wei et al. (2004) present potential ways how the input data could be processed and prepared for use in especially neural network models. They stress its importance for the learning speed of networks and the quality of predictions. They offer the following questions as a starting point for data preparation:

1. Is sufficient data available, and does this data contain the correct information?
2. Does the available data cover the range of the variables concerned as completely as possible?
3. Are there borderline cases that are not covered by the data?
4. Does the data contain irrelevant information?
5. Are there transformations or combinations of variables (e.g. ratios) that describe the problem more effectively than the individual variables themselves?

The next phase is to make required transformations into the most appropriate form for the model in question. The measures could be, for example, normalization, scaling linearly to some specific range, or using logarithm transformation to stabilize the series. However, there is no clear consensus whether and when normalization or other transformations should be used. Their effect on the learning ability of the models or the accuracy of predictions is inconsistent in existing literature on previous studies. Still, Wei et al. (2004) conclude the polemic by stating that "it is nevertheless inadvisable not to normalize data when using multivariate inputs". When normalization is done, variables with different ranges of values have similar kind of significance for the modeling process.

In general with ANNs, the used data is divided to the training set and to the test set. The training set is for teaching the model and the test is used to evaluate the quality of the predictions and forecasting ability. Third set, the validation set, is sometimes used to avoid the problem of overfitting or to determine the stopping point in the training process. Different sets can be chosen based on the modeling needs and there is no any specific guideline how it should be done (Wei et al., 2004). Yao et al. (2000) suggest a general principle to divide the data set into training, validation and testing set containing 70%, 20% and 10% respectively.

Sample size is also one factor affecting the forecasting ability of ANNs. It is often said that larger samples are more optimal to be used when teaching the network. However, some studies indicate that it is not always necessary to use a large data set for good performance (Wei et al., 2004). Huang et al. (2002) highlight an interesting idea for determining the optimal quantity of training data: As the behavior of exchange rates is evolving, the data series has a series of change points that could be used. They continue that "these points divide data into several homogeneous groups that take heterogeneous characteristics from each other".

2.8.4 Input Data Variables Used in the Project

Considering the use of input data variables in the project, important decisions have been made. After reviewing the literature and requesting information and data from Nordea, we decided to concentrate on the FX rates of the three target currency pairs – EUR/USD, EUR/SEK, EUR/RUB – as univariate inputs. It is justified to close out most multivariate inputs because the objective of the project is to predict changes in FX rates on a very short term basis and to model decision making in order to profit by exploiting these changes and the bid-ask spread. The true value of using several multivariate inputs would emerge when forecasting a longer term development. The actual values used in the simulations were the averages of bid and ask prices of currency time series and the average spreads for the whole simulated days.

2.9 Conclusion

Although there is a significant body of literature focusing on algorithmic FX trading, there are only a few algorithms that are adequately well-documented in view of our implementing purpose and that enable a relatively simple implementation process. At the same time – as consistent with the project task we were given, we give priority to algorithms that have already demonstrated promise. We therefore choose to model trading systems based on GA, RRL and ANN. All these approaches return results quickly and, thus, if implemented efficiently, enable the use of real-life data and the support of FX position management on-line. We focus on intraday data because by doing so we wish to obtain results that are closer to reality.

We note that Hidden Markov Experts and Support Vector Machine discussed above could add value if successfully applied to currency trading, but we chose to close them out due to the challenges we faced while constructing them. We conclude that standard linear time series models for regression (e.g. ARMA, its derivatives and GARCH) are not applied because such models' predictive performance would be inadequate. One of the reasons for the models poor performance is the heteroskedasticity of the studied high frequency FX data.

Based on the above review, the general idea is to construct an efficient trading system and create a systematic adaptation process for that system. Path dependency of trading is a key feature since any system state reflects market conditions and held positions. Investment performance depends on series of interdependent decisions. Modern portfolio theory suggests that one should optimize risk-adjusted investment returns, e.g. the Sharpe ratio, instead of simple profit or other economic utility. There is naturally also a need to properly account for transaction costs.[6]

Since the FX markets are becoming continuously more efficient, it is more difficult to perform well with RRL related techniques. Therefore, price data may be insufficient at times, and order book and order flow based trading systems should be considered as complementary aids.

[6] Concept of transaction cost: Moody and Saffel (2001) define market frictions to cover taxes and a variety of transaction costs, such as commissions, bid/ask spreads, price slippage and market impact. In addition, Dempster and Jones (2001) define the concept of slippage as a penalty resulting from a difference between actual price and previously quoted mid-price, which includes both time delay and transaction cost.

3. Simulation and Performance Criteria

As concluded in the previous section, we construct the GA, RRL and ANN models and simulate their performance. In addition, two hybrid models are constructed and simulated. The basic ideas and formalism of these algorithms are presented in the Appendices. Testing of the algorithms was done by running the algorithms with real-life FX rate data using EURUSD, EURSEK and EURRUB currency time series. The purpose of the simulations was to test the performance of the selected trading algorithms. This section presents the principles and performance criteria of simulations we have chosen to use in this work.

We chose to run simulations on a daily basis and defined one simulation day to start at 17:01 and end the following day at 17:00. This enabled us to include a long trading period each day before trading hours, starting at 10:00 in the morning. The algorithms were then allowed to trade between 10:00 and 16:59. Positions were closed at 17:00, consistently with the original description of theassignment. A trading decision was given each minute, as all of the data followed the same frequency. This resulted in a daily data set of 420 data points. We had a total of 33 days of data from all three currencies to be fed to the algorithms. Hence we simulated a total of 99 days. In preparation for the simulation event each algorithm was implemented in a single MATLAB m-file, which returned sell/buy decisions based on that algorithm. The trading data was collected then with a different m-file. The maximum value of currency positions was limited to EUR 20 million according to the assignment guideline. Thus, all the decisions could not always be realized when the limits were reached.

The following data was collected:

- Daily profit
- Sharpe ratio
- Number of transactions
- Decisions

In order to get a clear picture of the differences in the performances of the algorithms, we decided to measure algorithm performance with multiple criteria. We see that one of the most important criterions is naturally profit. However, we see that this shouldnot be the only criterion for success, as it gives too much weight to pure luck and ignores the risk. As is consistent with the above literature references, we decided to use the Sharpe ratio as a risk-aware profitability measure for the algorithms.

The theoretical Sharpe ratio is given as

$$S = \frac{E\left[R - R_f\right]}{\sqrt{\operatorname{var}\left(R - R_f\right)}}$$

where R is the risky profit and R_f is a risk free profit. In the case of constant R_f and with the given data we use the following definition of the Sharpe ratio:

$$S = \frac{\bar{R}}{\sqrt{\operatorname{var}(R)}}, \quad \bar{R} = \frac{1}{N}\sum_{i=1}^{N} R_i$$

Sharpe ratio measures the return compared to risk. The asset with a higher Sharpe ratio gives more return for the same risk. Therefore, assets with higher Sharpe ratio are usually preferred.

The number of transactions was decided to be one measure for performance because of the transaction cost coupled with each transaction, and because it gives important knowledge about the way the algorithms trade. Even if this kind of a measure is not strictly a measure of performance, it could still give some general clue about the functioning of the different algorithms.

For ranking purposes we invented a measure of our own because none of the above measures were suitable as such. For the new measure, we decided that the criteria behind the ranking should be the sum of all the profits generated by algorithms under the test period. However, we decided we wanted more concrete knowledge about daily performance. Hence we ranked the algorithms on a daily basis according to their daily profits.

To summarize those daily ranks we decided to use a measure called ‘Top Gun’. The Top Gun ratio is the sum of the first and the second ranked positions during the test period divided by the length of the test period. This measure aims at giving summarized knowledge about how well an algorithm performs each day in a daily competition. Thus, this measure gives only information about the rank order and not about how much better an algorithm has been. For instance, a difference of 1€ in profits is equal to difference of 1000€, as the order is the only relevant factor here.

4. Analysis and Results

In this section, we look into the simulation results obtained during the simulation event. The section is divided into two parts: First we discuss the general results obtained from the 33 simulated days with the three currencies. Then we take a closer look at the selected days that we find to reveal interesting characteristics of the models. We hope to showcase unique and interesting features of the models and to gain a better understanding of how the models should be applied in different conditions.

As for trading strategy, all our models assume that agents trade fixed position sizes in a single exchange rate and are able to draw on fixed credit line from which they can borrow in home or foreign currency during the trading period. It is important to note that this, if we have interpreted correctly, is also standard way of proceeding in the markets. However, we note that by using this approach, the number of trades generated per day is higher than would be realistic. When the position is to be closed, the cumulated value is converted at the new exchange rate and profit or loss is credited/debited from the account. An important difference between the implemented models' strategies is that agents can take long, neutral or short positions of constant magnitude in GA,RRL and hybrid models but cannot remain neutral in ANN. We note that behavioral change can be anticipated when near the closing of daily trading period and thus the close of positions approaches. For this reason, time value of money (and the related risks) could beneficially be incorporated in this context but we leave that for future research. In addition, we gave special attention to the fact that the constructed trading systems need to be comparable. Therefore, there are no such features that would lead to incoherence of the output.

4.1 Overview on the Performance of Models

In this section we look into the results obtained from the total data set, specifically the 33-day time period. The data is typically presented for each currency independently making it possible to assess the models' performance for each currency independently.

We start, however, with a short analysis of the models' performance on the whole data set. To investigate how the performance of the different models varies, we will look at the profits the models have generated each day. We use ANOVA to test whether the differences between the means of the samples could be due to random sampling error alone.[7] Figure 6 provides a Box-and-Whisker plot in the case that data from different currencies is combined. By looking at the diagram we can see a slight difference in the mean values of the daily profits for the four different models. The RRL and the hybrid models seem to perform slightly better than the other two. And indeed, the ANOVA test rejects the null-hypothesis about the common mean using any conventional confidence interval.

[7] We also used the non-parametric Kruskal-Wallis method for testing (equality of population medians among groups) but the differences between that and the ANOVA are to be considered as minor.
See http://en.wikipedia.org/wiki/Kruskal-Wallis_one-way_analysis_of_variance for more about this method.

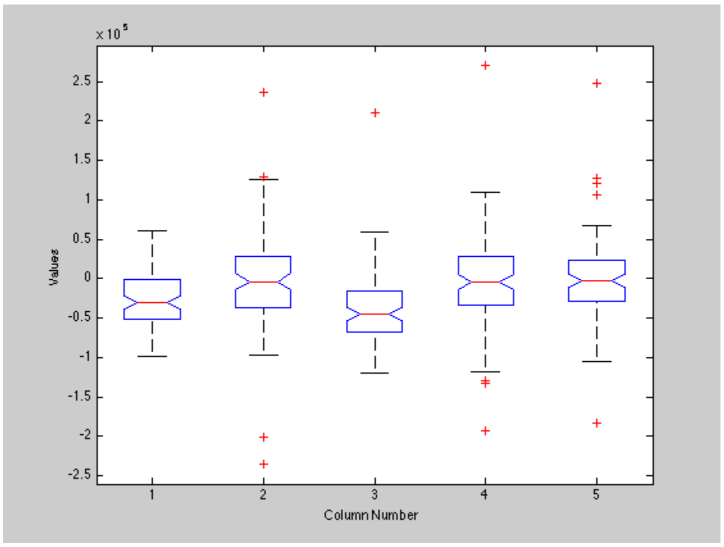


Figure 6 Box-and-Whisker plot for all currencies
(1=GA, 2=RRL, 3=Neural Network, 4=Hybrid 1, 5=Hybrid 2)

The red stripes represent the mean values, the top and the bottom of the boxes represent the value of the highest and lowest quartiles, the outreaching 'whiskers' indicate the minimum and maximum values and the individual ticks represent possible outliers.

Figure 7 provides some key indicators from the same data set. The chartleft-hand side depicts the profits the different models have generated over the 33-day period for all the selected currencies. The genetic algorithm and the neural network have performed poorly showing heavy losses. The first hybrid model has also done poorly but has achieved only slightly negative profit. The RRL model and the second hybrid model with the integrated technical indicators have achieved a positive profit the RRL being slightly more profitable. The Sharpe ratio, depicted in the chart on the right-hand side, gives a result that is consistent with the absolute profits. The ratio reduces the performance gap between the two worst performing models. The Top Gun measure ranks the RRL model as the best model leaving the two hybrids as the second and the third.

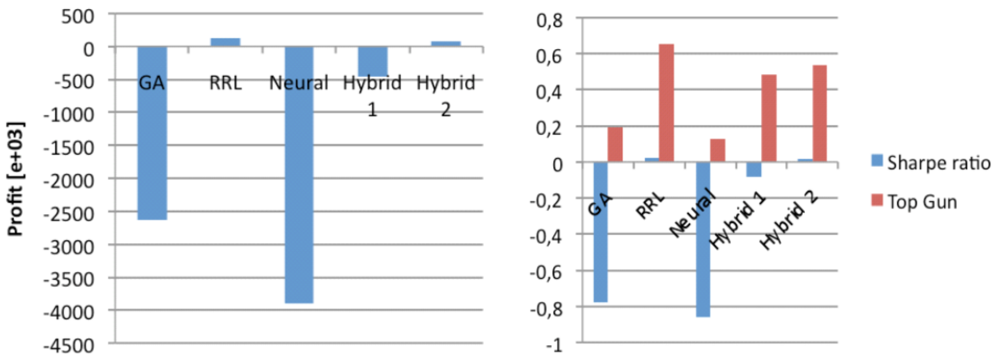


Figure 7 Performance of the models over the 33-day period from all currencies combined
(Profits on the left, Sharpe ratios and Top Gun measure on the right)

4.1.1 Analysis of Wins

Next we show how the models have performed when looking at the currencies separately. We expect to find more variance in the profits with the divided data sets as each currency has its own characteristics. Indeed, the currencies were deliberately chosen to represent different markets.

Figure 8 dissects how the models perform in respect to the different currencies in terms of simple profit. The most important result is that the worst performing models do not generate heavy losses when trading with US dollars. The GA and the ANN make the heaviest losses with the Russian ruble, and about half of those losses using the Swedish krona. The RRL does the opposite, as it achieves a small profit using the ruble and the krona, but lands on the negative using the USD. Both hybrid models are making profit on the US dollar but make small losses with the ruble and the krona. Consequently, all the currencies have their typical own characteristics. It can be also noticed that the most expensive currency to trade is the ruble (the largest spread) whereas the US dollar has lowest trading costs. Regarding the RRL, its trading activity is relatively low and therefore the different sizes of spreads are not fully reflected in its results.

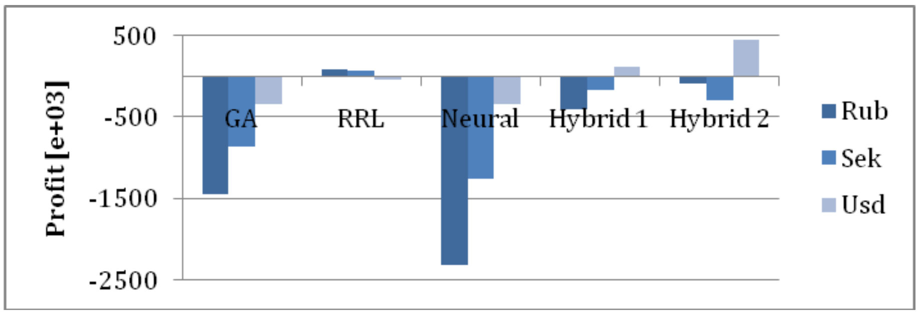


Figure 8 Profits of the models in terms of different currencies

Analysis on Sharpe ratios (Figure 9) gives a similar result. Again, this measure reduces the differences between the models but gives mostly the same ranking order performance-wise. Most notably the Sharpe ratio doesnot award the second hybrid model as generously in case of the US dollar indicating that the high profit is achieved with a relatively high risk.

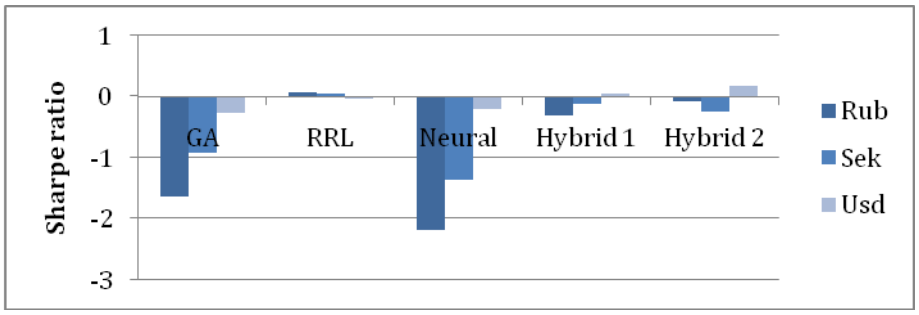


Figure 9 Sharpe ratios of the models in terms of different currencies

The Top Gun measures in Figure 10 again identify the RRL and the two hybrids as the three most efficient models. The RRL seems to perform best using the ruble. Using the two other currencies, the three are more on a par. The GA and the ANN show some success with the ruble and the krona but its performance is very weak when trading with the US dollar.

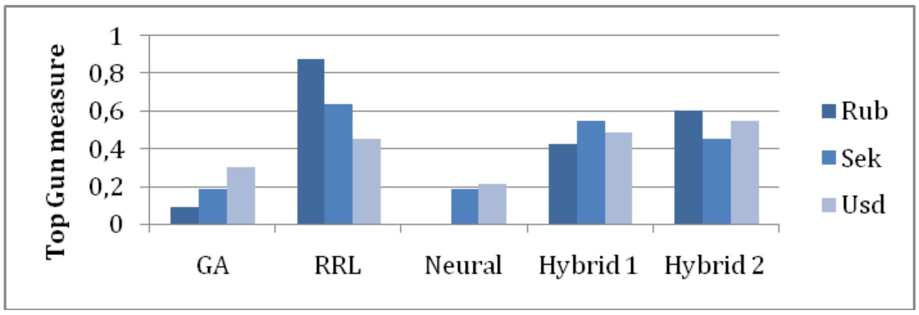


Figure 10 Top Gun measures of the models in terms of different currencies

Generally the RRL model seems to perform much better that the two other individual models. The hybrids are also doing well and are probably basing their decisions largely on the RRL's decisions. They are also sometimes able to perform better than the RRL, which indicates that they are in some cases able to take at least some advantage of the other models' strengths outperforming the RRL.

It is also worth noting that the RRL seems to lose some of its lead as the volatility of the liquidity of the currency increases. All the measures show a clear decline in performance as the currency changes from the ruble to the krona and then to the US dollar. The GA shows a reversed behavior.

4.1.2 Analysis of Daily Trading Volumes

Next we will analyze the differences in how the models react to the changes in the marketplace. Figure 11 and Figure 12 represent the daily profits for the different currencies as well as the number of transactions each model has decided to conduct during those days.

It is obvious that the ANN makes the most transactions because of its design, and thus it ends up doing poorly because of the high transactions costs. The genetic algorithm trades a lot less than the ANN but is still doing many more transactions than the RRL and the hybrids.

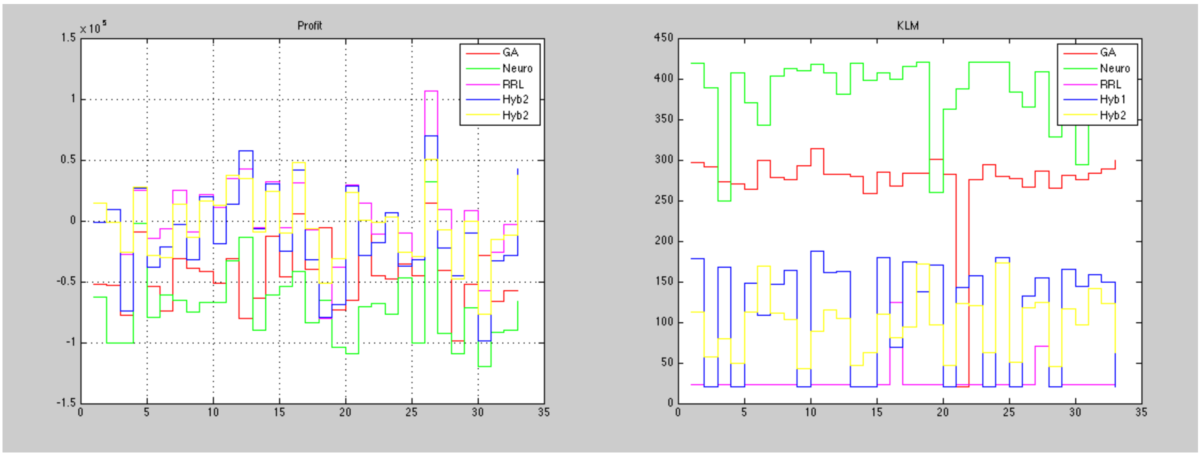


Figure 11 Daily trading data for the Russian ruble(profit on the left, number of transactions on the right)

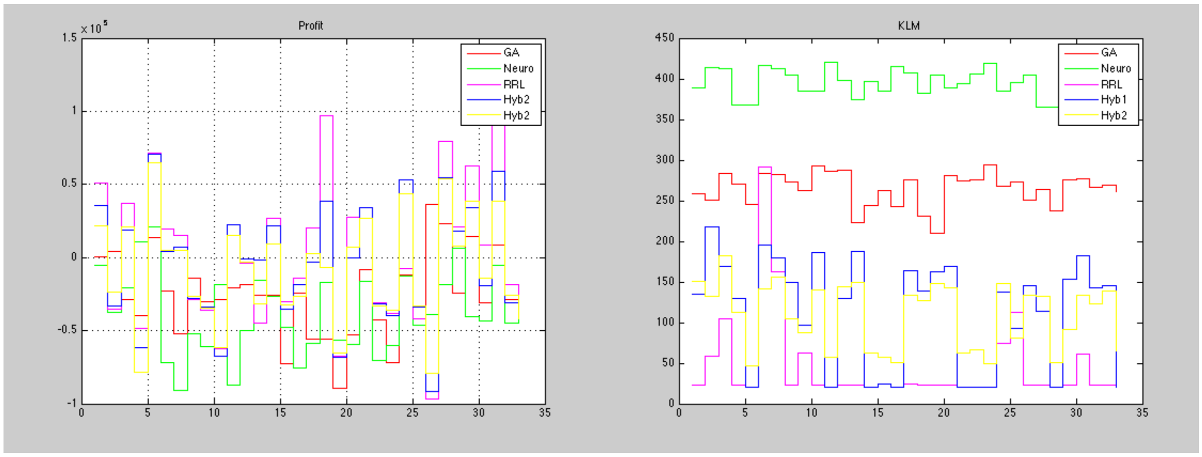


Figure 12 Daily trading data for the Swedish krona (profit on the left, number of transactions on the right)

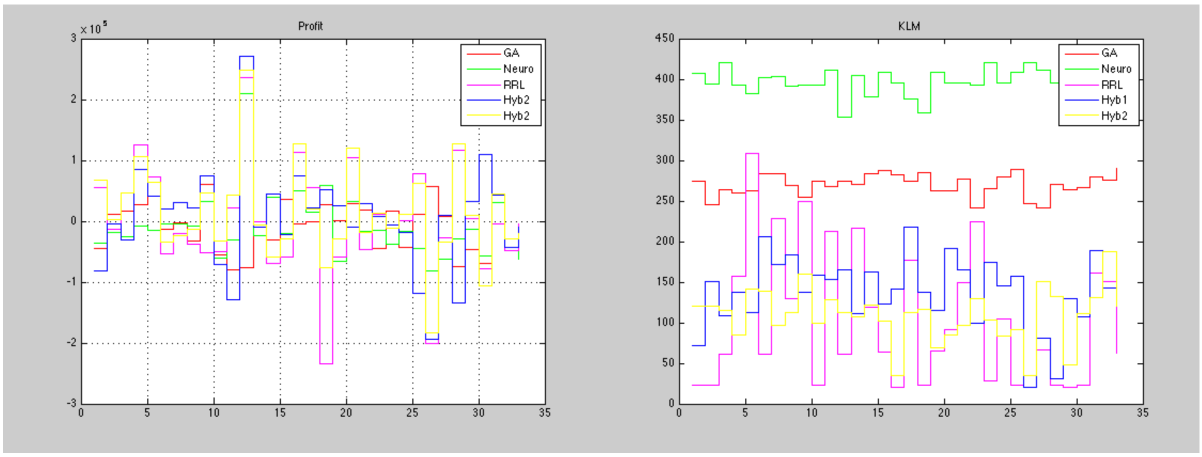


Figure 13 Daily trading data for the US dollar (profit on the left, number of transactions on the right)

The RRL model is trading a lot less with the ruble and the krona than it is when using the US dollar. With the dollar its behavior is much more erratic as can be seen from the pictures. The hybrid models do generally make more transactions than the RRL, but in case of the dollar the differences become less obvious. The second hybrid model shows less fluctuation in the number of transactions but does act on the same moments.

4.1.3 Statistical Significance of the Differences in Mean Profits

Next we will confirm analytically whether the differences seen in the models' behavior really are statistically significant. We test the null hypothesis that the means of the profits of the models are the same for each currency separately. Figure 14 and Figure 15 show the Box-and-Whisker plots for all the currency data sets along with the ANOVA P-values (see Figure 6 for detailed information on the interpretation of the plots).

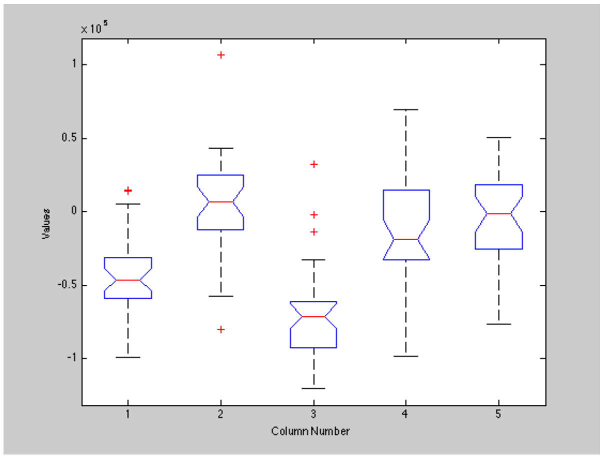


Figure 14Mean profits of different models using the Russian ruble; P-value 0.0000 (1=GA, 2=RRL, 3=ANN, 4=Hybrid 1, 5=Hybrid 2)

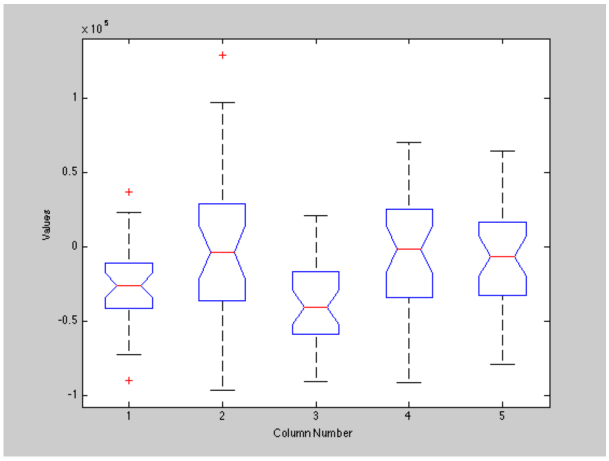


Figure 15Mean profits of different models using the Swedish krona; P-value 0.0001 (1=GA, 2=RRL, 3=ANN, 4=Hybrid 1, 5=Hybrid 2)

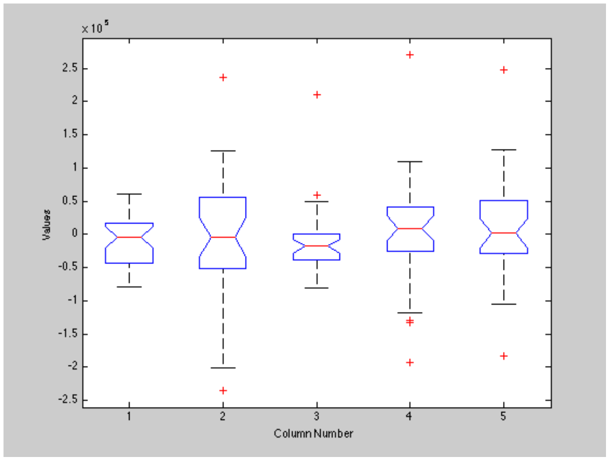


Figure 16Mean profits of different models using the US dollar; P-value 0.6199 (1=GA, 2=RRL, 3=ANN, 4=Hybrid 1, 5=Hybrid 2)

The test shows that the null hypothesis has to be rejected in the case of the ruble and the krona. <u>There are indeed statistically significant performance differences among the models.</u> With the US dollar such differences can, however, not be confirmed. The mean profits are too alike to make any statistically sound conclusions about the differences. More thorough comparison among the models might reveal more information.

4.1.4 Normality Assumption of Returns

Figure 17 summarizes the descriptive statistics of the model returns. The histograms should ideally resemble the normal distribution. We chose to test the assumed normality of the returns with an unknown mean and variance by means of Jarque-Bera test. We distinguish between the returns of the 5 selected algorithms – GA, RRL, ANN, Hybrid1 and Hybrid 2 – and conclude that only the returns of GA and RRL (represented on the first row) can be considered as normally distributed.

It is noteworthy that as soon as the number of tested days exceeds 20, the t-test that we use in this context is well-functioning even if there is a need to give up the normality assumption. Hence, the results from the other results in this paper should give credible results.

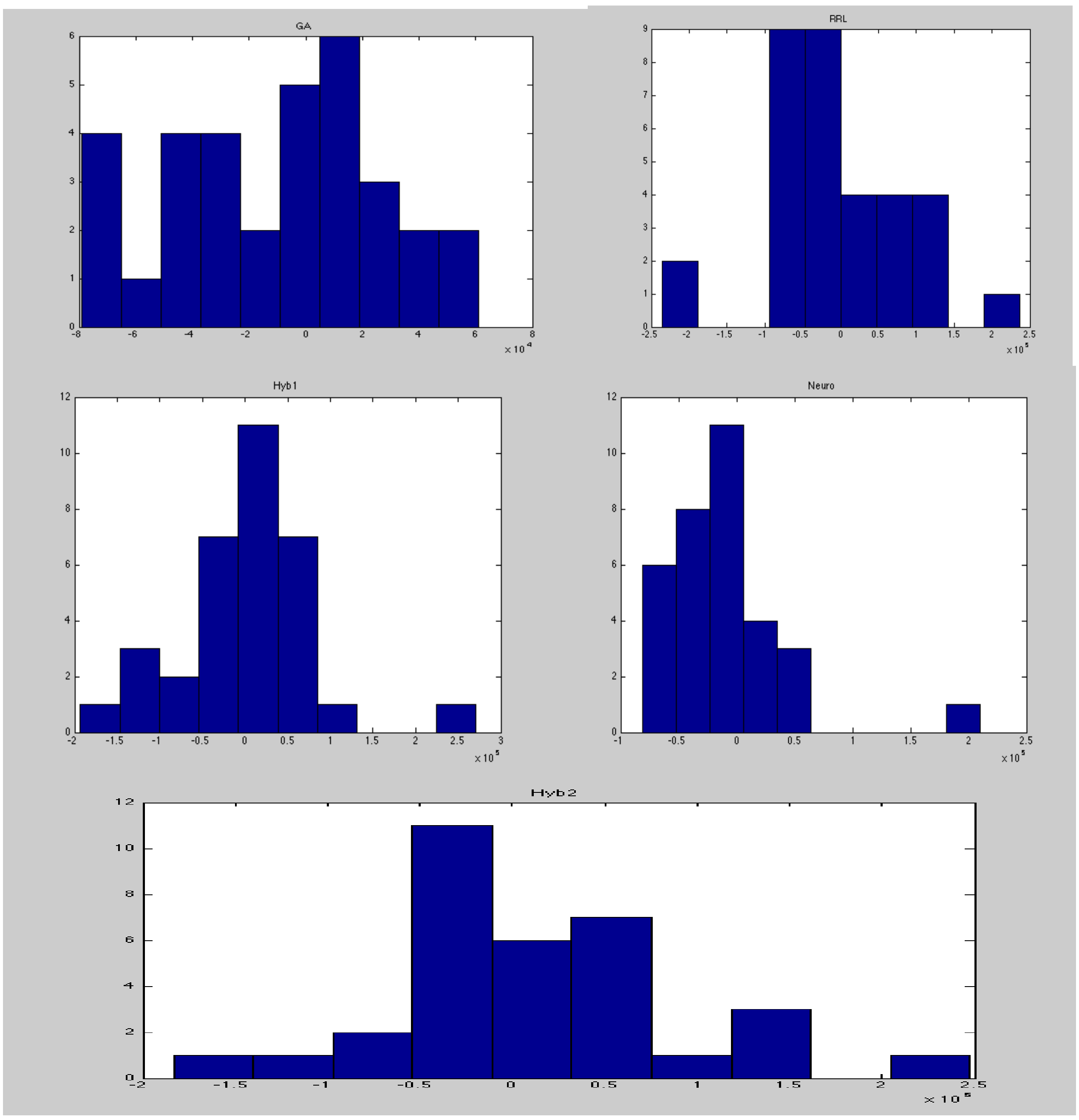


Figure 17 Descriptive statistics of the model returns

Model returns are represented on the horizontal and number of observations, ie days, on the vertical axis.

4.2 Analysis of Daily Results for Currencies

In this section, we examine the performance obtained when simulating the chosen algorithms. For clarity, we discuss the results of each currency in turn and aim at concluding why certain algorithms perform particularly well or weakly on certain types of days.

4.2.1 Results of EURUSD on Selected Days

The results of EURUSD on 11 March, 2008 (day 12 in our simulation) can be summarised as follows:

- None of the models was able to forecast the dramatic weakening of the euro around the time the US markets opened on 11 March.
- The best models were still able to correct their positions and place corresponding euro's sell orders which implies that relatively high cumulated profits of EUR 270 000 (in our context record high) were generated by Hybrid model 1, which put much weight on the RRL and the neural network based model. At the same time, the remaining models shifted to a strictly downward sloping profit curve and were ultimately loss-making.

Table 3 Performance summary of algorithms on March11, 2008 (EURUSD)

Algorithm	Hybrid 1	Hybrid 2	Genetic Algorithm	Neural Network	RRL
Cumulative profit	270 540	247 760	-76 379	247 660	268 850
Total variance	20 404 000	16 628 000	18 180 000	18 714 000	21 978 000
Number of trades *	165	113	274	347	187
Transaction cost **	3 382	2 316	5 616	7 112	3 833

*) Total number of trades during the day, maximum 420

**) Theoretical transaction cost related to number of trades

It is important to note that the hybrid 1 was able to adapt to market conditions and to take on board the best models (and drop the weak ones) when there were strong changes in the FX rate. The GA in turn was competing with the hybrid 2 in the race to the bottom. The RRL in turn seems to speculate during the morning hours – with not much success – until it assumes the euro has climbed to its record, which is the moment the RRL takes its long-standing position and starts to generate profits.

On 11 March, the euro quickly rose from $1.535 to above $1.545 during the morning hours (Finnish time). At the same time, the best trading model, hybrid 1, generated profits by rightly adjusting to this information. The overall number of orders placed during these hours by the hybrid 1 was reasonably low. The hybrid first decided to sell USD and then changed strategy to selling USD, remaining neutral and buying USD in turn until the euro reached 1.545. The rapid drop in the exchange rate occurred because the central banks of the US, the UK, Canada, and Switzerland announced their collective actions to loosen strict money markets, which made the dollar to strengthen against euro up from the record low levels of that time.

The top chart in Figure 18 illustrates the cumulative profits of the simulated algorithms and demonstrates the abilities of Hybrid Model 1 and the RRL. In the context of scaled positions, +20 can be interpreted as 20 million USD and vice versa. Decision “1” in turn can be interpreted as buying USD (selling EUR) and decision “-1” as buying EUR (selling USD).

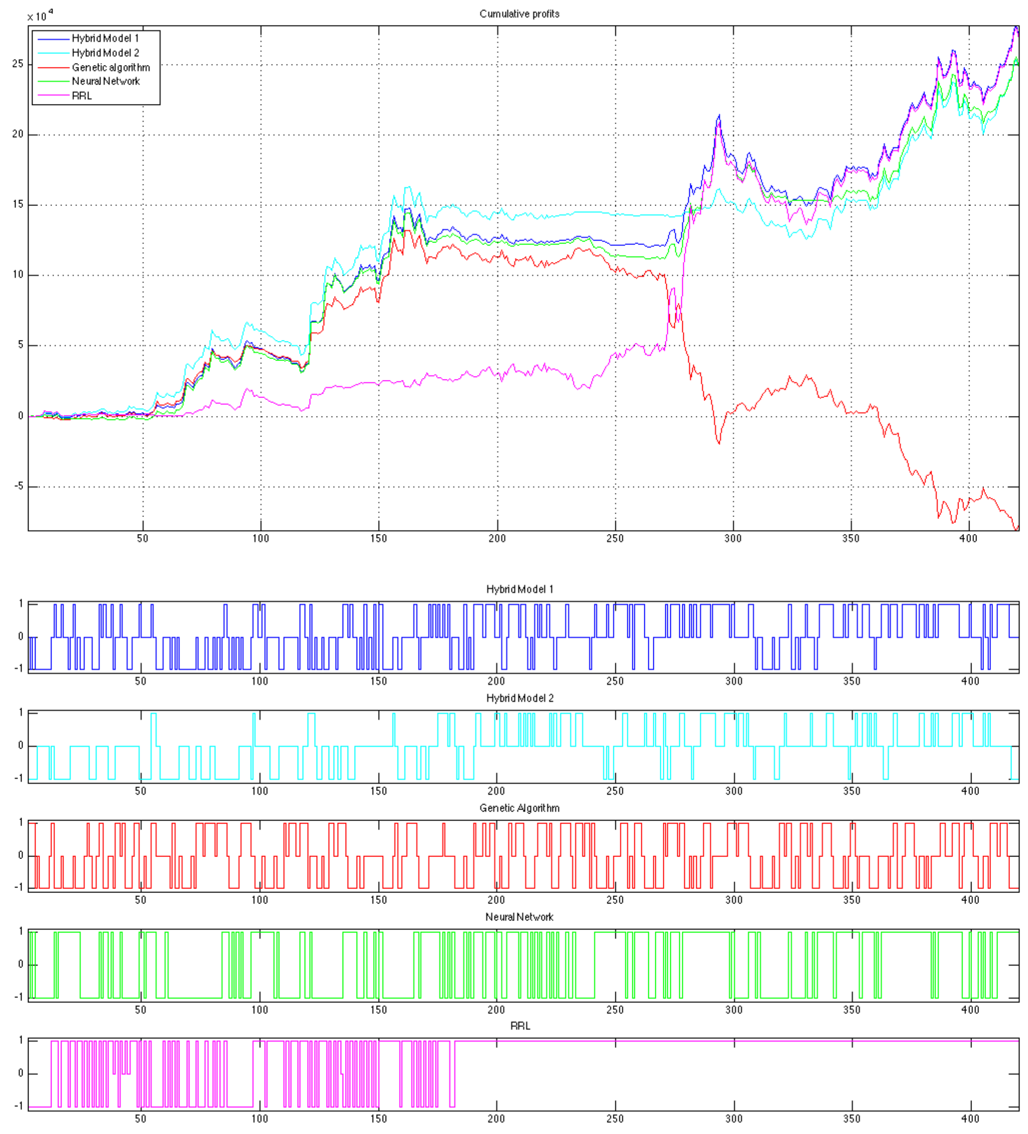


Figure 18 Cumulative profits on March 11, 2008 (Day 12) and buy/hold/sell decisions of models[8]

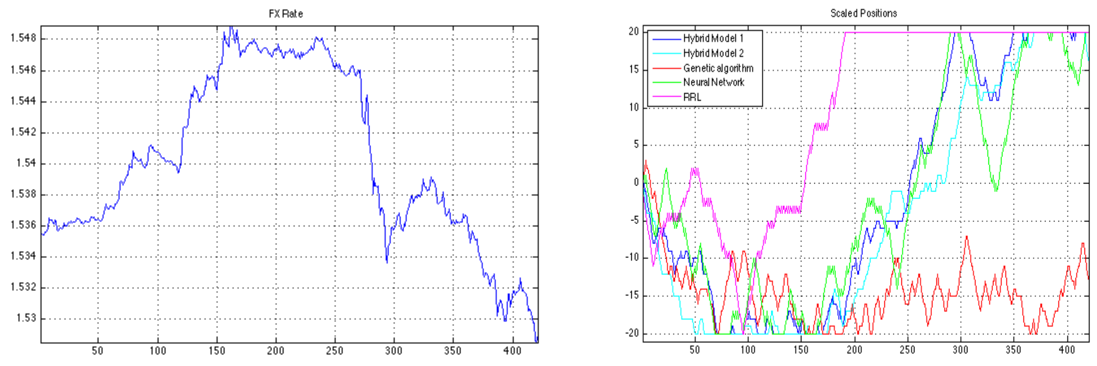


Figure 19EURUSD exchange rate on March 11, 2008 (Day 12) and scaled positions of models

[8] Neural network makes only buy and sell decisions

Somewhat exceptional trading patterns on January 22 (day 18 in simulations) brought out the following essential points:

- None of the models was able to profit significantly from the unexpected increase in the dollar rate of euro after its slightly decreasing phase at the latter half of the trading period. However, Hybrid model 1 was best to react to this sudden change.
- Performance of RRL and Hybrid model 2 totally collapsed in the course of the day and they broke the limits for maximum losses allowed.
- Tendency of RRL to over-fit to learning data might result in a catastrophic trading behavior in adverse circumstances for the model.

Table 4 Performance summary of algorithms on January 22, 2008 (EURUSD)

Algorithm	Hybrid 1	Hybrid 2	Genetic Algorithm	Neural Network	RRL
Cumulative profit	52 503	-77 083	26 734	18 131	-220 240
Total variance	18 444 000	36 646 000	9 680 200	12 262 000	185 660 000
Number of trades *	137	117	285	372	31
Transaction cost **	2 884	2 463	6 000	7 831	653

*) Total number of trades during the day, maximum 420

**) Theoretical transaction cost related to number of trades

On January 22, euro strengthened from $1.437 first relatively fast and then with a somewhat slower pace to $1.452 at the end of first half of the trading day (Figure 21). Then the rate declined slightly until it jumped suddenly to $1.458. This jump was a result of FED's actions as it unexpectedly lowered the interest rate resulting in the decrease of value of dollar. Neural Network system performed best through the day until the surprising raise in the euro rate (Figure 20). Also Hybrid Model 1 and Genetic Algorithm were able to generate some profits. However, Hybrid 1 performed best during the rest of the day after the jump and generated cumulative profits of about EUR 52 500 (Table 4). RRL and Hybrid 2, instead, collapsed through the floor of specified maximum level of losses and had to close down the trading positions for the day.

Especially the performance of RRL from the beginning of the day was exceptionally poor. Some reasons for this can be found from the learning data before the start of trading. In the learning data, dollar strengthened against euro right before the start of trading and this anchored the weights in the model making it rigid and over-fitted for this trend. Thus, when the euro rate started to climb step by step, RRL was incapable of detecting this changing trend and reacting to it accordingly. Instead, it bet all on declining dollar, which soon appeared to be highly risky business.

Furthermore, the nosedive of profits for Hybrid Model 2 on January 22 occurred at the same time of the sudden jump in the dollar value of euro at the latter half of the trading day. Hybrid 1, instead, weighted more the better performing models of GA and Neural Network and finished the day with profits.

During the day, hybrid models traded relatively actively. It is also interesting that Hybrid 1 increased strongly euro position without a single order placed on dollar during a long period in the middle of the day. When the dollar rate of euro started to decline slightly in the middle of the day, basically all the models started gradually to move their scaled positions towards dollar, which made them slow to react to sudden upcoming increase in the rate.

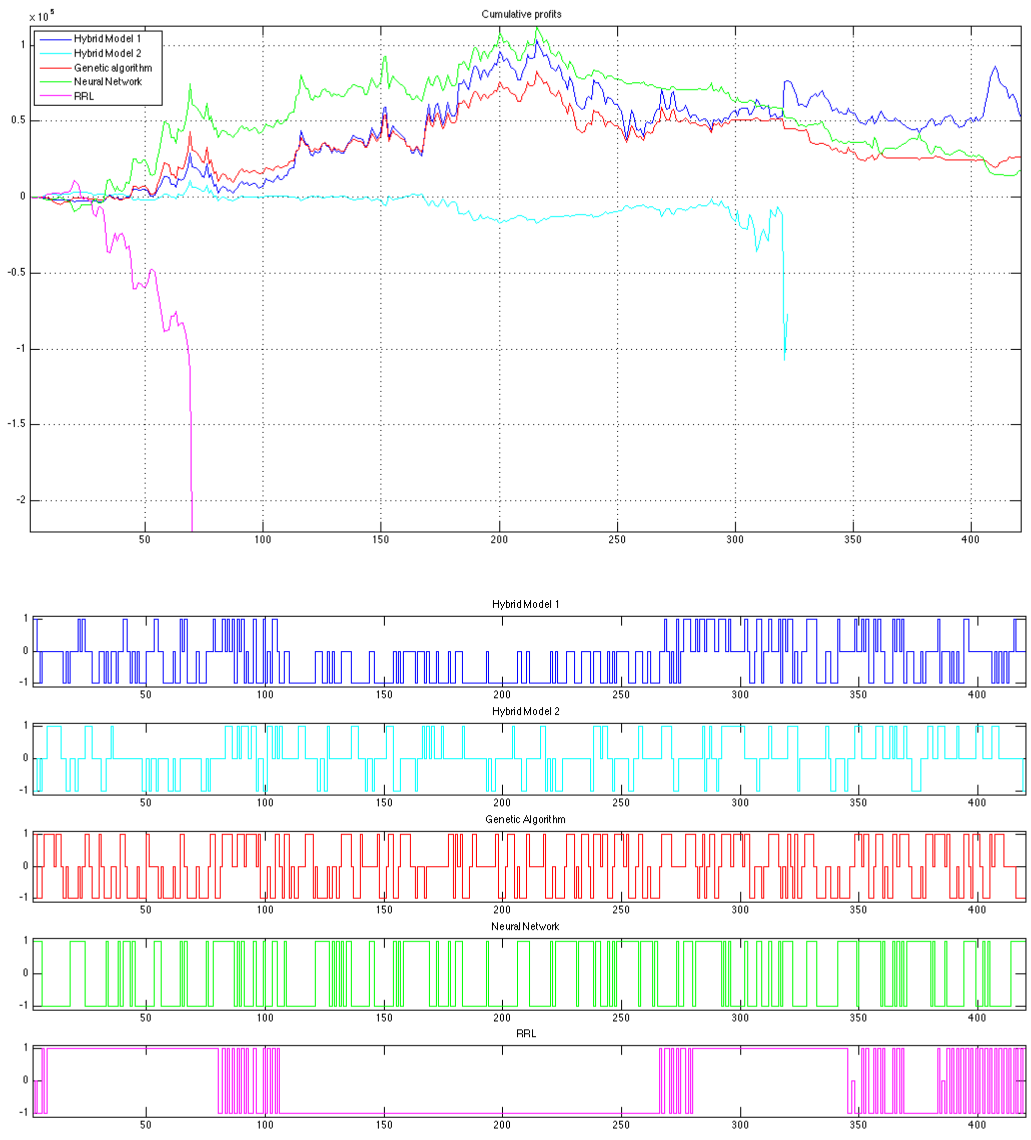


Figure 20 Cumulative profits on January 22, 2008 (Day 18) and buy/hold/sell decisions of models

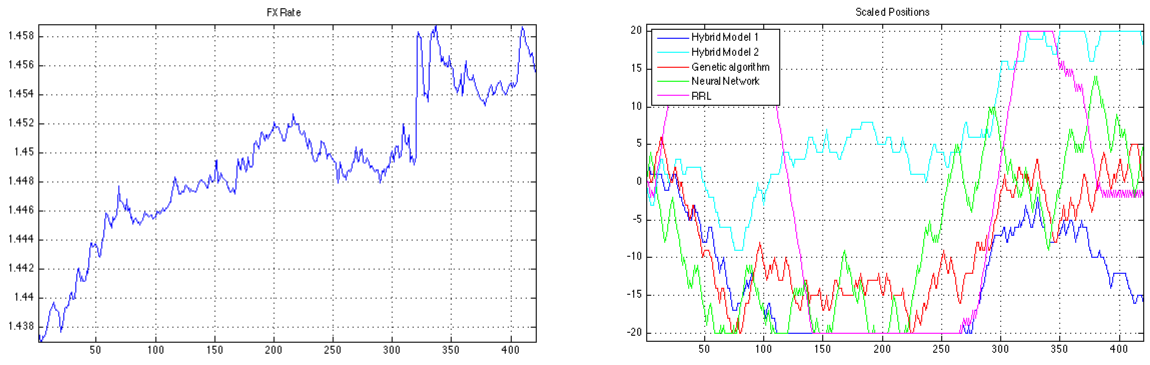


Figure 21EURUSD exchange rate on January 22, 2008 (Day 18) and scaled positions of models

4.2.2 Results of Trading EURSEK on 27 February

In sum, we can demonstrate that on 27 February (day 5 in our simulation):

- The RRL and the hybrid models perform well because they are able to immediately react to important market information;
- The hybrid drops trading systems based on neural network and GA as soon as they become loss-making;
- The results can be too optimistic because the over-fitting tendency of RRL.

Table 5 Performance summary of algorithms onFebruary27, 2008

Algorithm	Hybrid 1	Hybrid 2	Genetic Algorithm	Neural Network	RRL
Cumulative profit	70 347	64 540	13 745	20 659	71 088
Total variance	5 437 200	4 994 600	2 623 100	4 126 200	5 599 700
Number of trades *	21	47	246	368	21
Transaction cost **	22 463	50 274	263 130	393 630	22 463

*) Total number of trades during the day, maximum 420

**) Theoretical transaction cost related to number of trades

We first discuss results of the 27th of February 2008 (day 5 in our simulation) in light of the observed FX rate developments and the simulated trading systems performance. This day is particular since all our models generate profit. However, the cumulative profits are only at 70 000 euros at maximum (and with a variance as high as 5 600 000). The first chart in Figure 23 shows the FX rate with a notable jump early in the day and the fact that the value of home currency is continuously increasing. The RRL and both hybrid models are able to immediately react and take the right position, which implies a corresponding jump in these models cumulative profits. The same occurrence is oppositely interpreted by the neural network based and the GA based systems, which brings them to lower profit curves than those of the previously discussed models. An opposite jump of almost an equal magnitude follows later in the day. None of the models are able to interpret this information correctly, which has a negative but a temporary impact on all the models' cumulated profits.

The RRL system follows the intuitively right assumption to invest in the strong euro. The RRL system makes 21 trades during the day and performs well with this chosen strategy (see cumulative profits and scaled positions). Profits of the hybrid model 1 are almost equivalent with those of RRL. This can be explained by the rightly chosen model weights and the relatively high weighting of RRL. The other two models weights logically change because the hybrid model adapts rightly to market conditions by 'drop-ping' trading systems, i.e. neural network and GA, as soon as they become loss-making. The hybrid model's weights change with any relevant market developments.

We have to confess that the RRL model has a tendency to over-fit – which is the case also with some other days, a way before the start of the trading period. The reason for that is the high level of noise in the data and complex interdependencies among many of the parameters. The above good performance of RRL can be partly explained by the constraints we were bounded by, i.e. the fixed position size assumption laid down in the task we were given.

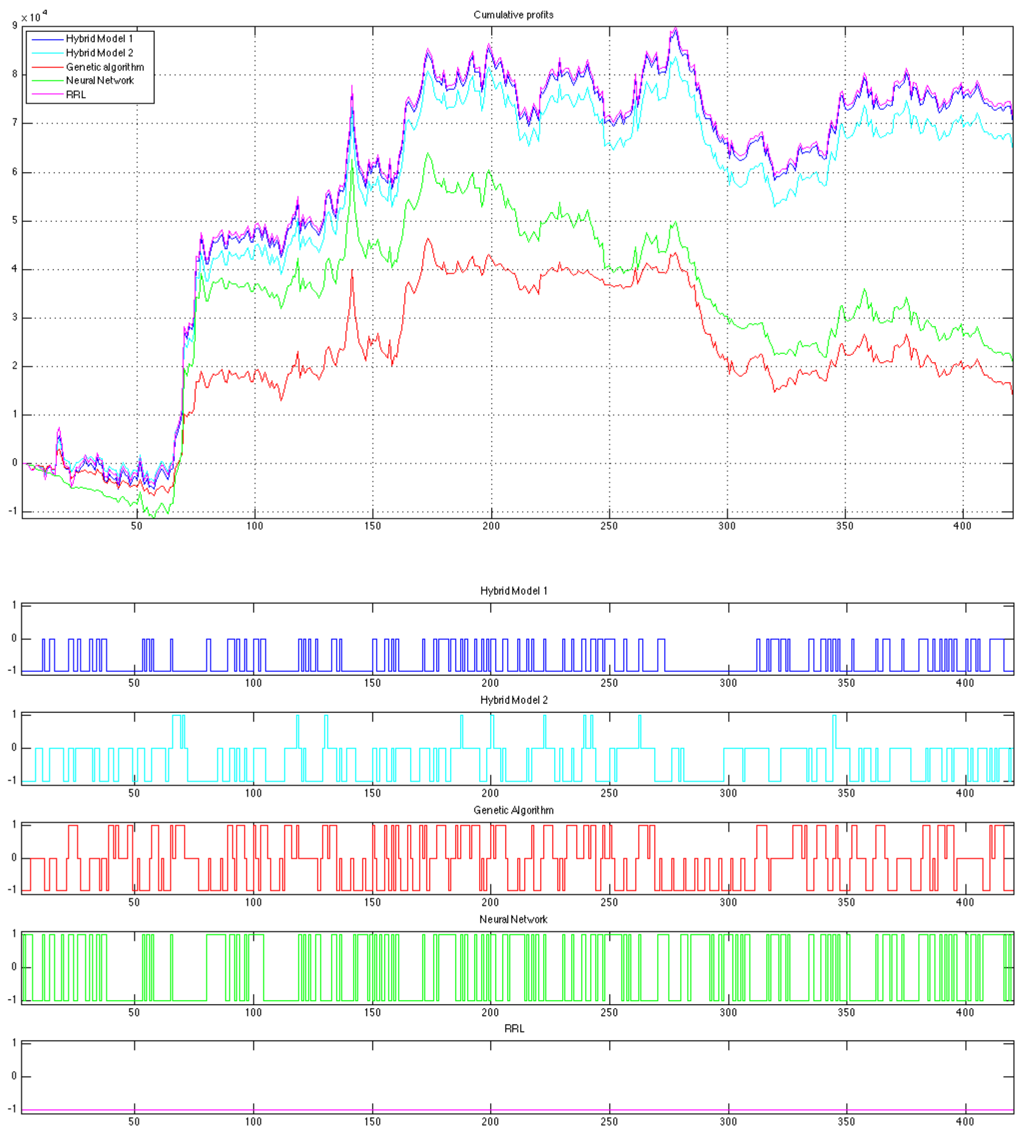


Figure 22 Cumulative profits on February 27, 2008 (Day 5) and buy/hold/sell decisions of models

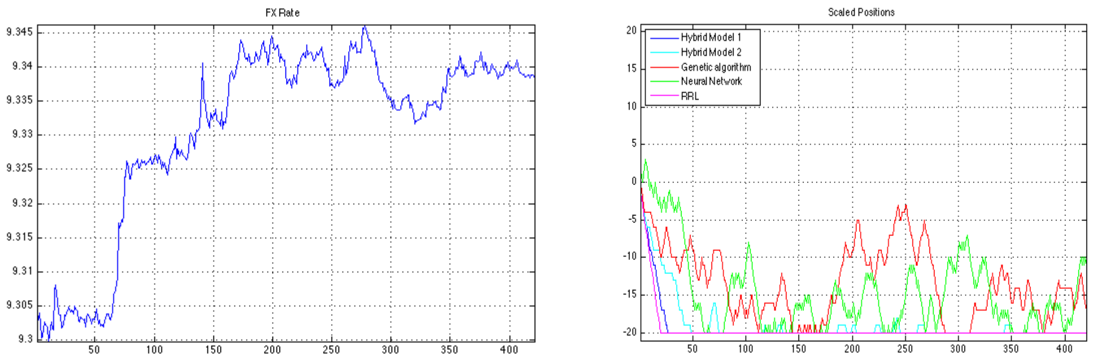


Figure 23EURSEK exchange rate on February 27, 2008 (Day 5) and scaled positions of models

4.2.3 Results of Trading EURRUB on 5 February

The key findings of the analysis of February 5, 2008, indicate the following:

- RRL predicts speculatively the strengthening of ruble and, accordingly, the continuous trend lends wings to RRL earning the highest daily profit of over EUR 100 000 with low trading activity. The over-fitting of RRL generates, thus, high return in favorable conditions.
- As the other individual models trade also profitably, the hybrid models do not lean exclusively only on extreme decisions of RRL and they end up generating notably lower profits.
- The patterns of cumulated profits are relatively similar in terms of their geometry and timing of decisions. However, the profit levels vary depending on the intensity of trading between different models.

Table 6 Performance summary of algorithms onFebruary 5, 2008

Algorithm	Hybrid 1	Hybrid 2	Genetic Algorithm	Neural Network	RRL
Cumulative profit	69 673	50 546	14 354	31 965	106 560
Total variance	9 191 100	6 699 600	4 419 700	8 061 400	10 760 000
Number of trades *	132	118	267	365	23
Transaction cost **	460 170	411 360	930 790	1 272 400	80 181

*) Total number of trades during the day, maximum 420

**) Theoretical transaction cost related to number of trades

On February 5, the trend in euro’s exchange rate in ruble terms behaved rather stably (Figure 25). The trading opens with the rate decreasing from the opening rate of RUB 36.29. In the beginning of the day, ruble strengthens at faster pace but later euro levels off between RUB 36.1 – RUB 36.15. There is also a short temporary decrease in the rate near to the closing of the day. All the models detect the dominant trend during the day and are able to generate at least some profits (Figure 24).

The cumulated profit patterns are also quite similar and synchronous. Nevertheless, the profit levels of different trading systems vary and they are relatively evenly distributed. The best performing algo- trader on February 5 is RRL which picks up the threads of the main trend very early on. On the other hand, this kind of over-fitting could be perceived as extremely risk-seeking behavior as well. RRL generates cumulated profits of over RUB 100 000. The next best traders are Hybrid Models 1 and 2 (Table 6).

The profits of all the models drop at the later stage of trading when the rate increases back to previous range after sharp decline after almost 6 hours of trading (350 minutes). However, the models are able to generate more profits after the drop except for Genetic algorithm, which overreacts to the change and starts to move the emphasis of scaled currency positions more towards euro (Figure 25) by selling rubles actively. The trading patterns of hybrid models indicate that they are quite consistently buying rubles throughout the trading day. Their trading activity was also relatively high (132 and 118 trades) compared to RRL which carried out only 23 trades during the day.

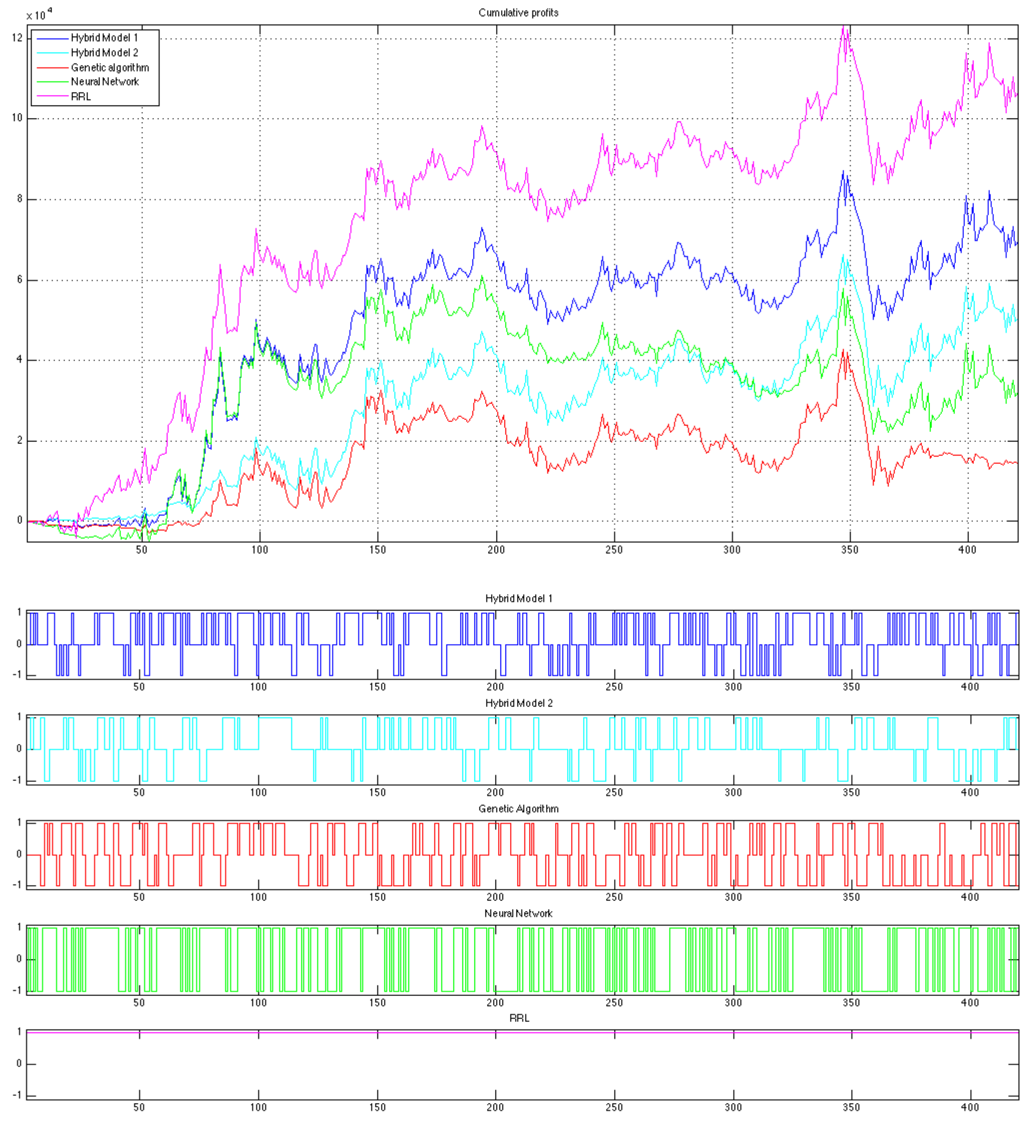


Figure 24 Cumulative profits on February 5, 2008 (Day 26) and buy/hold/sell decisions of models

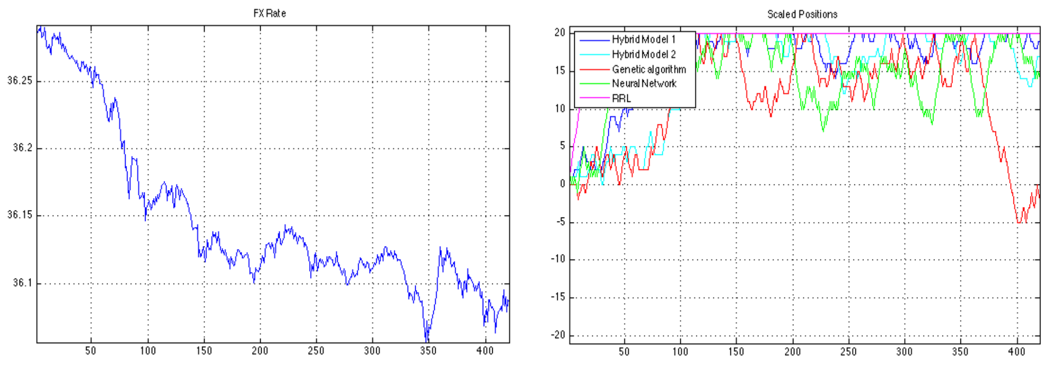


Figure 25EURRUB exchange rate on February 5, 2008 (Day 26) and scaled positions of models

5. Discussion and Conclusions

We have successfully built and tested a number of foreign exchange trading algorithms that we had found to be relevant for this work. We conducted thorough tests with these algorithms and were able to interpret the findings and explain some of the logic behind the successes and failures.

Out of the individual basic models we found the RRL model to outperform the other models quite clearly in most situations. The model is prone to over fitting but in most cases this isnot an issue unless something unexpected happens. The RRL model has the ability to generate a profit with the krona and the ruble with relatively few transactions. The ANN, on the other hand, suffers from high number of transactions it performs on daily basis, and whatever strengths it has,they are counterbalanced by the unnecessarily high transaction costs.

The RRL model was also the fastest one to run. The GA and the neural network needed considerably more time to run the daily cycle. With powerful computers and by rewriting the code slightly an online trading system could be implemented using these algorithms. Processing power is not an issue with this set of algorithms.

There are differences in how the models cope with different currencies, as it was expected. When trading euros and dollars, the differences between the models are much less pronounced than with the other currencies. Additionally, the second hybrid model performs notably better with dollars than it does with other currencies. That hybrid takes into account technical indicators, including moving averages, and is able to produce a profit with combining all this information. The relative equality between the models does hint at the high efficiency of the dollar market, which is after all by far the biggest market out of those tested here.Interestingly, the RRL model was the only one that could show a positive return in the ruble and krona markets in the long run. Nonetheless, its performance didnot help the hybrids to avoid losses.

Due to the limits of this project, some shortcuts had to be made. One point left for further development was the decision making algorithm for the ANN. The network gives currently only “buy” and “sell” orders and it does not allow for a neutral decisions. This leads inevitably to high transaction costs and drives the algorithm very often to the limit of its allowed position. Reaching the limit alters the algorithms behavior, as it cuts the transaction costs automatically and makes it possible to make profit with a bit of luck. It would be interesting to see how the ANN would performin this respect and would it be on a par with the other algorithms.

The success of the RRL algorithm was to high degree dependent on the right choice of model parameters. These specific parameters were mostly picked using heuristic methods, so the choice was not based on any more analytical process. The chosen parameters did prove to succeed quite well but their performance in other conditions remains unclear. The parameters used in simulations were selected differently for different currencies. In addition, they are expected to function relatively well with other data that is somewhat similar to the original data used with parameter selection. We feel this approach to building the model has been justified in the scope of this project, as little previous work is available to support further refinement of the parameters.

The results of this report indicate that the best algorithms examined during this project (RRL and hybrid models) are suitable for supporting foreign exchange trading on-line. However, it should be noted that since the FX markets are becoming more efficient, it is getting more difficult to perform well with algorithmic trading based on price data only. Therefore, order book and order flow data based trading systems should obtain closer attention in the future research.

References

M. Alvarez-Diaz, A. Alvarez. 2003. Forecasting Exchange Rates Using Genetic Algorithms. Applied Economic Letters, 2003, 10, 319-322.

G. Deboeck. 1994. Trading on the Edge: Neural, Genetic and Fuzzy Systems for Chaotic Financial Markets. New York Wiley, 1994.

M.A.H. Dempster, T.W. Payne, Y. Romahi, G.W.P Thompson. 2001. Computational Learning Techniques for Intraday FX Trading Using Popular Technical Indicators. IEEE Transactions on Neural Networks, Vol. 12, No. 4, July 2001.

M. A. H. Dempster and C. M. Jones. 2000. The profitability of intra-day FXtrading using technical indicators. Centre Financial Res., Judge Inst. Management, University of Cambridge, Dec. 2000.

M.A.H. Dempster and C.M. Jones. 2001. A real-time adaptive trading system using genetic programming. Quantitative Finance Volume (2001) 397-413. Institute Physics Publishing.

M. A. H. Dempster and C. M. Jones. 2001. Can channel pattern trading be successfully automated? European J. Finance, 2001.

M.A.H. Dempster, V. Leemans. 2006. An automated FX trading system using adaptive reinforcement learning. Expert Systems with Applications 30 (2006) 543-552.

F.X. Diabold, J. Gardeazabal and K. Yilmaz. 1994, On cointegration and exchange rate dynamics. The Journal of Finance, Vol. 49, No. 2 (Jun., 1994), pp. 727-735.

C. Gold. 2003. FX Trading via Recurrent Reinforcement Learning. Computation and Neural Systems, California Institute of Technology. January 12, 2003.

Nigel Meade. 2002. A comparison of the accuracy of short term foreign exchange forecasting methods. International Journal of Forecasting, Volume 18, Number 1, January 2002 , pp. 67-83(17)

R. Meese and K. Rogoff. 1983. Empirical exchange rate models of the seventies. J. International Economics, 14 3-24, 1983.

R. Meese and K. Rogoff. 1983. The out-of-sample failure of empirical exchange rate models. Exchange rate and international macroeconomics, E. J. Frenkel, U. of Chicago Press, 1983.

J. Moody, M. Saffell. 2001. Learning to Trade via Direct Reinforcement. IEEE Transactions on Neural Networks, Vol. 12, No. 4, July 2001.

W. Huang, Y. Nakamori and S. Y. Wang. 2002. Using change-point detection to seek optimal training set for neural networks in foreign exchange rates forecasting. Submitted to International Journal of Computational Intelligence and Organization, 2002b.

H. Ince, T. B. Trafalis. 2006. A hybrid model for exchange rate prediction. Decision Support Systems 42, (2006) 1054–1062.

J. Kamruzzaman and R. A. Sarker. 2003. Comparing ANN Based Models with ARIMA for Prediction of Forex Rates. Asor Bulletin, Volume 22 Number 2, June 2003.

Joarder Kamruzzaman, Ruhul A Sarker and Iftekhar Ahmad. 2003. SVM Based Models for Predicting Foreign Currency Exchange Rates. ICDM 2003: 557-560

Joarder Kamruzzaman and Ruhul A. Sarker. 2004. ANN-based Forecasting of Foreign Currency Exchange Rates. Neural Information Processing – Letters and Reviews, Vol. 3. No. 2, May 2004.

L. Kilian and M.P. Taylor. 2001. Why is it so difficult to beat random walk forecast of exchange rates? (October 2001). CEPR Discussion Paper No. 3024.

M. T. Leung, A. S. Chen and H. Dauk. 2000. Forecasting exchange rate using general regression neural networks. Computer and Operations Research 27, (2000) 1093–1110.

F. Lisi and R. A. Schiavo. 1999. A comparison between neural networks and chaotic models for exchange rate prediction. Computational Statistical and Data Analysis 30, (1999) 87–102.

C.J. Neely and P.A. Weller. 2003. Intraday technical trading in the foreign exchange market. Journal of International Money and Finance, Volume 22, Number 2, April 2003 , pp. 223-237(15).

Ping-Feng Pai and Chih-Sheng Lin. 2005. A hybrid ARIMA and support vector machines model in stock price forecasting. Omega, Volume 33, Issue 6, December 2005, Pages 497-505.

M. Santini, A. Tettamanzi. Genetic Programming for Financial Time Series Prediction. In J. Miller, M. Tomassini, P.L. Lanzi, C. Ryan, A.G.B. Tetamanzi, and W.B. Langdon, editors, Proceedings of 4th European Conference on Genetic Programming (EuroGP), number 2038 in Lecture Notes in Computer Science, pages 361-370. Springer, 2001.

M. R. El Shazly and H. E. El Shazly. 1997. Comparing the forecasting performance of neural networks and forward exchange rates. Journal of Multinational Financial Management 7, (1997) 345–356.

M. R. El Shazly and H. E. El Shazly. 1999. Forecasting currency prices using a genetically evolved neural network architecture. International Review of Financial Analysis 8, (1) (1999) 67–82.

M. Smith. 1993. Neural Networks for Statistical Modeling. Van Nostrand Reinhold, New York, 1993.

Taylor, M. P. and Allen H. 1992. The use of technical analysis in the foreign exchange market. J. Int. Money Finance, vol. 11, 304–314.

P. Tenti. 1996. Forecasting foreign exchange rates using recurrent neural networks. Applied Artificial Intelligence 10, (1996) 567–581.

A. Trapletti, A. Geyer and F. Leisch. 2002. Forecasting exchange rates using cointegration models and intra-day data. Journal of Forecasting 21 (2002) 151-166.

V. Vapnik. 1998. Statistical Learning Theory. New York: Wiley, 198

S. Walczak and N. Cerpa. 1999. Heuristic principles for the design of artificial neural networks. Information and Software Technology 41, (2) (1999) 107–117.

S.Walczak, A. Tahai and K. Karim. 1998. Improved cash flows using neural network models for forecasting foreign exchange rates. Applications of Fuzzy Sets and the Theory of Evidence to Accounting II, eds. P. Siegel, K. Omer, A. deKorvin and A. Zebda (JAI Press, Stamford, CN, 1998), 293–310.

Wei Huang, K. K. Lai, Y. Nakamori and Shouyang Wang. 2004. Forecasting Foreign Exchange Rates with Artificial Neural Networks. International Journal of Information Technology & Decision Making, Vol. 3, No. 1 (2004), 145–165.

J. T. Yao and C. L. Tan. 2000. A case study on using neural networks to perform technical forecasting of forex. Neurocomputing 34, (2000) 79–98.

S. Yaser and A. Atiya. 1996. Introduction to Financial Forecasting. Applied Intelligence, vol. 6, 1996, 205-213.

Shanming Shi and Andreas S. Weigend. 1997. Taking Time Seriously: Hidden Markov Experts Applied to Financial Engineering. Computational Intelligence for Financial Engineering (CIFEr), 1997., Proceedings of the IEEE/IAFE 1997.

Appendix 1: Basic idea and formalism of Artificial Neural Network (ANN)

In this appendix we describe the main idea of constructing a trading system based on an artificial neural network. We also specify the main assumptions made and the limitations of the system.

As is consistent with the objectives of the current study, we wanted to create a model, which can estimate upcoming data points from a nonlinear time series. Thus, we decided to construct a neural network based model that estimates the next change of the FX rate. The implemented feed-forward network has two layers, and takes as input vectors 10 rate change lags and returns an output vector which last element represents the change of the rate in the next minute. The hidden layer uses hyberbolic tangent sigmoid transfer function with 10 neurons, and the output layer linear transfer function called 'purelin'. The two-layer architecture with a nonlinear and linear transfer functions gives the network the opportunity to learn nonlinear and linear relationships between input vector and output.

The network is trained just before trading using the FX rate data from 17.01 till 9.59, altogether 1009 input elements and one element for the prediction. Our network uses the Levenberg-Marquardt algorithm to train the net. After each time step the net is adapted again with new input vectors and output vector. The adaption trains the net with weight and bias learning rules with 10 sequential updates.

While testing the network, we noticed that the estimated rate change included a static error. To compensate the static error, a moving average is subtracted of the last element of the output vector. The model makes the decision based on this difference. If the difference is negative, the model returns 1 which is a sign to buy foreign currency. If the value of the difference is positive, the model returns –1, which stands for selling foreign currency. The model returns 0 a neutral decision, when the difference is zero. The essential elements of the constructed system are summarizedin the following:

- Feed-forward with two layers.
    - 1 hidden unit and 1 output unit;
    - the transfer function of the hidden unit is tansig;
    - the hidden layer has 10 neurons and the output layer 1;
    - the transfer function of the output unit is purelin;
    - the learning algorithm is Levenberg-Marquardt technique.
- Input: 10 lags for the difference of FX rate series.
- Output: Decision either buy (1) or sell (-1) foreign currency.
- Learning data is the FX rate from 17.01-9.59.
- The network is trained with learning data (10 input vectors with the length of 1009 elements: 16 hours and 49min), takes approximately 0.1 seconds.
- After each decision the network is not trained again but it is adapted with new data.
- Adaption: 10 passes through the sequence, the network is trained with weight and bias learning rules with sequential updates.
- Network estimates the change of the FX rate 1 step ahead.
- Decision rules:
    - if the FX rate is rising, output is –1, buy euros/sell foreign currency;
    - if the FX rate is falling, output is 1, buy foreign currency/sell euros.
- First decision is made at 10.00 and the last at 16.59, altogether 420 decisions.

The strengths and weaknesses of the implemented model aresummarized in the following:

- We decided to keep the network as simple as possible, so we could understand better the behavior of the algorithm. The real challenge is to determine all numbers of layers and neurons and the used parameters.
- In practice, the model gives only decisions weather to buy or sell. It is very expensive to trade at every time step.
- The model is very fast in decision making. The training takes only 0.1 seconds and the decision making loop returns the value in 0.2 seconds. The time to compute the next decision does not cause any problems for the implementation of the model in to a real-time system.
- There is a possibility to implement a neutral decision to the model. One way is to put a gap between decisions –1 and 1. The size of the gap depends strong on the FX rate series. This extension brings more parameters to the model to determine. We decided to leave the development of the extended model outside of this project
- Can we make decisions based on 1-step-prediction? Do we need a long-term prediction? Now the model predicts 3 out of 4 rate changes correctly, but we are currently not aware how reliable a long-term prediction would be.
- The FX rate series is noisy. That means at almost every second step the series has increased and every other step it has decreased. Thus, the trading positions will grow with low probability up to their limit. If the position has reached its maximum, the model is not able to buy more currency. Keeping the maximum position is risky, but it saves the transaction costs.
- The model cannot predict unusual behavior of the FX rate, for example huge jumps.

Next, we present the underlying mathematical tools. We first provide two schematic illustrative examples of the network:

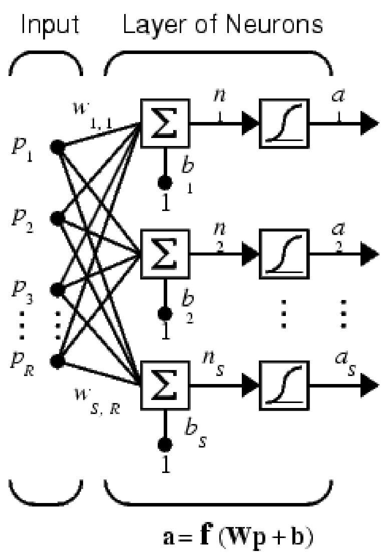


Figure 26Example of a network layer with a sigmoid transfer function (Source: MATLAB help)

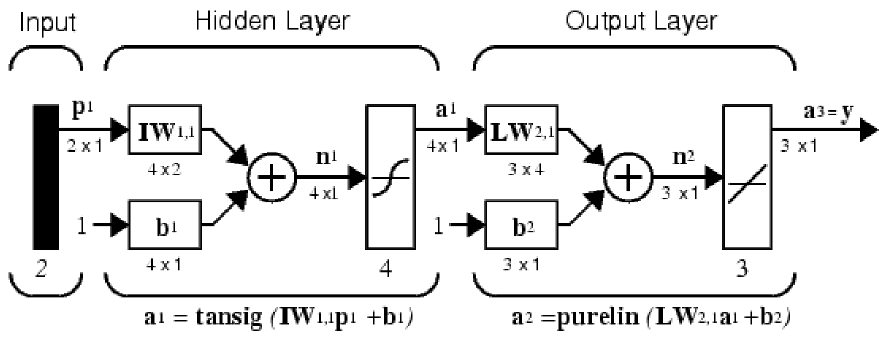
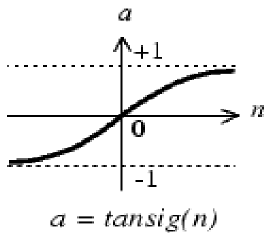


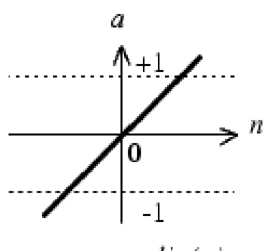
Figure 27Example of two-layer network. The example of the picture has only 2 input vectors but it has also two layers with the same transfer functions. (Source: MATLAB help)

Tan-Sigmoid Tranfer Function: Hyberbolic tangent sigmoid transfer function is a nonlinear function which takes values from the range of $(-\infty,\infty)$ and returns values from the range of $[-1,1]$. It has the same shape as the hyberbolic tangent function, but it is not as accurate as hyberbolic tangent. The transfer function has the following algorithm:

$$y = \frac{2}{1+e^{-2x}} - 1, \quad x \in (-\infty,\infty)\ y \in [-1,1]$$



Linear Transfer Function: Purelin is a linear transfer function. We use the linear transfer function for the output layer for it produces values outside the range $[-1,1]$.



$a = purelin(n)$

Levenberg-Marquardt (L-M): The L-M algorithm is a numerical optimization technique for neural network training. This method is one of the ‘faster training’ algorithms. The algorithm is based on Newton’s method, but this does not calculate the second derivatives. The high speed is reached by using a approximated Hessian matrix $H = J^T J$, where H is the Hessian matrix:

$$H(f) = \begin{bmatrix} \frac{\partial^2 f}{\partial x_1{}^2} & \frac{\partial^2 f}{\partial x_1 \partial x_2} & \cdots & \frac{\partial^2 f}{\partial x_1 \partial x_n} \\ \frac{\partial^2 f}{\partial x_2 \partial x_1} & \frac{\partial^2 f}{\partial x_2{}^2} & \cdots & \frac{\partial^2 f}{\partial x_2 \partial x_n} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial^2 f}{\partial x_n \partial x_1} & \frac{\partial^2 f}{\partial x_n \partial x_2} & \cdots & \frac{\partial^2 f}{\partial x_n{}^2} \end{bmatrix}$$

and J is the Jacobian matrix:

$$H(f) = \begin{bmatrix} \frac{\partial y_1}{\partial x_1} & \cdots & \frac{\partial y_1}{\partial x_n} \\ \vdots & \ddots & \vdots \\ \frac{\partial y_m}{\partial x_1} & \cdots & \frac{\partial y_m}{\partial x_n} \end{bmatrix}$$

The gradient is computed as $g = J^T e$, where e is the error matrix.
The L-M algorithm uses the approximation to the Hessian matrix in the update:

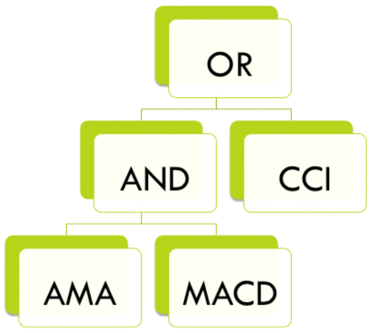
$$x_{k+1} = x_k - [J^T J + \mu I]^{-1} J^T e$$

When the scalar μ is zero, this update is same as Newton’s method with the approximation of Hessian matrix. When μ is large, the method is gradient descent with a small step size. The aim is to get the method more like Newton’s method as quickly as possible because it’s faster and more accurate near an error minimum. Thereby, after every step the error gets smaller, μ is decreased closer to zero and vice versa. So, the performance function will be reducedat each iteration.

Appendix 2: Implementing Genetic Algorithm strategy search using indicators of technical analysis

The idea of this algorithm is based on the fact that traders do have some ideas or even a strategy about how to combine information from various technical indicators and make decisions based on that. This algorithm seeks the optimal strategy that uses indicators and bases it decisions on that strategy. To be able to respond to a changing situation the algorithm updates it’s strategy every 5 minutes.

These strategies are defined with Boolean operators (nor,or,and,xor) and with seven technical indicators (ama,cci,macd,pcb,rsi,sma, stochastic),for example:



The strategies are described as trees like the one represented above. There is a similar strategy tree for short and long position and the final decision is a combination of these two decisions. The neutral position is taken if both short and long position strategy trees give opposite decisions. As can be seen, there are quite many different combinations of the strategies. However, only few will give different decisions. The two trees, one for short and one for long position, are results from those decisions and they are easily calculated with Boolean values (0,1) compared to the set of decisions which algorithm returns (-1,0,1).

These strategies form the population in GA. The consistency of this population is then optimized in respect to a certain criterion. The criterion was decided to be the sum of the profit made in the last 5 minutes. The sum determines the fitness of a single member of the population. The fitness of the overall population is calculated by summing up the fitness values of all the members of the population. The overall fitness is a criterion to stop the GA’s search for the optimum and to choose the best performing strategies for those 5 minutes. We decided that it would be more realistic to take few best performing strategies and construct the final decision out of those best strategies. Other kinds of criteria for fitness were also considered, such as a risk adjusted return similar to the one in RRL.

The size of the population and the tightness of the fitness criterion affect greatly the running time of the algorithm. In this case, we decided to do a compromise on accuracy rather than speed when searching for the strategy. This was due to tight testing schedule and limited computing capacity.

Implementation algorithm was done with MATLAB. A single member of the population was represented as avector of18 units of zeros and ones. This ‘chromosome’ was enough to code all necessary information about the strategy.

Appendix 3: Basic idea and formalism of RRL

The implementation of RRL algorithm was done by MATLAB which was seen a versatile and easily used platform in this kind of a project. The key components of the algorithm are the decision function and learning function of process. There are several ways to see whole algorithm but in our case RRL algorithm can be seen as a single layer neural network (see e.g. Dempster et al 2003 and 2006, and Moody and Saffel 2001). The decision function should give the actions which algorithm suggests. In this project it was decided that there are three possible actions which trading algorithm could take: short, neutral and long position. Decision set was coded to represent discrete set of $F_t \in$ [-1,0,1]. Decision function is:

$$F_t = \tanh(uF_{t-1} + v_0 r_t + v_1 r_{t-1} + v_2 r_{t-2} + v_3 r_{t-3} + v_4 r_{t-4} + w), \quad r_t = z_t - z_{t-1}$$

where $F_{t-1}$is the last decision, , r_t's are price returns and [u,v,w] are weights a.k.a system parameters. The tanh is used due to its differentiability when compared to sign. However, discrete decisions are made by rounding with certain parameter. Thus this parameter enables one way to affect how big changes in value of decision function represent the change to different decision. The replacing sign with tanh has been seen one option in literature and it is preferred due it straight forward nature.

The performance criterion which is used to adapt system parameter weight is similar ratio like Sterling ratio. The differential ratio of risk adjusted return is used in online updating weight parameters. However, other measures could be used as well, for example Sharpe ratio. Thus we decided to use similar ratio like Sterling ratio because of its ability topenalize only loses and that way give good view of drawdown. Let's define differential ratio for measuring risk adjusted performance:

$$DD_T = \left(\frac{1}{T}\sum_{t=1}^{T}\min\{R_t,0\}^2\right)^{1/2}$$

$$DDR_T = \frac{average(R_t)}{DD_T}$$

As we see, the measure rewards large average returns and penalizes risky returns where risky refers to down side returns.

To be able to use this ratio to update weights in the algorithm, we need to differentiate it and compute return at time t affect to value of ratio:

$$D_t = \frac{dDDR_t}{d\eta} =$$

$$= \frac{R_t - \frac{1}{2}A_{t-1}}{DD_{t-1}}, \quad R_t > 0$$

$$= \frac{DD_{t-1}^2 \cdot (R_t - \frac{1}{2}A_{t-1}) - \frac{1}{2}A_{t-1}R_t^2}{DD_{t-1}^3}, \quad R \le 0$$

$$A_t = A_{t-1} + \eta(R_t - A_{t-1})$$

$$DD_t^2 = DD_{t-1}^2 + \eta\left(\min\{R_t,0\}^2 - DD_{t-1}^2\right) \quad ,$$

whereη is the learning rate. The learning rate is a parameter that greatly affects the performance of the algorithm and how it acts. Further discussion concerning this specific parameter will be provided in the forthcoming.

We learn the system to trade by adjusting parameters [u,v,w] in to order to maximize 'utility' thatis in our case differential ratio risk adjusted return. We define the change in utility respect to change of parameters:

$$\frac{dU_T(\theta)}{d\theta} = \sum_{t=1}^{T} \frac{dU_T}{dR_t} \left\{ \frac{dR_t}{dF_t} \frac{dF_t}{d\theta} - \frac{dR_t}{dF_{t-1}} \frac{dF_{t-1}}{d\theta} \right\}, \quad \theta \textit{ is parameters}$$

Optimization of the trading parameters can be done with gradient ascent method. This is done repeatedly computing batches forward.

$$\Delta\theta = \rho \frac{dU_T}{d\theta},$$

where ρ is the learning rate. Note that due to inherit recurrence, the value of value of utility is depending on all the previous time periods and the derivatives are total derivatives that we need to define with recursive update equation:

$$\frac{dU_t(\theta)}{d\theta_t} = \frac{dU_t}{dR_t} \left\{ \frac{dR_t}{dF_t} \frac{dF_t}{d\theta_t} - \frac{dR_t}{dF_{t-1}} \frac{dF_{t-1}}{d\theta_{t-1}} \right\}$$

$$\frac{dF_t}{d\theta_t} = \frac{\partial F_t}{\partial \theta_t} + \frac{\partial F_t}{\partial F_{t-1}} \frac{\partial F_{t-1}}{\partial \theta_{t-1}}$$

Using these equations we get an equation for updating weight parameters:

$$\Delta\theta = \rho \frac{dD_t(\theta_t)}{d\theta_t} = \rho \frac{dD_t}{dR_t} \left\{ \frac{dR_t}{dF_t} \frac{dF_t}{d\theta_t} - \frac{dR_t}{dF_{t-1}} \frac{dF_{t-1}}{d\theta_{t-1}} \right\}$$

Define parameters for next time step are

$$\theta_{t+1} = \theta_t \beta + \Delta\theta_t,$$

where β is a decay rate. At now we are defined decision function and how weights in it are defined and repeatedly update 'learned'.

The learning parameters make all the difference when it comes to an algorithm working or not working. The decision function is primarily responsible for the dynamics of the algorithm but if the weights are somehow adjusted towards 'wrong' values, it could lead to quite irrational behavior of the algorithm. The algorithm can become 'stiff', in other words unresponsive to inputs or only slowly responsive. We also noted that choosing the right learning parameters is closer to art that science. One could argue that there cannot be only one set of parameters, which would work efficiently with all the currencies. The relations between the parameters and the returns seem highly nonlinear.

Appendix 4: Basic idea and formalism of hybrid model

The goal of the hybrid model is to combine all the strengths of each implemented submodel described above. The underlying idea is that a group of experts have a higher probability to make the right decision than any expert of the group alone:

- The hybrid model uses all the decision vectors of the underlying models as inputs. There is no limit for the number of methods that can be used in the hybrid model. The most important thing is that the input vectors have the same length and they are synchronized.
- The hybrid model returns 420 decisions as an output vector
- The model makes its decision based on decisions other models have made. Decisions will be weighted. The weights are calculated with respect to their earlier performance, so that the last decision matters the most and the importance of older decisions is depreciated.
- The hybrid model calculates the right decisions for the last iteration round after it gets the FX rate. The hybrid model compares the right decision to the decisions the models have made. After comparing, the model updates the weight vector. Then it is ready to make a new decision.

The hybrid model takes following input matrix:

$$x = \begin{bmatrix} x_{1,1} & \cdots & x_{1,420} \\ \vdots & \ddots & \vdots \\ x_{n,1} & \cdots & x_{n,420} \end{bmatrix},$$

where n is the number of models the hybrid uses.

The hybrid model returns a set of decisions as an output vector:

$$y = \begin{bmatrix} y_1 & \cdots & y_{420} \end{bmatrix}.$$

The hybrid model makes its decisions based on decisions other models have made. Decisions of the other models will be weighted. The weights are calculated with respect to their earlier performance, so that the last decision matters the most and the older decisions are depreciated.

The cost function:

$$e(m,i) = \frac{\left| r(t-i) - x(m,t-i) \right|}{(1+\mu)^{i}},$$

where i is the lag of the decision, e(m,i) is the cost function for the decision of the model m and r is the right decision[9] and x is the decision of the model m. The cost function measures the distance between the right decision and the decision of the model m. $(1+\mu)$ is the depreciation rate.

Weights are calculated with the following function:

[9] The right decision means here the decision that would have made profit in one-minute time step after the transaction cost.

$$w(m) = \frac{1 - \dfrac{\sum_{i=1}^{l} e(m,i)}{\sum_{m=1}^{n}\sum_{i=1}^{l} e(m,i)}}{n-1},$$

where *i* is the number of lags that are used in the weighting the decisions.The maximum weight for a model is 1/(n-1), and the minimum is 0.

- After the hybrid model gets the FX rate, it calculates the right decisions[10] for the last iteration round. The hybrid model compares the right decision to the decisions the models have made. After the comparison, the model updates the weight vector. Then the model is ready to make a new decision.

- We used the hybrid model with two different input matrixes in the simulation.

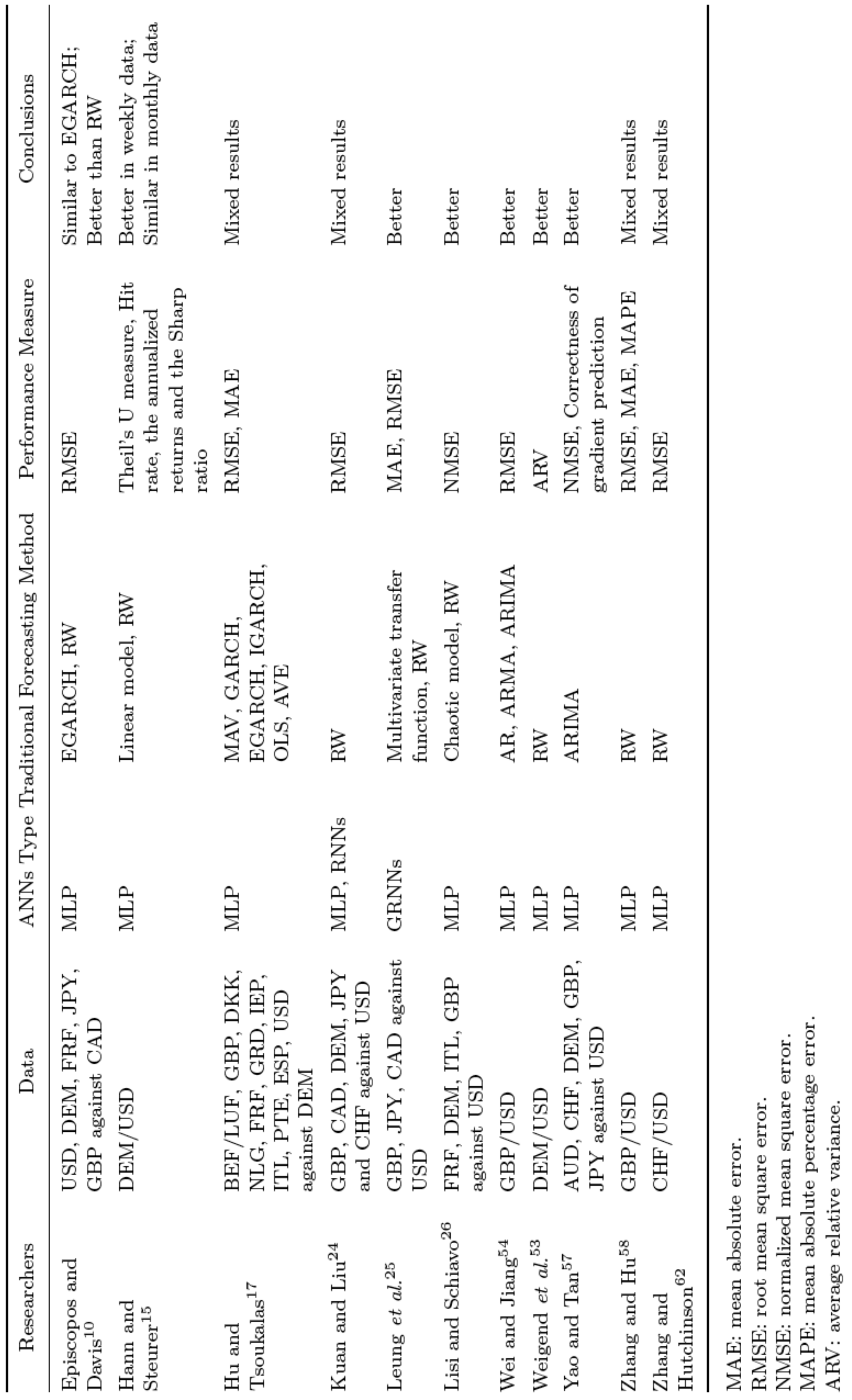
- The Hybrid Model 1 uses decisions of three models: RRL, Genetic Algorithm and Neural Network.

- The Hybrid Model 2 takes the decisions of 11 modelsas an input. The models we used are the same three as the first hybrid model and eight different technical indicators.

[10] The right decision means here the decision that would have made profit in one-minute time step after the transaction cost.

Appendix 5: ANN performance comparison

Source: Huang et al. (2004)



Researchers	Data	ANNs Type	Traditional Forecasting Method	Performance Measure	Conclusions
Episcopos and Davis[10]	USD, DEM, FRF, JPY, GBP against CAD	MLP	EGARCH, RW	RMSE	Similar to EGARCH; Better than RW
Hann and Steurer[15]	DEM/USD	MLP	Linear model, RW	Theil's U measure, Hit rate, the annualized returns and the Sharp ratio	Better in weekly data; Similar in monthly data
Hu and Tsoukalas[17]	BEF/LUF, GBP, DKK, NLG, FRF, GRD, IEP, ITL, PTE, ESP, USD against DEM	MLP	MAV, GARCH, EGARCH, IGARCH, OLS, AVE	RMSE, MAE	Mixed results
Kuan and Liu[24]	GBP, CAD, DEM, JPY and CHF against USD	MLP, RNNs	RW	RMSE	Mixed results
Leung *et al.*[25]	GBP, JPY, CAD against USD	GRNNs	Multivariate transfer function, RW	MAE, RMSE	Better
Lisi and Schiavo[26]	FRF, DEM, ITL, GBP against USD	MLP	Chaotic model, RW	NMSE	Better
Wei and Jiang[54]	GBP/USD	MLP	AR, ARMA, ARIMA	RMSE	Better
Weigend *et al.*[53]	DEM/USD	MLP	RW	ARV	Better
Yao and Tan[57]	AUD, CHF, DEM, GBP, JPY against USD	MLP	ARIMA	NMSE, Correctness of gradient prediction	Better
Zhang and Hu[58]	GBP/USD	MLP	RW	RMSE, MAE, MAPE	Mixed results
Zhang and Hutchinson[62]	CHF/USD	MLP	RW	RMSE	Mixed results

MAE: mean absolute error.
RMSE: root mean square error.
NMSE: normalized mean square error.
MAPE: mean absolute percentage error.
ARV: average relative variance.