Charles University in Prague

Faculty of Social Sciences
Institute of Economic Studies

MASTER THESIS

Modelling Durations Using Artificial Neural Networks

Author: **Bc. Martin Žofka**

Supervisor: **PhDr. Jozef Baruník Ph.D.**

Academic Year: **2013/2014**

Declaration of Authorship

The author hereby declares that he compiled this thesis independently, using only the listed resources and literature, and the thesis has not been used to obtain a different or the same degree.

The author grants to Charles University permission to reproduce and to distribute copies of this thesis document in whole or in part.

Prague, January 6, 2014

Signature

Acknowledgments

I would like to express my gratitude to PhDr. Jozef Baruník Ph.D. from the Institute of Economic Studies, Charles University and Institute of Information Theory and Automation at Academy of Sciences of the Czech Republic for supervising my thesis.

I would also like to thank Michal Žofka for his patience and help in the course of the work.

Abstract

The thesis introduces Artificial Neural Networks (ANN) to the field of financial durations. We begin by reviewing the findings about financial durations and models applied to analyze them. ANNs are then surveyed and one of the possible network architectures is selected for the forecasting. The selected ANN is a feed-forward network, with one hidden layer, a sigmoid activation function and a genetic algorithm for optimization. We use original and diurnally adjusted data for estimation and in contrast to other duration models, ANNs do not require data pre-processing. Therefore forecasts are estimated in one step without removing seasonalities for raw data. The estimates of the ANN are compared to estimates of the Autoregressive Conditional Duration (ACD) model, which serves as a benchmark for forecasting capabilities of the ANNs. The findings confirm that ANNs can be used to model durations with a similar accuracy as the ACD model. In the case of raw data the model slightly outperforms the ACD model, while the opposite is true for adjusted data, however the forecasting ability difference is not significant.

JEL Classification	C32, C45, C53, G17
Keywords	price durations,artificial neural networks, genetic algorithms

Author's e-mail	marzof_6@hotmail.com
Supervisor's e-mail	barunik@fsv.cuni.cz

Abstrakt

Hlavním cílem této diplomové práce je zavedení umělých neuronových sítí (UNS) pro modelování finančních durací. Na začátku shrneme stávající poznatky ohledně finančních durací a modelů pro jejich analýzu. Následně prozkoumáme stávající druhy UNS a vybereme jednu z možných architektur sítí pro následné modelování. Zvolená síť je vícevrstvá dopředná, má sigmoidní aktivační funkcí, jednu skrytou vrstvu a genetický algoritmus optimalizace. V práci používáme jak očištěná tak neočištěná data pro předpovídání, ale na rozdíl od ostatních modelů pro durace, neuronové sítě nevyžadují očištění dat. Lze je tedy odhadnout v jednom kroku bez potřeby odstranit sezónnosti. V další části práce porovnáme UNS s odhadem získaným modelem autoregresivních podmíněných durací (APD), který nám slouží jako měřítko pro

porovnání výkonnosti. Výsledky potvrzují, že UNS jsou schopné předpovídat durace s přibližně shodnou přesností jako APD model. Pro neočištěná data jsou lepší UNS, zatímco pro očištěná data vychází o něco lépe APD model. Nicméně rozdíly v předpovědích nejsou signifikantní.

Klasifikace JEL	C32, C45, C53, G17
Klíčová slova	cenové durace, neuronové sítě, genetické algoritmy

E-mail autora	marzof_6@hotmail.com
E-mail vedoucího práce	barunik@fsv.cuni.cz

Contents

List of Tables

List of Figures

Acronyms

ACD Autoregressive Conditional Duration

ACF Autocorrelation Function

ARIMA Autoregressive Integrated Moving Average

ANN Artificial Neural Network

CME Chicago Mercantile Exchange

FX Foreign Exchange

GARCH Generalized Autoregressive Conditional Heteroskedasticity

HMM Hidden Markov Model

LMSD Long Memory Stochastic Duration

MAE Mean Absolute Error

MLE Maximum Likelihood Estimation

MSE Mean Squared Error

PSO Particle Swarm Optimization

RMSE Root Mean Squared Error

SCD Stochastic Conditional Durations

VAR Vector Autoregressive

Master Thesis Proposal

Author	Bc. Martin Žofka
Supervisor	PhDr. Jozef Baruník Ph.D.
Proposed topic	Modelling Durations Using Artificial Neural Networks

Topic characteristics Due to technological changes over the past years the amount of data has increased significantly. The frequency, at which the data is available, has therefore shifted to higher-frequencies. In correspondence with the technological leap, the focus of analysis is now on individual transaction in financial markets rather than data on a daily basis. As a consequence of recording individual transactions, the time-series are now irregularly spaced. In comparison with equally spaced time-series the challenge at hand is to forecast the occurrence of the next transaction. In the sense of irregularly spaced time-series the term duration is used to specify the time-interval between two transactions. This work examines financial durations and uses Artificial Neural Networks (ANNs) as a method for forecasting in order to evaluate if they are able to outperform traditional models. The use of ANNs is motivated by their successful application to forecasting financial data on a daily basis. ANNs are able to discover patterns in complex nonlinear systems and so they are an alternative to traditional models in many fields. In the case of this work the traditional model to serve as a benchmark is the autoregressive conditional duration (ACD) model introduced by Engle and Russell in 1998.

The data consist of foreign exchange (FX) futures contracts traded on the Chicago Mercantile Exchange (CME). The data set includes all transactions for the Swiss Franc (CHF), Euro (EUR) and Japanese Yen (JPY) for futures contracts.

Hypotheses

1. ANNs are suitable for predicting durations (they outperform the traditional ACD model)

2. The predicted durations confirm stylized facts about the market microstructure

3. ANNs confirm hypotheses tested on ACD models

Methodology The term Artificial Neural Networks stems from the use of the biological concept of neurons, where a neuron is able to receive impulses from connecting neurons and send impulses, if a certain activation value is exceeded. Based on these assumptions a set of neurons is constructed with connections between neurons, where each of the connections is assigned a weight.

All inputs for the given neuron are multiplied by their respective weights, summed up and then transformed using an activation function. This transformed value then serves as the input for the next neuron. A neural network consists of three types of layers-input, hidden and output. The input layer contains the data supplied to the network. In the hidden layers the transformation of the inputs takes place and the output layer produces the estimate.

At the beginning a random set of weights is selected. Later a training algorithm is used for the determination of optimal weights between neurons. These optimal weights are trained by minimizing the error function.

ANNs have various specifications based on the architecture, activation function and training algorithm. The aim of the selection process is to determine the optimal setting of the ANN and compare if with the ACD model.

Outline

1. Introduction
2. Survey of related literature
3. Data description
4. Artificial neural networks model
5. Benchmark model (ACD)
6. Empirical results
7. Comparison of ANN and ACD

8. Conclusion

Core bibliography

1. Bishop, C.M. (1995): "Neural Networks for Pattern Recognition." *Clarendon Press, Oxford 1995.*
2. Engle, R.F.(2000): "The econometrics of ultra-high-frequency data." *Econometrica 68(1): 1-22.*
3. Engle, R.F. & J.R. Russell (1998): "Autoregressive conditional duration: A new model for irregularly spaced time series data." *Econometrica* **66(5)**: pp. 1127–1162.
4. Madhavan, A. (2000): "Market Microstructure: A Survey." *Journal of Financial Markets* **3**: pp. 205–258.
5. McNelis, P.D. (2005): "Neural networks in finance: gaining predictive edge in the market." *Elsevier, Burlington* .
6. Pacurar, M. (2008): "Autoregressive conditional duration models in Finance: A survey of the theoretical and empirical literature." *Journal of Economic Surveys* **22(4)**: pp. 711–751.
7. Sanchiz, A., G. Gutierrez, X. Li, & J. Peralta (2010): *"Time series forecasting by evolving artificial neural networks using genetic algorithms and differential evolution, "* WCCI 2010 IEEE World Congress on Computational Intelligence – July , 18-23, 2010 – CCIB, Barcelona, Spain.

Author Supervisor

Chapter 1

Introduction

The technological advancements in recent years have caused an enormous increase in the data availability. The data on trades are recorded up to the highest available frequency, which are individual transaction. The information about individual transactions allows us to predict future transactions and a crucial part of forecasting transactions is the timing of the next transaction, where durations are defined as the time interval between two transactions.

The interest in financial durations is caused by the influence durations have on the volatility and therefore the price creation process. The understanding of the impact and functioning of financial durations might be employed to model volatility and thus have far reaching impacts on risk management and high-frequency volatility trading. Furthermore durations are crucial for exploring the market microstructure in general.

An empirical fact of durations is medium memory, as Engle and Russel (1998) have discovered and subsequent studies have confirmed. The initial models on durations, such as the Autoregressive Conditional Duration (ACD) model, only had short memory capabilities, while later models were designed to incorporate long memory capabilities. However the medium memory property of durations was not accounted for in these models, which might indicate that these models are not able to model the characteristics of durations appropriately. Therefore no model currently exists to model the specific dynamics of medium memory in durations.

The objective of this thesis is the application of Artificial Neural Networks (ANN) to the field of financial durations, which to our knowledge has not been attempted. ANNs have been successfully applied to a range of areas, which

included complex non-linear systems. The reason for the application of ANNs, is the ability to approximate to arbitrary precision any function with finitely many discontinuities as was shown by Cybenko (1989) and Hornik, Stinchcombe and White (1989). Therefore we believe that they are able to adapt to the medium memory characteristics of durations and outperform standard econometric techniques. In addition to the approximation capabilities ANNs also have the desirable property of not requiring data pre-processing.

We use the basic ACD model of Engle and Russell (1998) to function as a benchmark for the ANN forecasts, because the model has widely been used as an indicator of the applicability of new models.

The first part of the thesis gives a literature review of ACD models and empirical findings for durations in general. Including the definition of duration models, extensions of the model as well as market microstructure findings based on the results of duration models. We then proceed to reviewing ANNs, where we show the range of possible methods and specifications. Chapter 4 deals with the data description and data pre-processing. In the following chapter we describe the methodology employed in the process of estimation, including the motivation for using certain procedures, such as the genetic algorithm and the selection method for the final model specification of the ANNs.

Finally we compare the performance of our ANN model with the performance of the traditional ACD model. The work is concluded by a summarizing the findings and suggesting future steps for the application of ANNs to duration modelling.

Chapter 2

Literature Review

The literature review is divided into two parts. The first part deals with the ACD model, its extensions and empirical findings in the field of durations, while the second part focuses on ANNs, the various specifications and their development over the past years.

In the ACD part of the literature review we first define a time process for durations, then a general group of duration models is introduced. Out of the group of possible duration models we describe the ACD model. We then review the extensions of the model including the use of marks in the models and other duration models are introduced for comparison. Lastly we describe basic testing methods, specifics of transaction data and stylized facts of trade durations.

2.1 Autoregressive Conditional Durations

As Engle and Russel (1998) pointed out the technological advances led to an increasing frequency of data being collected, this tendency stopped only after reaching the maximum level of detail available, which were individual transactions. As a result the data to be analyzed became irregularly spaced. However most econometric techniques were based on fixed time intervals, because the data analyzed historically were in the form of regularly spaced time intervals (e.g. daily, monthly or yearly data). Certainly the available data on the level of individual transactions could be transformed or aggregated into regularly spaced intervals, nevertheless this transformation would lead to a loss of information, as the microstructure of the markets would be removed or distorted. Therefore Engle introduced an alternative model, where the timing of individ-

ual transactions was treated as a random variable. The model worked with durations conditional on the past transactions and was called the Autoregressive Conditional Duration (ACD) model.

To fully understand the model we have to introduce the assumptions of a time process. A time process is an increasing function of transactions, where for every new transaction the time is higher than for the preceding transaction.

$$t_i \leq t_{i+1}, \forall i \in R$$

The point process we are working with can be defined as a stochastic process, which generates a collection of points on the time axis. Since the time process is irregularly spaced, we don not need any additional assumptions about the relation among the transactions. A duration is defined as the time between two transactions $x_i = t_i - t_{i-1}$. The durations have thus the property of being only non-negative values. Additionally we define a mark y_i as the set of available information for one specific transaction at time t (e.g. the volume or the price of the transaction). So for any given transaction we have the pair(x_i, y_i), denoting the duration and the mark respectively. We are estimating the probability that a transaction will occur at time t, while knowing all the times of previous transactions $(t_{i-1} \dots t_0)$ and the previously observed marks. Therefore we estimate the joint probability density of another transaction based on the complete previous information set. The information set includes all previous transactions. Previous transactions are denoted as $\hat{x}$ or $\hat{y}$ for durations and marks, respectively. We jointly estimate the future duration and mark as

$$f(x_i, y_i | \hat{x}_{i-1}, \hat{y}_{i-1}; \theta_f)$$

, where θ_f is the set of parameters of the selected distribution. Depending on the distribution the number of parameters varies.

The joint density can be rewritten to

$$f(x_i, y_i | \hat{x}_{i-1}, \hat{y}_{i-1}; \theta_f) = g(x_i | \hat{x}_{i-1}, \hat{y}_{i-1}; \theta_x) q(y_i | \hat{x}_{i-1}, \hat{y}_{i-1}; \theta_y)$$

,where the first term of the product is the marginal density of durations and the second term is the conditional density of marks, both conditionally based on previous observations.In the next section we restrict our attention to the marginal density of durations, because the standard ACD model used in this

thesis is restricted only to durations. For a complete description of the maximum likelihood estimation, including marks we refer the reader to the paper by Engle (2000).

2.1.1 Duration Model

The conditional expected duration is defined as

$$\psi_i = E(x_i|F_{i-1}) = \psi_i(\hat{x}_{i-1}, \hat{y}_{i-1})$$

,where F_{i-1} is the information set available at time t_{i-1}.

the model works with the assumption that the standardized durations

$$\epsilon_i = \frac{x_i}{\psi_i}$$

are independent and identically distributed, where $E(\epsilon_i) = 1$. ψ_i is constructed in such a way to ensure that the conditional expected duration is equal to one.

This leads to

$$g(x_i|\hat{x}_{i-1}, \hat{y}_{i-1}; \theta_x) = g(x_i|\psi_i; \theta_x)$$

The conditional expected duration, therefore includes all temporal dependence of the duration.

The above gives a number of specifications for a duration model based on the choice of the form of the conditional expected duration or different distributions of the innovations ϵ. Once we choose a parametric distribution the optimal parameter values θ are estimated by maximum likelihood.

2.1.2 ACD Model

The ACD model used in this thesis was proposed by Engle and Russel (1998) and is based on a linear parametrization of the conditional expected duration. The conditional expected duration ψ_i depends on m past durations and q past expected durations. The specification of the conditional expected duration is

as follows:

$$\psi_i = \omega + \sum_{j=1}^{m} \alpha_j x_{i-j} + \sum_{j=1}^{q} \beta_j \psi_{i-j}$$

, where ω, α_i and β_i are parameters to be estimated.

This is referred to as an ACD (m,q) model. In order to ensure the existence of positive durations, the required conditions are $\alpha, \beta \geq 0$ and $\omega > 0$. The model works with the assumption that the standardized durations

$$\epsilon_i = \frac{x_i}{\psi_i}$$

are independent and identically distributed, with $E(\epsilon_i) = 1$ and have a distribution with positive support. Similarly as for the GARCH model the conditions for ACD model to be covariance-stationary are

$$\sum_{j=1}^{m} \alpha_j + \sum_{j=1}^{q} \beta_j < 1$$

The ACD model is an analogy to the GARCH model used for modeling equally spaced data in financial time-series analysis. Similarly as the GARCH allows for conditional variance of returns, the ACD model allows for conditional mean in durations. As a result both models are able to model clusters of high/low volatility respectively short/long durations. Clustering in the context of durations describes a situation where short durations are followed by short durations, or long durations are followed by long durations. This suggests that there is some dependence among observations that has to be taken into account, for local variations of the model. Both GARCH and ACD allow us to incorporate information from past observations in order to vary volatility or the mean of durations over time. Therefore we are able to remove the temporal dependence in durations. The situation where clusters occur has been empirically proven, an example of this situation is the availability of new information. The new information was not included in the price before; consequently new transactions will be executed in order to adjust the price according to the new information. Information in the above mentioned sense can also take the form of an event.

Most ACD models work with a standard exponential distribution, which leads to the so-called EACD model. The advantage of using an exponential distribution lies in the fact that it provides a quasi-maximum likelihood (QML)

estimator for the ACD parameters. However for obtaining consistent estimates of the ACD model the conditional expectation of durations ψ_i has to be specified correctly. As we have no knowledge of the correct distribution we are limited to testing if the chosen distribution is correct only after estimating the model. Even though QML estimates are consistent they are not as efficient as ML estimates. Furthermore an exponential specification implies a flat conditional hazard function which is restrictive and has been rejected in empirical applications (Engle and Russel (1998)). A number of other distributions have been used to allow incorporating some specific behavior observed empirically, such as the Weibull distribution, the gamma distribution and others. In the next section we review the advancements of ACD models, since the introduction of the model in 1998.

2.1.3 ACD Extensions

The basic model has been extended in many ways. Most of the extensions are based on extensions made to the GARCH model, as empirical results have often encountered similar imperfections in both models. A brief review of some extensions is covered in the following section.

Jasiak (1998) proposes the Fractionally Integrated ACD (FIACD) model, which is an analogy to the FIGARCH model of Baillie, Bollerslev, and Mikkelsen (1996). The model builds on the Integrated ACD model (IACD), the specification of the FIACD model ensures strict stationarity and ergodicity. The model allows for modelling long-memory behavior, which is an empirically proven characteristic of duration data. As the classical ACD model requires certain conditions on the parameters to ensure the positivity of duration (as mentioned before), Bauwens and Giot (2000) deal with this issue by constructing a more flexible logarithmic-ACD model. The model is specified on the logarithm of the conditional duration ψ_i. This model ensures that the non-negativity conditions of the parameters in the conditional duration are not needed. Furthermore it allows for nonlinear effects on short and long durations without a need for additional parameters.

Fernandes and Grammig (2006) on the other hand prefer an EXponential ACD (EXACD) model to deal with short and long durations. The model is specified similarly to the EGARCH model and allows for asymmetric impacts

for situations when the durations deviate from the conditional mean. The asymmetric specification solves the problems of short and long durations, however as the impact of short durations can differ from long durations a model, which treats short and long durations differently was introduced by Zhang, Russell and Tsay (2001). The threshold ACD (TACD) model is designed in a way to enable settings, where a number of regimes are introduced to treat durations differently based on past durations and to allow for different intensities associated with various movements. Therefore parameters in the equations differ in relation to the regime they are in. The challenge of such a regime switching model is to choose the appropriate number of regimes and to update the values in each regime over time. This means that the regimes are subject to changes over time based on some economic events; Meitz and Terasvirta (2006) therefore propose a model where the parameters of the ACD model change smoothly over time, to allow the model to adjust to the changing conditions. The logistic function is used as the transition function with the time as a transition variable.

2.1.4 ACD Model Extensions to Marks

The above mentioned extensions of ACD were based on predicting only durations. However as already Engle and Russel (1998) showed in their initial paper on ACD, the models or rather the data can be transformed in such a way, that we are not limited to only durations, but we are able to predict other factors such as prices or volumes of upcoming transactions. The motivation behind the prediction of other factors is that there is expected to be a link between the information contained in marks and the durations. Furthermore the prediction of when a transaction will occur is only a part of the information. Similarly as the ACD model is motivated by including all available data to make good forecasts and thus works with irregularly spaced events, it is rational to use all available data to make predictions. In the case of durations we use the marks and past durations to make predictions, however without predicting future values of marks more than one-day ahead forecasts are virtually impossible.

The motivation behind predicting marks is not only for the sake of better predictions; it is also for the testing of market microstructure theories.

The first joint model was developed by Engle (2000), the so-called Ultra-

High-Frequency GARCH model. The model worked with ACD for durations and with GARCH for prices. The GARCH model was adapted to irregularly time-spaced data, therefore it measures the volatility per unit time. The ACD model is estimated first and then the GARCH model uses the past and predicted durations to model volatility together with other explanatory variables. However the mentioned method does not take into account the impact of volatility on durations. The model has been extended by Grammig and Wellner (2002) to allow for interdependence of volatility and durations, the so-called interdependent duration-volatility (IDV) model. Other approaches to modeling jointly marks and durations include vector autoregressive (VAR) models or models that are based on discrete variables, such as the autoregressive conditional multinomial (ACM) model.

2.1.5 Competing Duration Models

The Long Memory Stochastic Duration (LMSD) model was introduced by Deo, Hsieh and Hurvich (2006). It extends the Stochastic Conditional Durations (SCD) model introduced by Bauwens and Veredas (2004). The SCD model is given by

$$x_i = \epsilon_i e^{\psi_i}$$

$$\psi_i = \omega + \beta\psi_{i-1} + \upsilon_i$$

,where ϵ_i and υ_i are mutually independent and identically distributed innovations, υ_i have zero mean and ϵ_i have a unit mean and positive support. The parameters ω and β are to be estimated, where $\omega \in \mathbb{R}$ and $|\beta| < 1$ to ensure weak stationarity. In comparison to the ACD model, where we have one observable random variable in the case of the SCD model we have two random variables one is observable and one is latent. The latent random variable is part of a AR(1) process, which influences the conditional duration. Bauwens and Veredas (2004) show that while an increase in dispersion in the ACD model (by increasing the parameter α_i) leads to a decrease in the ACF, the same does not apply for the SCD model, because the dispersion parameter is independent of the ACF rate for SCD. Therefore the SCD model allows a higher dispersion, while not affecting the persistence. The SCD model in general enables more complex structures.

Similarly as the ACD model the SCD model belongs to the group of short

memory models, because the decay of the ACF is geometric. However with the extension of Deo, Hsieh and Hurvich (2006) it becomes a long memory process. The model takes the form of

$$x_i = \epsilon_i e^{\psi_i}$$

$$\psi_i = \omega + \beta\psi_{i-1} + (1-L)^{-d}\upsilon_i$$

,where L is the lag operator and $d \in [0, 0.5)$. The AR(1) process is changed into an ARFIMA process with increased persistence.

Estimation of the SCD and LMSD is difficult, because the likelihood function is difficult to evaluate for models with unobservable variables. The suggested method by Bauwens and Veredas (2004) is a QMLE together with the Kalman filter after adjusting the model to a linear state space representation. The estimates are asymptotically consistent, but not efficient. On the other hand Deo, Hsieh and Hurvich (2006) suggest a QMLE with the use of a Whittle aproximation.

2.1.6 Duration Model Testing

While the extensions of the ACD model have received a lot of attention, the evaluation of the models has somehow been neglected. In general there are three types of tests based on the area they are testing. The first types of tests are tests based on evaluating the properties of the estimated standardized residuals of an ACD model. We work with the assumption that the standardized residuals are identically and independently distributed, if the estimated model fulfills this assumption we can assume that the model is adequate. The most widely used test for residuals in the area of ACD models is the Ljung-Box Q-statistic, to check for remaining serial dependence. This approach was also used by Engle and Russell (1998). Other tests, including the portmanteau test, are tests based on comparing marginal densities of durations from the model with marginal densities observed from the durations or test based on testing the moment conditions of the estimated residuals.

The second types of tests are tests which work with the functional form of the conditional mean duration. The reason why we want to achieve the correct functional form is to ensure that our maximum likelihood estimates will be

efficient. These include tests of no remaining ACD effects, usually Langrange multiplier tests. Other possible tests include tests based on analyzing the spectral density of the estimated residuals.

The final types of tests are test based on examining the distribution of the error term. Since most models work with an exponential distribution, because it leads to consistent QML estimates, it is necessary to test if this specification is correct. Fernandes and Grammig (2005) introduced two tests for the distribution of the error term (the so-called D-test and H-test). The tests are based on comparing parametric and nonparametric estimates of the density of the hazard rate function of the estimated residuals, respectively.

2.1.7 Transaction Data for Duration Models

The problem with the data for durations is that it is difficult to remove the intraday patterns present in the data. Therefore we have to adjust the transaction data in order to remove these intraday patterns. The intraday patterns are based on traders' habits or the availability of information. The frequency of trades is higher at the beginning and the closing of the trading day than during the day as reported by Engle and Russell (1998). This is caused by the fact that at the beginning trading takes place in order to include new information, while at the closing traders want to close their positions. Also other habits such as less activity during lunch or the influence of opening and closing of other markets are present in the data. Therefore most authors decompose intraday durations into a deterministic part based on the time of the day of the transaction, and a stochastic part for modeling the dynamics of the durations.

The ACD models are estimated after adjusting the data and removing the seasonalities. The seasonal factor is obtained either by regressing durations on the time of the day and then taking the ratios of durations to fitted values, or estimating the factor as the expectation of duration conditioned on the time-of-day and smoothing the time-of-day function (the intraday seasonal factor is estimated for each day of the week). Bauwens and Verdas (2004) use a nonparametric regression to estimate the seasonal component of price durations, for each day of the week. These adjustments to transaction data cause the estimation process to consist of two parts the diurnal adjustment of the data and the subsequent estimation of the data. The alternative is to estimate

the seasonal parameters jointly by including the seasonality into the duration model.

Another issue with transaction data is how to work with transactions that occurred at identical times. Usually observations with zero duration are either omitted or they are included as a single observation. This means that they are aggregated or averaged out. Engle (2000) removes transactions at identical times, because the definition of a trade is the transfer of ownership from one or more sellers to one or more buyers at a point in time.

2.1.8 Stylized Facts of Trade Durations

The introduction of models of durations has led to the testing of hypothesis on market microstructure. The following stylized facts of trade durations have been confirmed in papers.

Clustering - the phenomenon of clustering has been proven in empirical literature (see, for example Engle and Russell(1998) or Jasiak (1998)). As was already mentioned, this means that either long durations are followed by long durations or short durations are followed by short durations. Therefore there is a positive autocorrelation with previous observations. The existence of clustering for durations supports the use of ACD for modeling durations, as it allows for conditional duration. Market microstructure explains the phenomenon by assuming that there are two or more types of traders. Informed traders (trading based on a superior information set) and not informed traders (liquidity traders) or rather traders not trading based on a superior information set (there can be more types of traders that belong to the uninformed trader group. Even though we can not observe the type of individual traders, we know the composition of the traders in the market. Based on new information informed traders adjust their bid and ask quotes. According to Easley and O'Hara (1992) informed traders only trade when new information enters the market. Thus after the release of new information a number of transactions takes place, because the information becomes available to a higher number of traders (uninformed traders are able to learn the information by observing the actions of informed traders), who change their prices according to the information. An alternative explanation for the cause of clustering is the introduction of another uninformed trader. Thus non-discretionary and discretionary uninformed traders exist, where discretionary traders select the timing of their

transactions (even though they are uninformed). In the setting of the model Admati and Pfleiderer (1988) show, that the optimal behavior is a clumping behavior. Therefore discretionary traders select the same period for transacting in order to minimize the adverse selection costs and informed traders follow the pattern introduced by the discretionary traders.

Overdispersion of durations- the majority of papers report that trade durations are overdispersed, meaning that the standard deviation is greater than the mean. The fact of overdispersion suggests that the exponential distribution is not appropriate for the unconditional distribution of trade durations (reported in Bauwens and Giot (2001) or Engle and Russell (1998)). However this does not imply that conditional duration can not be modeled using the exponential distribution.

Existence of zero trade durations- the existence of zero trade durations is caused by the fact that the highest precision of the timestamp attached to each transaction is a second. Therefore no durations smaller than a second are recorded and all transactions, that take place within a second are reported as occurring at the same time. The usual approach to deal with these zero durations is to aggregate the transactions and all transactions with the same time are then discarded (Engle (2000) includes only unique transaction times in the analyzed sample). The justification of the method lies in the microstructure argument that simultaneous observations correspond to split-transactions, which means that large orders are broken down into smaller orders to facilitate faster execution. However some authors argue that multiple transactions contain some information and replacing them with a single transaction leads to a loss of information. The pace of the market is deformed by ignoring those transactions. For example Zhang, Russell, and Tsay (2001) incorporate the information about multiple transactions into their TACD model through a lagged indicator variable.

Relation to Volatility in Returns - Durations influence counts, because if durations are shorter the number of transactions is higher and therefore the number of transactions within a given time-interval (counts) increases. Similarly the number of counts decreases when durations are longer. Furthermore counts have an impact on volatility in returns. Ultimately this means that durations influence volatility. Deo, Hsieh and Hurvich (2010) have shown that

long memory in volatility is a result of long memory in durations. These findings raise the importance of understanding durations, because they influence volatility, which in term is an important part of the price creation process.

2.2 Artificial Neural Networks

Artificial neural networks (ANN) have been used widely in recent years to model various types of data across many fields. The reason for the interest in ANN is the fact that they are less sensitive to error term assumptions, they can tolerate noise, chaotic components, and heavy tails better than most other methods, as stated by Masters (1993). ANNs are a very flexible method to model dependencies, whether they are linear or not. In recent years the improvement in the performance of computers has made the application of ANNs possible, because previously, the quiet simple method of ANNs required a number of calculations during the optimization algorithm, which were technically not feasible.

The first successful attempt to use ANN for forecasting was achieved by Lapedes and Faber (1987),(1988). They used two chaotic time-series generated by the logistic map and the Glass-Mackey equation. A designed feed-forward network was able to accurately predict these dynamic nonlinear systems. The results proved that ANNs could be used to forecast nonlinear time-series with high precision. Based on the results of Lapedes and Faber a number of papers studied the application of ANNs to chaotic time-series, mostly in the field of physics and engineering, where many phenomena are generated by nonlinear chaotic systems. Later ANNs were used in financial applications ranging from forecasting bankruptcy and business failure (Odom and Sharda, 1990), foreign exchange rate forecasting (Borisov and Pavlov, 1995) to stock prices predictions (Kaastra and Boyd, 1995). The trend of applying ANNs for forecasting continued, especially with the advancements made in the optimization techniques of ANNs.

The most recent trend is the design and application of hybrid models including ANNs. Atsalakis and Valvanis (2009) use an ANN with fuzzy system logic to successfully predict next day's stock price trends, while Wang (2009) uses a model combining ANNs and GARCH to forecast stock index option prices and the model is able to outperform the traditional GARCH model.

The advantage of combining various models and creating hybrid models is the possibility of combining the strengths of individual models and reducing their weaknesses. Often hybrid models use one optimization technique at the beginning and another for later parts of the optimization, because for example some techniques are good at determining the region in the search space where the optimal solution exists, however they fail to converge to the optimal solution in this reduced region. On the other hand for example hill-climbing algorithms perform very well in finding an optimal solution, but they are limited to local optima in case of complex search spaces and because the initial position in the search space is random we have no certainty about the performance. In combining those characteristics we enable the model to reach a region where the local optimal solution is also the global optimal solution before starting the hill-climbing part of the algorithm.

The following sections start with a definition of the key component of ANNs a single neuron and the signal transfer within a network. We then continue with the construction of a complete network and we review the possible fundamental parts of an ANNs such as activation functions, error functions and optimization algorithms.

2.2.1 Neural Network Definition

The term Artificial Neural Networks stems from the use of the biological concept of neurons, where a neuron is able to receive impulses from connecting neurons, transform the impulses and send them on, if a certain activation value is exceeded. Based on these assumptions a set of neurons is constructed with connections between neurons, where each of the connections is assigned a weight. A neuron x_i connected to a neuron x_j will have a weight w_{ij} assigned to the connection.

All inputs for the given neuron are multiplied by their respective weights, summed up and then transformed using an activation function. Activation functions usually restrict the range of values of the neuron output, for example from minus one to one. This transformed value then serves as the input for the next neuron. The above mentioned procedure can be defined as

$$s_i = \sum_{j=1}^{n} w_{ij} x_j$$

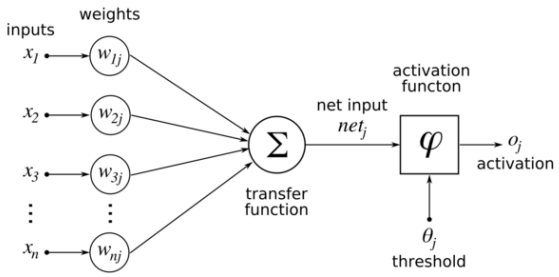
,where $x_1 \dots x_m$ are the neuron inputs $w_{i1} \dots w_{im}$ and s_i is the linear combination of the inputs.

The expression s_i describes the input for a single neuron before passing through the activation function φ.

$$y_i = \varphi(s_i)$$

The activation function then produces the output y_i, which can serve as an input for the same process in a different layer of the network.

Figure 2.1: Signal transfer in ANNs



A neural network consists of three types of layers-input, hidden and output. The input layer contains the data supplied to the network. In the hidden layers the transformation of the inputs takes place and the output layer produces the estimate. The number of hidden layers is not limited, however (Cybenko, 1989; Hornik, Stinchcombe and White, 1989) have shown that a single hidden layer is sufficient to approximate to arbitrary precision any function with finitely many discontinuities, given the activation function is non-linear. Nonetheless some authors use ANNs with up to 8 hidden layers. The disadvantage of using many hidden layers is that the selection of initial parameters and the correction becomes complex. The usual method of determining the problem in the output layer is to analyze the output of the previous layer. In the case of many hidden layers this process is very time demanding.

2.2.2 Network Architecture

The general setting described in the previous section allows for a number of different ways to construct an ANN to model the same data. If we focus only on the network topology we can change the number of inputs, hidden layers, hidden neurons, output neurons and weights. Furthermore we choose from a variety of activation functions, error functions and optimization algorithms. The number of possible network specifications is vast and we have no prior information to restrict the set. If we further take into account the data we can either use the data raw or transform them the number of combinations once again increases. Although the effect of data pre-processing for ANNs is not clear and in general it is not necessary to pre-process the data. Nonetheless some authors argue that pre-processing the data improves the forecasting capabilities.

The main advantage of the ANNs, their great flexibility, is also a complication in the sense that we have too many options to explore and no indications on a correct specification. The process of finding the correct specification is usually trial and error and iterating through all possibilities is virtually impossible, due to time restrictions. Therefore most authors choose a certain method and then they search for the best performing specification within the given method, thereby restricting the number of possibilities, but still providing a certain range for fine tuning of the networks. As a result of this method the poor performance of ANNs on a certain modelling task can be caused by choosing a wrong type of methods for the given problem rather then the inability of ANNs to model the problem. For a detailed discussion of the correct network architecture we recommend the paper by Kaastra and Boyd (1996).

The following sections focus on individual parts of the ANNs. We first review the network topology and some of the fall backs of the flexibility of ANNs. The next section describes various types of activation functions, learning algorithms and we describe in more detail some of the optimization methods available for ANNs.

2.2.3 Network Specifications

A network specification is a setting with a set number of neurons in each layer. We treat two networks that have the same number of neurons in each layer as identical if they also have the same connections between neurons. The standard procedure for selecting the best network specifications is to set a maximum number of neurons for each layer and to iterate through all specifications up to the given maximum value. It is irrelevant if we always choose the same input neurons for the same input variables as long as the neurons have identical connections then the result of the training procedure remains identical (i.e. the predictions and weights will be identical only the weights will be between different neurons). The notation for network specification in this thesis will have the standard form of notation for multilayer neural networks '"i-h-o"', where i denotes the number of input neurons, h the number of hidden neurons and o the number of output neurons (e.g. 4-2-1). Yao(1999) states that too trivial networks might not be able to process the data, while too complex networks will have redundant connections.

The interpretation of the significance of connections for non-linear activation functions is not possible, we are simply not able to determine the impact of the weight value on the signal. Therefore the connections are created based on some simple rule and the connections are not deleted even when they seem redundant. The idea behind leaving weights that carry little or no information is that during the training process these weights will be given values that have almost no impact on the connecting neuron, so rather then deleting some significant connection we let the network decide. Teng and Wah (1996) suggest creating a fully connected network first and then removing connections to identify redudant weights. Networks that are created with all connections between adjacent layers are called fully connected, while networks without some connections are called partially connected. The fall back of not removing any weights is an increase in complexity and an associated increase of the computational burden.

There are two basic types of networks based on the logic of connections that they have. These are feed-forward and recurrent networks. The main difference is the direction of connections. For feed-forward networks the signal can only pass to the next layer, while in recurrent networks the signal can be passed to

any layer.

Feed forward networks - feed forward networks are designed in such a manner that connections lead only to the next layer. Therefore in a feed forward network the signal is transmitted from the input layer to the first hidden layer and so on till the nth hidden layer and lastly to the output layer. In the estimation of the output only information from the previous layer is used.

Recurrent neural networks - in contrast to feed forward networks, where the outputs of the previous neuron were only used as inputs for the next one, recurrent neural networks allow the output from a neuron in a higher layer to be used as input for a neuron in a lower layer. This type of setting is useful when we are working with lagged values of some sort, where the calculation of a previous value can serve as the input for the following value.

2.2.4 ANN Activation Functions

Several activation functions exist and they also influence the performance of the ANNs. Perhaps the simplest activation function is *threshold function*. If the signal from a neuron is larger or equal to a certain set value then the output value is equal to 1 else the output value is equal to zero. This type of signal is good for binary problems, but for continuous values we lose information.

The extension of the threshold function is a *piecewise-linear function*. Similarly as in the previous case above a certain value the output is 1 and below a certain value the output is 0, however in between theses values the output is equal to the input. It is a threshold function with a linear part rather than a sudden jump.

Probably the most widely used activation function is a *sigmoid function*, defined as

$$\varphi(x) = \frac{1}{1 + exp(-ax)}$$

,where a serves as the slope parameter. The sigmoid function is differentiable and assumes a range of values from 0 to 1.

For other choices of the activation function, see Chen, Racine and Swanson (2001).

2.2.5 Network Learning

In general ANNs learn from the data and they are able to improve in iterations by adjusting the weights, thereby identical input signals are transformed differently by the network during each iteration, this defines the process of learning for neural networks. There are 2 basic categories of learning: supervised and unsupervised. During supervised learning the networks receive some sort of feedback to determine if the output was right or wrong and in numerical cases how far the estimated value was from the original value. Supervised training can either be automatic or a person can evaluate the output and provide the neural network with the feedback. The difference between the estimate and the original value is the error term. Then in subsequent iterations the values of weights change in order to improve the performance, thus reducing the error term. The learning part for ANNs then terminates based on some performance criteria or number of iterations. The usual stopping criterion is a percentage improvement of the error term. When the error term improves by less than the set value the process is terminated. The most well known supervised learning method is backpropagation. The principle of backpropagation is described later in the chapter.

Unsupervised learning on the other hand has no feedback about the quality of the predictions. The aim is to find a structure in the data by identifying similarities and extracting the underlying pattern. An example of unsupervised learning are self-organizing maps. The aim of self-organizing maps is to reduce the dimensionality of the data and to cluster them in a two-dimensional space based on similarities. Most learning methods divide the data into two or three parts. The first part is the training data, second validation data and the third is testing data. Sometimes the validation data are left out.

In the case of durations we are training the network on historical values, where we know the future value, therefore the method used in this thesis is supervised learning.

2.2.6 Network Optimization Algorithms

All optimization algorithms in ANNs try to determine the optimal weights with the minimal amount of computation necessary. The problem in general can be described as searching for a global optimal value in a multidimensional space, in order to minimize the error term.

ANNs are able to model anything, however therefore the danger of over fitting exists. In other words if we try to fit the data with a high number of iterations the ANN forecasts deteriorate after a certain rate. This is given by the fact that the model is over trained to fit the training data and does not perform well in the validation or test afterward. A number of supervised optimization algorithms are reviewed in the following paragraphs.

The first method is called backpropagation. The method uses a standard forward movement to calculate the estimated value, without changing the weights of the neurons. Once the output layer is reached a backward movement starts with the information about the error, which updates the weights based on some specific rule. This calculation is done for each neuron updating the weights in the process in order to find the optimal weights to minimize the error function of the ANN. A number of algorithms can be used for backpropagation learning., A few examples are provided in the following paragraphs.

Gradient descent- gradient descent is a method where the update of weights is based on the direction of the steepest gradient of the error function. The weights are updated in the direction which causes the biggest improvement of the error term, thus gradient descent, because the algorithm moves the weights in the direction of the steepest increase. The problem of gradient descent or backpropagation in general is the possibility of attaining a local minimum of the error function rather than a global optimum. This is caused by the fact that the first iteration is based on random values, which are updated afterward, so depending on the starting point the algorithm will reach different local optimal values. The gradient descent method is therefore guaranteed to find the global minimum if there is only one minimum for the given error function. For other situations we are not able to ensure that the optimization algorithm resulted in globally optimal values for the weights. To be able to employ the gradient descent algorithm we require a differentiable activation function, such

as the sigmoid activation function. However the fallback of differentiable activation functions is the increased likelihood of local minima compared to for example step functions.

Quasi-Newton methods-The update of weights is determined similarly as in gradient descent, however information about the second moment is used in the update. Therefore the update of weights is based on the product of the local gradient vector and the inverse of the Hessian matrix. This is called the Newton method, however since the computation associated with the inversion of Hessian matrix is large, the quasi-Newton method is used instead where we only use a positive definite estimate of the inverse of the Hessian matrix. The disadvantage of this method is the computational complexity, it is therefore used to train less complex ANNs.

The following algorithms belong to the group of metaheuristics.

Genetic algorithms- in contrast to backpropagation algorithms this algorithm belongs to the group of metaheuristics. Metaheuristics are iterative processes which guide subordinated operations in order to find near optimal solutions. Genetic algorithms are based on the principles of biological evolution or rather the principal of the survival of the fittest. The process takes place in three steps selection, reproduction and replacement. During each iteration new individuals are created (a process called breeding) from randomly selected existing ones by either mixing some characteristics or randomly changing some characteristics. Then the new population is compared with the existing population and the fittest individuals are kept, while the other ones are discarded. Therefore any new generation can not be worse than the previous one. Characteristics of individuals are called chromosomes or genes depending on the level of detail the characteristics are divided into. In the context of neural networks a gene can be considered a weight. The convergence process to a global optimum depends on the number of iterations or new generations and the extent of the original population. The overall advantage of metaheuristic procedures is the ability to find global optima rather than local ones, which results in outperforming algorithms such as the gradient descent method, where the algorithm has no option of attaining the global optimum once the process starts near a local optimum.

The problem with genetic algorithms is correctly determining the initial parameters, such as the size of the initial population the number of iterations and the range for random values. However genetic algorithms are in general designed to produce good estimates even with poor parameters. The disadvantage of bad initial conditions is a larger number of iterations to converge to the optimal solution.

For a detailed review of evolutionary algorithms in economics we recommend the work by Safarzynska and van den Bergh (2010).

Simulated annealing-the methods name and principle comes from metallurgy, where it is a technique of heating the metal and then controlling the cooling of the material to increase the size of its crystals and reduce their defects. The properties of the material depend on factors such as the initial temperature and the cooling rate. The aim of the method is to get the system into a state with minimum required energy and minimal defects, thus avoid a premature convergence to local optima. In the sense of ANNs a local optimal solution is a crystal with malformations, while the global optimal solution is a perfect crystal. We start with an initial position and at each iteration the algorithm computes a random nearby solution, better solutions are always accepted (solutions with a lower error term), while worse solutions are accepted in case of some existing probability to attain a better state in the next iteration. The fact that the algorithm accepts even deterioration allows it to escape from local optima and to reach a global optimal value. The acceptance of worse motions is based on a distribution function and it decreases over time.

Particle Swarm Optimization - The Particle Swarm Optimization (PSO) algorithm is based on the movements of swarms of social insects (e.g. insects that live in larger groups or colonies), birds or for example fish. In general we have a number of particles in the parameter space of some function, for each of the particles we compute a fitness values based on some error function. So far the algorithm is similar to creating a initial population for a genetic algorithm, however in the next step instead of breeding new individual or in this case new particles, we determine the movements based on the history of fitness values for the given particle and the history of fitness values for one or more other members of the swarm. Therefore the size and direction of the change are determined not only by the individual, but partially by the group.

The members of the swarm are able to interact with their neighbors and they create a so called social neighborhood. If we combine the social neighborhoods of all particles we get a PSO social network. The social interaction in the system enables the particles to move in the direction of the best location so far encountered by any particle, while still taking into account the effectiveness of the past movements of the individual particle.

For a detailed review we suggest the analysis of PSO applications by Poli (2008).

Differential evolution - In differential evolution we take an initial random sampling of individuals, generated by a distribution function within the search space. In the next step three mutually distinctive individuals x_m,x_n,x_o are selected at random from the current population and new offspring is generated.

$$x_i = x_m + F(x_n - x_o)$$

,where F is a positive scaling factor, which determines the distance of the exploration vector $(x_n - x_o)$.

The offspring is temporary in the mutation phase of differential evolution and replaces an existing parent individual only if the new individual outperforms the parent in a fitness test based on a error function. The method of combining existing individuals has many variations, which can include more than two individuals or the best individual in the current population. In general in differential evolutions new individuals are created from existing ones by mutation, generating a new vector of possible values, therefore enabling the algorithm to search the whole solution space. The algorithm explores a large space at the beginning and in later stages the individuals converge to a small solution space. The problem with differential evolution is that the number of possible combinations is limited and if there is no possibility of creating a new better solution (individual) then stagnation occurs and the algorithm does not converge to a solution. For a more detailed description of differential evolution we refer to the work of Neri and Tirronen (2010).

Hybrid algorithms - Hybrid algorithms in general try to combine the benefits of different algorithms in various stages of the optimization. For example genetic algorithms can be used in the first stages of the iteration to

ensure that we can avoid local optima, however after a certain number of iterations gradient descent algorithms are more effective in finding the optimal values. Another hybrid model combines the Hidden Markov Model (HMM) with a neural network with a genetic algorithm. The neural network served for data transformation and optimizing the initial values of the HMM. The model outperformed the traditional HMM as well as a ARIMA model or ANN models. The variety of hybrid models is extensive, because the number of optimization methods is vast and most of these methods can be combined in some way. Furthermore methods outside the ANN framework can be used as well.

2.2.7 Performance Evaluation

The performance evaluation for ANNs with supervised learning is done by various ways of evaluating the error term. The error term is defined as the difference between the estimate and the actual value. The basic performance evaluation methods are mean squared error (MSE), mean absolute error (MAE) and root mean squared error (RMSE).

Mean square error is calculated as

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(o_i - x_i)^2$$

,where o_i is the observation and x_i is the estimate.

Root mean square error is calculated as

$$RMSE = \sqrt{\frac{\sum_{i=1}^{n}(o_i - x_i)^2}{n}}$$

,where o_i is the observation and x_i is the estimate.

Mean absolute error is calculated as

$$MAE = \frac{1}{n}\sum_{i=1}^{n}|o_i - x_i|$$

,where o_i is the observation and x_i is the estimate.

Chapter 3

Data Description

The data consists of price durations of three major foreign exchange (FX) futures contracts traded on the Chicago Mercantile Exchange (CME). The data sets consist of transactions of the Swiss Franc (CHF), Euro (EUR) and Japanese Yen (JPY) between 9 November 2009 and 29 January 2010. The data is supplied by TickData, Inc. The analysis is restricted on the main trading hours of 7:20-14:00 Chicago time. US and UK Bank holidays are discarded.

The data consist of individual transactions, however we are interested in price durations, which are defined as the minimum time it takes for the price to move by a certain amount. Therefore in order to extract the price durations we apply a process called thinning. The process of removing transactions can lead to a loss of information, because we are removing observations and not taking into account all available data, on the other hand price durations might be more informative, because the thinning process reduces the distortions due to microstructure noise, furthermore we remove duplicate prices, as these are transactions with a similar price, therefore a zero price change.

We construct the price durations by measuring the minimum time required for the futures price to move by at least c, starting from the first transaction of the day and discarding overnight durations. The FX futures are highly liquid and usually trade with a tight bid-ask spread of 1-2 ticks, where the tick size equals 0.0001 for CHF, EUR and 0.01 for JPY. To eliminate price changes caused by the bid-ask bounce, we set c=0.0003 for CHF and EUR and c=0.03 for JPY. To enable a simple comparison across currencies we work with the first 12,000 price durations for each FX futures contract in our sample period.

Table 3.1: Descriptive statistics

	CHF		EUR		JPY	
	raw	adj	raw	adj	raw	adj
Mean	105.60	1.0005	91.621	1.0015	74.9111	1.0004
Median	59.000	0.6031	52.000	0.6096	37.00	0.57124
Minimum	1.0000	0.00549	1.0000	0.00653	1.00	0.00442
Maximum	2703.0	19.841	2271.0	24.803	1922.00	17.911
Std. Dev.	142.25	1.2451	121.31	1.1957	112.87	1.3143
C.V.	1.3470	1.2444	1.3240	1.1939	1.50667	1.3138
Skewness	4.0586	3.6422	4.0655	3.6951	4.57785	3.7323
Ex. kurtosis	31.310	22.967	30.928	28.936	36.7713	23.518

Therefore the number of price durations is fixed while the sampling period changes based on the volatility of the given currency, the date shared among the samples is the starting date of 9th November 2009.

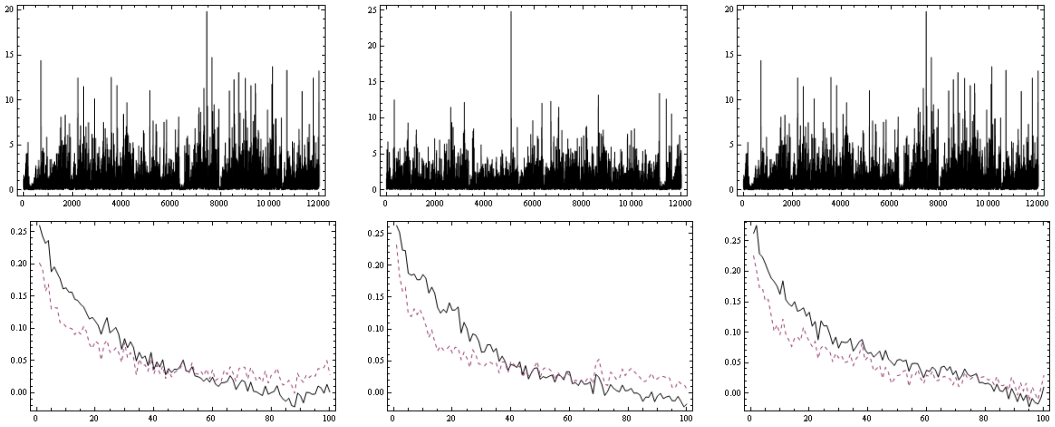
The mean of the price durations is 106s, 92s, 75s for CHF, EUR and JPY, respectively. With the median being 59s, 52s, 37s, respectively, suggesting that the distributions are positively skewed. The minimum is equal to 1 for all currencies, whereas the maximum is 45 minutes, 38 minutes, 32 minutes, respectively. In line with previous findings the data confirm the existence of over-dispersion in price durations.

We mentioned before that a problem with duration data is the presence of intraday patterns, which is caused for example by traders' habits and the availability of information. Theses factors cause among others a higher frequency of trades at the beginning and the closing of the trading day. To be able to correctly model the durations the seasonal components in the data have to be removed. A non-parametric regression (the Nadaraya-Watson estimator) is applied to estimate the seasonal component of the price durations. The estimation is done seperately for each day of the week as in Bauwens & and Veredas (2004).

The adjusted data have, by construction a mean close to one, the median significantly below the mean, while the over-dispersion of the data has been slightly reduced. As figure 3.1 [1] shows the auto-correlation functions of the adjusted data have lower values for initial lags and higher values for lags exceeding 60, the overall persistence remains similar.

[1] All figures having multiple graphs show the data sets in the order CHF, EUR and JPY

Figure 3.1: Standardized durations and autocorrelation functions



The top row shows standardized durations and the bottom row shows autocorrelation functions of raw and adjusted data. The dashed line depicts the adjusted data.

The ACF functions of durations data are a medium memory process, while the original ACD model is a short memory model. Therefore the model is not able to capture the longer memory characteristics of durations data. The insufficient memory length of the ACD led to the FIACD extension. The FIACD model is a long memory model, as well as the previously described LMSD model. However none of these models is designed for the medium memory present in the data. Therefore we propose the use of ANNs, because due to their flexibility they may be able to adapt to the persistence of durations. Furthermore in comparison with all the mentioned models ANNs are able to model price durations without the need to diurnally adjust the data. Although preprocessing of the data can sometimes improve the performance of the ANNs it is not required. If the ANN model is successful, the standard two step procedure of estimating the seasonal component and subsequent estimation of price durations might be replaced by a single ANN estimation. The advantage of using a single estimation is also the fact that we do not distort the results. For example Meitz and Terasvirta (2006) point out that the impact of deseasonalisation is not clear and they suggest that it should receive further attention.

Chapter 4

Methodology

4.1 ANN Methodology

In our paper we use a feed-forward network with one hidden layer and a sigmoid activation function. The reason for using only one hidden layer is, that it has been shown by (Cybenko, 1989; Hornik, Stinchcombe and White, 1989) that it is sufficient to model any non-linear space of solutions, if we use a non-linear activation function. The sigmoid activation function fulfills the requirements of being a non-linear transformation. The network is a completely connected network, which means that all neurons from the previous layer are connected to all the neurons from the next layer, therefore the number of weights is always determined by the following formula:

$$n = ih + ho$$

, where n denotes the number of connections, i the number of input neurons, h the number of hidden neurons and o the number of output neurons.

The problem with removing the connection between two neurons is that as the relationship is not linear (such as for example in ordinary least squares), therefore we have problems determining which connections are insignificant, especially for large networks as Yao(1999) states.. We are going to omit the removal of any weights in the ANN with the reasoning, that if the weights are insignificant they will not influence our forecast. During the training of the neural network the weights obtained for these insignificant connections will get values, which will have little to no influence on the forecast.

The number of input neurons is in our case equivalent to the number of lags of the time-series. We will set the maximum number of lags to 4, the minimum number of lags to 2 and in the process of finding the best network topology we will go through all integer values between the minimum and the maximum number of lags. The same process will be applied for the number of hidden neurons, again we determine a minimum and maximum number of neurons (2-4) and in the construction of the network we go through all possible specifications. The only exception is the specification 4-5-1 (the first number denotes the input neurons, the second the hidden neurons and the third the output neurons), which we added to have a total number of 10 specification for the optimization stage of the ANNs. By iterating through the above mentioned parameters we will get the best possible network specification given our defined restrictions. This is a different approach than Teng and Wah (1996) implement, they start with a complex network and then remove existing connections. The reason for creating different specifications is because we have no prior knowledge of the best specification for the given task. As Yao (1999) proves too simple networks may have a too limited processing power for the given task.

The training algorithm for the ANN is a genetic algorithm. The algorithm creates an initial population, where the number of individuals in the population is an input parameter for the optimization procedure. Each individual is created by randomly determining all the weights between neurons. Therefore after the creation of the initial population we have a given set of values for each weight, which are afterward combined during the offspring breeding phase. The initial population gives us the genetic diversity, which ensures that we are searching over the whole solution space and we do not limit the search to some local optimum, in the case of a too restricted search premature convergence may occur as identified by Eiben and Smith (2010). If we create genetically similar individuals we are in fact searching a very narrow region of the solution space, the advantages of using a big initial population vanish, while the computational burden remains. The number of individuals can not be determined for a given problem in advance; therefore we set a fixed number of individuals for the optimization and compare the performance for different specifications. The next step after creating the initial population is the iterative process of creating new individuals and evaluating their fitness. Once again the number of iterations is an input parameter of the optimization, which is fixed for all specifications.

In general the usual approach is to define a stopping criterion in order to limit the number of iterations based on performance rather than computational complexity, therefore the process stops once it reaches a point where the improvement in performance is below a certain level, which is logical because the subsequent iterations no longer give such an improvement to justify the computational burden. Since we have specifications with different complexities the optimal number of iterations might differ. However we limit the number of iterations rather than using a stopping criterion. There are two reasons for doing so. The first reason is that we want to compare results, which were achieved with the same computational burden; this gives a slight advantage to simpler specifications of the ANNs as they should in theory converge to a solution faster, than their more complex counterparts. The second reason is to avoid over-training or spurious dependencies caused by noise in the data. A more complex network might perform very well on the training data, but it will be outperformed by simpler specifications, because they are better at adapting to patterns or changes in data, which were not present in the training sample. Concerning noisy data complex networks can extract patterns, which are only noise and they are useless for predicting future price durations. On the other hand adding noise to the data is an effective way of reducing over-fitting, as was shown by Sietsema and Dow (1991).

New offspring is created by combining the genetic information of the parents (two random existing individuals), each individual in the population has the same probability to be chosen to be a parent in a given iteration. If a dominant individual is present (in terms of fitness) it is possible that his genetic information after a number of iterations will prevail and so we end up with more identical individuals. We do not limit the creation of identical individuals as it is a form of elitism, in other words the genetic information is dominant. By increasing the number of dominant individuals we increase the probability, that in the next offspring the information will be present as well (it is a desirable property, if it does not lead to premature convergence). In each iteration two new individuals are created, each weight between neurons is considered a gene and the offspring can get the gene from one or the other parent randomly with the same probability. Furthermore mutation is present in the creation of the offspring, therefore with a certain probability neither the gene from one or the other parent is selected, but a random number is assigned

to the given weight between neurons. The mutation ensures that we are able to add additional diversity to our population, however if the probability of a mutation is too high, then we are in fact doing again a random search over the whole solution space. The probability of mutation is set in line with the range suggested by Srinivas and Patnaik (1994) to 4%.

The newly created individuals then pass a fitness test. The fitness test consists of calculating the squared error of the estimated value to the real observation. We sum-up the squared error for all observations in the training set and compare the new individuals with the existing ones based on the summed squared error. In each iteration the two new individuals and the two original individuals are compared and the two worst performing individuals based on the fitness test are discarded. This procedure ensures that we always have the same number of individuals after each iteration and that the quality of the population after each iteration is either identical or improved. Once the given number of iterations has been reached we compare the fitness of all individuals and choose the best individual. The given weights are then used to calculate the estimates of the validation set. Since the performance of the genetic algorithm depends on the initial population, we have to perform the same process repeatedly and work with average values in order to compare the performance with our benchmark ACD model.

The optimization process will be divided into two parts for all data samples we start with the following specifications of ANNS:

Table 4.1: ANN specifications

number of inputs	number of hidden neurons	number of outputs
2	2	1
2	3	1
2	4	1
3	2	1
3	3	1
3	4	1
4	2	1
4	3	1
4	4	1
4	5	1

The data consist of 12000 observations and are divided into a 10000 observations training set (in-sample) and a 2000 observations validation set (out-of-sample). For the initial specification we run the optimization algorithm with the following parameters. The initial population is 300 the number of iterations is 500 and the number of trial runs for each specification is 10. From each run we take the sum of squared errors and calculate the average. Based on this average we rank the ANN specification by performance. The top 3 networks are then selected and run with the following parameters. The initial population is 500 the number of iterations is 1000 and the number of trial runs for each specification is 5. The networks specification with the lowest average of the sum of squared errors is then selected and compared with the ACD model. The reason for dividing the process in two steps is to ensure that we choose not only the best, but also a consistent network specification. In ANNs which use genetic algorithms the results are largely influenced by the initial population quality. So in the first step we choose networks that repeatedly outperform the other specifications and in the second step we increase the initial population and the number of iterations and repeat the process, therefore the network performance should improve. However to avoid overtraining the ANNs the performance in the second is measured on the validation data. The predictions for validation data are estimated similarly as for the ACD model on 1-day ahead forecasts.

The programming language used for the ANN estimation is PLSQL. The motivation behind using a relational database is the applicability to real-life data analysis situations, where most of the data are stored in databases based on SQL. Also in contrast to other SQL based languages PLSQL is an object-oriented language, thus allowing us to design objects and data types dependent on the specific requirements of ANNs.

4.1.1 Adjustments for Raw Data

During the estimation process we encountered a problem with training the ANNs on the raw data sets. The training often terminated with either minimal variance around the mean of the original data, thus not having any predictive ability or the estimates had an upper or lower bound and only captured either decreasing or increasing trends. Both situations led to very poor estimates. The situation was caused by a combination of the activation function we use

(the sigmoid function) and the range of randomized values for the weights.

The raw data sets have values, which exceed 2000. In order to transform them to the interval of (0;1), (after passing through the activation function) the weights between the input neurons and hidden neurons have to be rather low (in most estimations the numbers were below 0.05 in absolute terms), else the value passed on to the next level is in the majority of cases either 1 or 0. However the weights between the hidden layer and the output layer are in the range of -10 to 10, therefore randomizing values from -10 to 10 is not optimal for both layers. The search space becomes too large and convergence takes too long. Furthermore improvements of the genetic algorithm are limited since all weights which result in the output of either 1 or 0 are identical, so even though the values change in the correct direction if the values of the sigmoid activation function do not change the genetic algorithm is not able to determine the improvement.

The solution for the problem was to isolate the generation of random values for the first and the second layer. For the first layer values between -0.1 and 0.1 were generated, while for the second layer the generated values were between -10 and 10. The side effect of this step might be a too restrictive search space and thereby limiting the performance of the ANNs.

4.2 ACD Methodology

The ACD model used for estimation was introduced in section 2.1.2. The model is estimated using maximum likelihood estimation with either the exponential or the Weibull distribution of innovations. The order of the model is ACD(1,1). The model is estimated in Matlab using the native *fminsearch* function to determine the parameter values of the maximum likelihood estimation. We estimate the ACD model for raw and diurnally adjusted data. In both cases we estimate the model in order to compare it to the ANN estimates. In the case of adjusted data we use the model, because it is a standard benchmark in duration literature. On the other hand in the case of raw data we use the model for a lack of possible models, which we could use for comparison. To our knowledge no model exists for estimating raw durations. Therefore we use the ACD to at least have some comparison of the capabilities of ANNs, because it is fairly simple to estimate.

Chapter 5

Empirical results

5.1 ACD Results

The ACD results for raw data resulted in estimates that are not covariance stationary. For adjusted data this would be a serious issue, however since we are applying a model for adjusted data to raw data, for a lack of suitable models to compare to ANNs, we still use the ACD raw results to have at least some performance benchmark. Nonetheless we restrict the comparison only to MSE values and the Diebold-Mariano test statistics. The results of the raw estimates are reported in the table A.1. The rest of the section deals only with ACD results of adjusted data. The ACD results show high persistence with the sum of alpha and beta being close to unity, while the necessary condition to be covariance stationary (not exceeding unity) is fulfilled for adjusted data. The beta parameter is relatively high with values always exceeding 0.76 and the alpha parameter is relatively low. All estimated parameters are significant. The performance of the models is almost identical with exponentially distributed innovations as with Weibull distributed innovations. The results are reported in table 5.1.

Table 5.1: ACD estimated parameters adjusted

	CHF		EUR		JPY	
	exp	wbl	exp	wbl	exp	wbl
ω	0.016	0.016	0.034	0.034	0.040	0.040
α	0.143	0.142	0.151	0.150	0.193	0.194
β	0.841	0.842	0.815	0.817	0.768	0.767
κ	(-)	1.016	(-)	1.046	(-)	0.978
log likelihood	-10803	-10800	-11055	-11033	-10653	-10648

The out-of-sample performance is better than the in-sample performance for both EUR and JPY. The most striking difference is for JPY, where the improvement measured in MSE (an improvement is understood as a decrease in the error) is almost 21%. The CHF shows similar performance in and out-of sample. The difference in performance in and out-of-sample is probably caused by changes in the variance of the data. MSE results are reported in table 5.2.

.

Table 5.2: MSE ACD for raw data

Data/Model	in-sample ACD		out-of-sample ACD	
	exp	wbl	exp	wbl
CHF	1.4390	1.4388	1.4825	1.4824
EUR	1.3844	1.3837	1.0955	1.0950
JPY	1.7170	1.7177	1.0720	1.0723

Exp denotes the exponential distribution, while wbl denotes the Weibull distribution.

The ACD one day ahead forecasts and the corresponding ACF of raw and adjusted data are in figure A.1 and figure A.2, respectively.

5.2 ANN Results

The results are divided into two parts one for raw data and one for adjusted data. A general tendency of the estimation procedure is the convex shape of the error function in relation to the number of iterations. The improvement of the error function is reduced with an increasing number of iterations. The improvements becomes less frequent and the magnitude of the improvement decreases. This indicates that the estimation has reached a point, where increasing the number of iterations has little or no effect on performance.

Another common feature of the results is the dependence on the initial population. Therefore performance varies for individual runs of the model estimation. The difference for individual runs after the initial population is immense, while after the breeding part of the algorithm the range was significantly reduced. The behaviour is a known property of genetic algorithms and motivates the use of more runs and averages rather than a single estimation.

5.2.1 ANN Raw Data

In the first step we estimate 10 different specifications out of which we choose 3 based on in-sample performance. The results are shown in table 5.3. For raw data clearly specifications with lower complexity performed better, with the majority of specifications having only two input neurons. The two exceptions were the 3-2-1 specification for EUR and 3-3-1 specification for JPY. The reason for preference of less complex specifications may be a lower number of iterations to converge to stable values, therefore under a setting with a larger number of iterations more complex networks might outperform less complex networks. Comparing the variability of networks the values for the specification 2-2-1 EUR the range of MSE values was from 13497 to 14060, while for the 4-5-1 specification the values ranged from 13910 to 16012. This trend was present in other currencies as well.

Table 5.3: ANN specification raw in-sample results

number of inputs-hidden-outputs	CHF	EUR	JPY
2-2-1	18111(*)	13825(*)	12384(*)
2-3-1	17722(*)	13680(*)	12760
2-4-1	18088(*)	14003	12635(*)
3-2-1	18914	13652(*)	14648
3-3-1	19529	14072	12465(*)
3-4-1	19653	13912	12868
4-2-1	18874	14859	13220
4-3-1	19137	14554	13674
4-4-1	19755	14296	13783
4-5-1	20848	14786	13895

In the first part of the procedure specifications denoted (*) have the smallest MSE and are chosen for the second part of the estimation procedure

In the second step of the estimation models with a lower number of neurons once again outperform their more complex counterparts, with the only exception being EUR. The in-sample performance leads to similar out-of-sample performance in comparison, the similar order of in-sample and out-of-sample forecasts suggests that none of the specifications is over-trained. The MSE of EUR and JPY decreased in the validation data, while for CHF the value increased. The results are reported in table 5.7.

We report the descriptive statistics for the predicted values of the final specification and compare them to the original values, the results are shown in

Table 5.4: ANN second step raw results

	CHF			EUR			JPY		
Order	Spec.	in-sample	out-of-sample	Spec.	in-sample	out-of-sample	Spec.	in-sample	out-of-sample
1	2-2-1	17339	23625	2-3-1	13286	12383	2-2-1	12132	8974
2	2-3-1	17394	23962	3-2-1	13440	12679	2-3-1	11981	9031
3	2-4-1	18265	24679	2-2-1	13917	12797	2-4-1	12331	9059

The order for every specification in the second step is determined by the out-of-sample MSE

table 5.5. In all data sets the predicted mean and median exceed the values of the original dataset. The original data are positively skewed with values above 4, while the estimated have values below 2. The long right tail with extremely high values in combination with a MSE function causes the ANNs to over-compensate for high values and results in the higher mean and median.

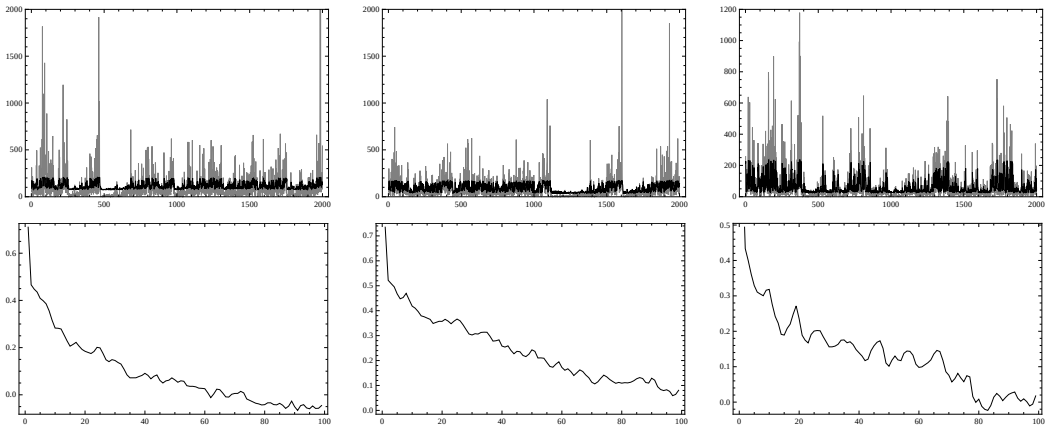
Table 5.5: ANN raw results data comparison

	CHF		EUR		JPY	
	ori	pre	ori	pre	ori	pre
Mean	105.33	116.10	82.044	91.712	66.639	62.696
Median	55.000	105.29	46.000	86.548	33.000	42.256
Minimum	1.0000	75.254	1.0000	35.987	1.0000	24.370
Maximum	2703.0	205.39	2271.0	171.43	1179.0	236.16
Std. Dev.	161.55	35.231	118.59	41.018	99.877	47.520
C.V.	1.5337	0.3035	1.4454	0.44724	1.4988	0.7579
Skewness	6.0134	0.9323	6.5361	0.35654	4.0499	1.8387
Ex. kurtosis	63.435	-0.14270	87.935	-1.1111	24.755	2.7615

Descriptive statistics of original and predicted values.

The graph of the original data and the predicted data together with ACF are reported in figure 5.1. The graph shows that ANNs have limitations regarding the range of possible values they can attain. In the time-series graphs we can see lasting intervals of low values where the ANN prediction has attained the lower bound of possible values and it can not adjust. The upper bound is not so evident, because lasting intervals of consecutive high values are not present in the data. Even though ANNs can model the clustering of price durations, which is a requirement for efficient modelling of duration data, in the current setting the range is restricted.

Figure 5.1: ANN raw predicted values and ACF



The top row shows raw data one day ahead forecasts of ANN in comparison with the original time-series and the bottom row shows ACF of the forecasts.

5.2.2 ANN Adjusted Data

In the first step we estimate 10 specifications and choose 3 for the second part of the estimation. The results are reported in table 5.6. ANNs with lower number of neurons perform better, similarly as for the raw data. However the number of more complex networks, selected for the second step of the procedure, is higher than for raw data. The preference in more complex networks might suggest that after the data adjustment more complicated patterns are present in the data, which the networks are not able to perceive in raw data. The networks, which have more weights to optimize, might profit from the increase in the size of the initial population and the number of iterations, therefore the performance improvement might be more substantial than for less complex networks that are already close to their optimal performance.

The results of the second part of the estimation give no indication to the preferred specification in terms of complexity. For example the best estimates for each data set have a different number of inputs. The effect increasing the number of iterations has on the performance is not clear, while some specifications clearly improve other specification even get worse. The reason that some specifications get worse can be caused by the fact that we only repeat the estimation 5 times in the second step, therefore the average is more likely to be affected by extreme values than in the first step. More complex specifications do not improve more than their less complex counterparts.

An interesting result is the fact that relative in-sample performance does

Table 5.6: ANN specification adjusted in-sample results

number of inputs-hidden-outputs	CHF	EUR	JPY
2-2-1	1.460(*)	1.407	1.767(*)
2-3-1	1.462	1.397(*)	1.742(*)
2-4-1	1.466	1.406(*)	1.795
3-2-1	1.458(*)	1.409	1.840
3-3-1	1.466	1.411	1.764(*)
3-4-1	1.467	1.421	1.853
4-2-1	1.462	1.420	1.950
4-3-1	1.456(*)	1.406(*)	1.820
4-4-1	1.466	1.455	1.830
4-5-1	1.471	1.434	1.912

In the first part of the procedure specifications denoted (*) had the smallest MSE and were chosen for the second part of the estimation procedure

not lead to relative out-of-sample performance. The term relative performance refers to the order specifications have based on MSE, while for raw data good in-sample performance indicated good out-of-sample performance, the same is not true for adjusted data. This may suggest that some specifications are already over-trained and adjust poorly to new patterns in the out-of-sample data. The results of the second step of the procedure are reported in table 5.7.

Table 5.7: ANN second step adjusted results

	CHF			EUR			JPY		
Order	Spec.	in-sample	out-of-sample	Spec.	in-sample	out-of-sample	Spec.	in-sample	out-of-sample
1	3-2-1	1.461	1.506	4-3-1	1.408	1.112	2-2-1	1.747	1.088
2	2-2-1	1.455	1.520	2-3-1	1.393	1.120	2-3-1	1.753	1.093
3	4-3-1	1.468	1.526	2-4-1	1.389	1.130	3-3-1	1.838	1.192

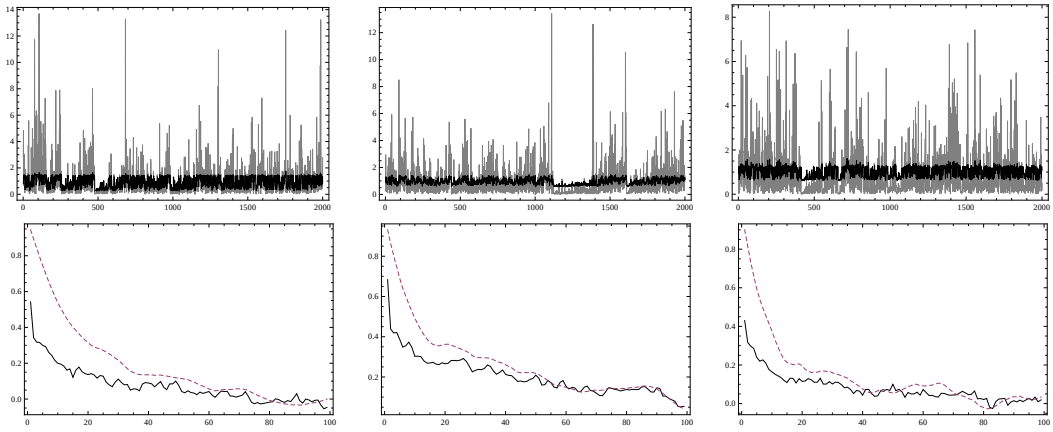
The order for every specification in the second step is determined by the out-of-sample MSE

The predictions of specifications with the lowest out-of-sample MSE are reported in the figure 5.2. The figure includes the ACF of the predictions. Similarly as for the raw data ANN predictions, have a restricted range of values.

5.3 ANN and ACD Comparison

We compare the in and out-of-sample performance of the ANNs to the ACD model for raw and adjusted data based on MSE values and the Diebold-Mariano test statistic for the equality of forecast accuracy (a Newey-West type estima-

Figure 5.2: ANN adjusted predicted values and ACF



The top row shows adjusted data one day ahead forecasts of ANN in comparison with the original time-series and the bottom row shows ACF of the forecasts, where the dashed line is the ACF of the ACD model for comparison.

tor for sample variance of the loss differential is used). The MSE and Diebold-Mariano test statistics are reported in table 5.8 and 5.9, respectively. ANNs outperform the ACD model in the case of raw data by an average of 1.9%, which includes in-sample as well as out-of-sample performance. Only for the out-of-sample performance for CHF is the ACD model better by 1.1%. On the other hand the difference for out-of-sample performance of EUR is 6.9% in favour of the ANN model. However based on the Diebold-Mariano test statistic for comparing predictive accuracy we can not reject the null hypothesis, that the forecast accuracy is equal on a 95% significance level.

Table 5.8: MSE comparison ACD and ANN

Model	in-sample ACD		in-sample ANN		out-of-sample ACD		out-of-sample ANN	
	raw	adj	raw	adj	raw	adj	raw	adj
CHF	17605	1.439	17339	1.461	23363	1.482	23625	1.506
EUR	13466	1.384	13286	1.408	13313	1.095	12383	1.112
JPY	12323	1.717	12132	1.747	9099	1.072	8974	1.088

The distribution (exponential or Weibull) with lower MSE values of the ACD model is reported.

The opposite is true for the average difference for adjusted data, because the ANN estimates have a higher MSE by about 1.6%. For no currency has the ANN model managed to outperform the basic ACD model. During the estimation of ANNs we managed to obtain values below the ones reported for the ACD model, with best estimates having MSEs 1.433, 1.364 , 1.716 for

CHF, EUR, and JPY, respectively. This suggests, that there are combinations of weights for various specifications that are able outperform the ACD model, however we are not able to attain those values repeatedly. When we compare the Diebold-Mariano test statistic we once again can not reject the null hypothesis, that the forecast accuracy is equal on a 95% significance level.

Table 5.9: Diebold-Mariano test statistics

	raw	adj
CHF	0.1288	0.6783
EUR	-0.2583	-0.3156
JPY	-0.6201	0.6966

The comparison of ACF for ACD and ANN reported in figure 5.2 reveals that the ANN model has lower ACF values for all currencies up to about 60 lags. Although the ACD model is a short memory process the graphs show that feed-forward ANNs have an even shorter memory.

5.4 Discussion

The results show that a feed-forward ANN network with a sigmoid activation function is able to model durations. In the case of raw data the ANN models outperform the ACD model based on MSE, however not significantly based on Diebold-Mariano test statistics. The comparison with the ACD model in the case of raw data is a good indication, nonetheless we can not rely on it heavily, because the model is designed for adjusted data. Furthermore the estimates are not covariance stationary. The issue with comparing the performance of the ANN model as stated before, is that no model estimates the data in one step such as ANNs, so we lack a direct comparison. On the other hand the limitations of the model are similar in the case of raw and adjusted data, therefore we can assume that changes to the model that improve the forecasts for raw data will also improve the forecasts for adjusted data and vice versa.

In the case of adjusted data the ACD model outperforms the ANN model, but again the differences are not significant as the Diebold-Mariano statistic shows. However the results from individual estimations suggest that the ANNs are able to achieve better estimates, so there definitely is potential for outper-

forming the ACD model. The results in general indicate that ANNs have the capabilities to model durations, nonetheless the empirical results also show a severe imperfection, that needs to be resolved to improve the performance.

The limitation of the range of possible values the model can forecast is the only alarming issue. Even with this limitation the ANNs are able to compete with the ACD model, but in situations where the structure of data changes significantly out-of-sample or extended periods of long or short durations exist the ACD models is more flexible and will outperform the ANN. There are two possible explanations for the existing upper and lower bound of forecasts.

The first reason can be that we restricted the range of possible values for the weights, therefore the ANN is not able to reach a global optimum and it converges to a local optimal combination of weights. However this is unlikely because many authors restrict the randomly generated weights to values ranging from -1 to 1, while we restricted the values from -10 to 10. In our case this might lead to a need for more iterations, because the space in which we are searching for a value is larger. Nonetheless the results show that increasing the number of iterations in the second part of the estimation process causes no significant improvement, which indicates that only minor adjustments occur in later parts of the estimation and that the parameters are within the right region of values.

Another cause might be the use of the sigmoid activation function. We use the sigmoid activation function because it has two required properties. Firstly it is a non-linear activation function, therefore it is sufficient to have one hidden layer to model a function with finitely many discontinuities. Secondly the data are transformed into the range from 0 to 1, since durations are non-negative we can easily transform the output into the required range. The problem is the output transformation in combination with the sigmoid activation function, because for our adjusted data we require values usually in the range of 0 to 20, but for the output of the activation function after the transformation to be for example 0.05 we need an input of -6. These extremely low or extremely high outputs require input values that the network is not able to achieve and thus fails to model these situations. A linear activation function for the output neurons may solve the problem, but we may have to restrict cases when the output is below zero.

An interesting finding is the fact that the algorithm manages to converge to weight values, which outperform the ACD model, but it fails to repeatedly achieve such a performance. For a genetic algorithm this is surprising, because evolutionary algorithms have the ability to escape local optima.

The reason may be the known inability of ANNs with genetic algorithms to attain values, which are not part of the genetic information. For example when the optimal value for a weight is 4, however in our population for the given weight the value is nonexistent, therefore the value can only became a part of the genetic information by mutation. Given that the mutation parameter is usually below 5% the whole process might take a considerable number of iterations before the optimal value is attained. A possible solution to this problem is the use of a nomadic genetic algorithm or a hill-climbing procedure for the final population. The difference between the two methods is that the nomadic algorithm changes the values of each newly created individual toward better values in the region of the current values, so basically a hill-climbing method is employed during the algorithm and not only at the end. Compared to the current algorithm these steps would improve performance and might lead to outperform the ACD repeatedly.

Finally based on the ACF results comparison of the ACD model and the ANN models, together with the knowledge of long-memory characteristics of price duration data a logical alteration to the current model is the transformation of the feed-forward network into a recurrent network. The results show that ANNs have an even shorter memory than ACD models, which belong to the category of short-memory models. To achieve the property of longer memory the network needs the ability to function in a recurrent matter. During the training process the networks learns to decide, which information to keep and how long it should keep it. Theoretically a value can be stored in the network for an arbitrarily long time.

Chapter 6

Conclusion

The aim of this thesis was to apply ANNs in the context of price durations, because to our knowledge no attempt to model durations using ANNs has been done so far. The motivation for understanding price durations or financial durations in general, is a better knowledge of the market micro-structure and the price creation process. The advancements and the theoretical framework of financial durations and ANNs were described in the literature review.

We then designed a feed-forward neural network with a sigmoid activation function and a genetic algorithm. Because a genetic algorithm was used, we repeatedly estimated the models and worked with averages. The reason for working with averages in case of evolutionary algorithms is that the performance depends on the initial population, which is generated at random. Therefore the performance varies across estimations. The neural network was then trained and tested on data of foreign exchange futures contracts traded on the Chicago Mercantile Exchange. The basic ACD model was used as a benchmark model to evaluate the capabilities of ANNs in terms of duration forecasting.

The estimation was done for raw data as well as diurnally adjusted data. We used the ACD model for raw data estimation, because no competing model that uses a one step estimation process for forecasting durations exists. Competing duration models use a two step estimation procedure, where in the first step the seasonalities are removed and in the second step the durations are estimated. Therefore to have at least some comparison of the capabilities of ANNs to model raw data, we used the ACD model, even though the estimated model was not covariance stationary.

ANNs were able to outperform the ACD for raw data based on the MSE, however the forecast accuracy was similar for both models. This indicates that ANNs can be used for duration forecasting without a need for data preprocessing, however these results are to be treated with caution, beacuse the ACD model was not designed for raw data. The possibility to apply the model to raw data eliminates the risk of distorting the data during the diurnal adjustment and thus consequently distorting the results. Furthermore ANNs can easily be extended to work with marks of durations. Therefore the models can either be extended to use the additional information as inputs for better forecasts or even to jointly estimate durations and marks.

ANNs were not able to outperform the traditional ACD model, they performed worse on average by 1.6% (measured by MSE), but again the forecasting accuracy based on the Diebold-Mariano test statistic was similar. Furthermore for some estimations ANNs outperformed the ACD model, however the average performance was worse. Therefore ANNs have in general the ability to model durations, yet they need adjustments before being able to significantly outperform the ACD model.

The ANN results for raw and adjusted data showed similar limitations in both cases. ANNs were no able to adjust sufficiently to prolonged periods of long or short durations. A lower and upper bound for the for the forecasts existed and the ANNs were not flexible enough to new patterns in the data. The application of a linear activation function for the output neuron, rather than the sigmoid function, might be able to remove this limitation and enable to improve the forecast accuracy.

In comparison with the ACD model ANNs had a shorter memory, this would suggest that an extension of the memory of ANNs could lead to a performance improvement. In conclusion the results show that ANNs can be applied to forecasting raw and adjusted data durations they also provide insights on the necessary and possible adjustments to improve forecast accuracy.

Future research should focus on the incorporation of longer memory into the ANNs, because the results in current financial duration literature indicate that longer memory processes perform better in comparison to their short memory counterparts.

Bibliography

Admati, A. & P. Pfleiderer (1988): "A theory of intraday patterns: volume and price variability." *The Review of Financial Studies* **1(1)**: pp. 3–40.

Baillie, R., T. Bollerslev, & H. Mikkelsen (1996): "Fractionally integrated generalized autoregressive conditional heteroskedasticity." *Journal of Econometrics* **74(1)**: pp. 3–30.

Bauwens, L. & P. Giot (2000): "The Logarithmic ACD Model: An Application to the Bid-Ask Quote Process of Three NYSE Stocks." *Annals of Economics and Statistics* **60**: pp. 117–149.

Bauwens, L. & P. Giot (2001): *Econometric Modeling of Stock Market Intraday Activity*. Dordrecht: Kluwer Academic Publishers.

Bauwens, L. & D. Veredas (2004): "The stochastic conditional duration model: a latent variable model for the analysis of financial durations." *Journal of Econometrics* **119(2)**: pp. 381–412.

Bishop, C. M. (1995): *Neural Networks for Pattern Recognition*. Oxford: Clarendon Press.

Borisov, A. N. & V. A. Pavlov (1995): "Prediction of a continuous function with the aid of neural networks." *Automatic Control and Computer Sciences* **29(5)**: pp. 39–50.

Chen, X., J. Racine, & S. N.R. (2001): "Semiparametric ARX neural-network models with an application to forecasting." *IEEE Transactions on Neural Networks* **12**: pp. 674–683.

Cybenko, G. (1989): "Approximation by superpositions of a sigmoidal function." *Mathematics of Control, Signals, and Systems* **2(4)**: pp. 303–314.

Deo, R., M. Hsieh, & C. Hurvich (2005): "Tracing the Source of Long Memory in Volatility ." *Statistics Working Papers Series* .

Deo, R., M. Hsieh, & C. Hurvich (2010): "Long memory in intertrade durations, counts and realized volatility of NYSE stocks." *Journal of Statistical Planning and Inference* **140(12)**: pp. 3715–3733.

Deo, R., C. Hurvich, & Y. Lu (2006): "Forecasting realized volatility using a long-memory stochastic volatility model: estimation, prediction and seasonal adjustment." *Journal of Econometrics* **131**: pp. 29–58.

Deo, R., C. Hurvich, & Y. Lu (2009): "Conditions for the propagation of memory parameter from durations to counts and realized volatility." *Econometric Theory* **25(3)**: pp. 764–792.

Diebold, F. & R. Mariano (1995): "Comparing Predictive Accuracy." *Journal of Business & Economic Statistics* **13(3)**: pp. 253–263.

Easley, D. & M. O'Hara (1992): "Time and the Process of Security Price Adjustment." *The Journal of Finance* **47(21)**: pp. 577–605.

Eiben, A. & M. Schoenauer (2002): "Evolutionary computing." *Information Processing Letters* **82(1)**: pp. 1–6.

Eiben, A. & J. Smith (2010): *Introduction to evolutionary computing*. Berlin: Springer.

Engle, R. F. (2000): "The econometrics of ultra-high-frequency data." *Econometrica* **68(1)**: pp. 1–22.

Engle, R. F. & J. R. Russell (1998): "Autoregressive Conditional Duration: A New Model for Irregularly Spaced Transaction Data." *Econometrica* **66(5)**: pp. 1127–1162.

Fernandes, M. & J. Grammig (2005): "Nonparametric specification tests for conditional duration models." *Journal of Econometrics* **127(1)**: pp. 35–68.

Fernandes, M. & J. Grammig (2006): "A family of autoregressive conditional duration models." *Journal of Econometrics* **130(1)**: pp. 1–23.

Ferreira, T. A. E., G. C. Vasconcelos, & P. J. L. Adeodato (2008): "A New Intelligent System Methodology for Time Series Forecasting with Artificial Neural Networks." *Neural Processing Letters* **28**: pp. 113–129.

Glorot, X. & Y. Bengio (2010): "Understanding the difficulty of training deep feedforward neural networks." *Proceedings of the 13th International Conference on Artificial Intelligence and Statistic* **9**.

Grammig, J. & M. Wellner (2002): "Modeling the interdependence of volatility and inter-transaction duration processes." *Journal of Econometrics* **106(2)**: pp. 369–400.

G.S., A. & V. V.P. (2009): "Forecasting stock market short-term trends using a neuro-fuzzy based methodology." *Expert Systems with Applications* **36**.

Hadavandi, E., H. Shavandi, & A. Ghanbari (2010): "Integration of genetic fuzzy systems and artificial neural networks for stock price forecasting." *Knowledge-Based Systems* **8**: pp. 800–808.

Hornik, K., M. Stinchcombe, & H. White (1989): "Multilayer feedforward networks are universal approximators." *Neural Newtworks* **2(5)**: pp. 359–366.

Jasiak, J. (1998): "Persistence in Intertrade Durations." *Finance,* **19**: pp. 166–195.

Kaastra, I. & M. Boyd (1995): "Forecasting futures trading volume using neural networks." *The Journal of Futures Markets* **15(8)**: pp. 953–970.

Kaastra, I. & M. Boyd (1996): "Designing a neural network for forecasting financial and economic time series." *Neurocomputing* **10(3)**: pp. 215–236.

Kinjal, J. & P. Mahesh (2012): "Optimizing Weights of Artificial Neural Networks using Genetic Algorithms." *International Journal of Advanced Research in Computer Science and Electronics Engineering* **1(10)**.

Lapedes, A. & R. Farber (1987): "Nonlinear signal processing using neural networks: prediction and system modeling." *Technical report*, Technical Report LA-UR-87-2662, Los Alamos National Laboratory, Los Alamos, NM.

Lapedes, A. & R. Farber (1988): "How neural nets work." *Neural Information Processing Systems, American Institute of Physics, New York* pp. 442–456.

Leung, F., H. Lam, S. Ling, & P. Tam (2003): "Tuning of the structure and parameters of a neural network using an improved genetic algorithm ." *IEEE Transactions on Neural Networks* **14(1)**: pp. 79–88.

Masters, T. (1993): *Practical Neural Network recipes in C + +*. New York: Academic Press.

McNelis, P. (2005): *Neural networks in finance: gaining predictive edge in the market*. Burlington: Elsevier.

Meitz, M. & T. Terasvirta (2006): "Evaluating Models of Autoregressive Conditional Duration." *Journal of Business and Economic Statistics* **24(1)**: pp. 104–124.

Neri, F. & V. Tirronen (2010): "Recent advances in differential evolution: a survey and experimental analysis." *Artificial Intelligence Review* **33(1-2)**: pp. 61–106.

Odom, M. & R. Sharda (1990): "A neural network model for bankruptcy prediction." *Proceedings of the IEEE International Joint Conference on Neural Networks* pp. 163–168.

Pacurar, M. (2008): "Autoregressive conditional duration models in finance: A survey of the theoretical and empirical literature." *Journal of Economic Surveys* **22(4)**: pp. 711–751.

Peralta, J., L. Xiaodong, G. Gutierrez, & A. Sanchis (2010): "Time series forecasting by evolving artificial neural networks using genetic algorithms and differential evolution." *The 2010 International Joint Conference on Neural Networks* pp. 1–8.

Poli, R. (2008): "Analysis of the publications on the applications of particle swarm optimisation." *Journal of Artificial Evolution and Applications* **(3)**.

Safarzynska, K. & J. van den Bergh (2010): "Evolutionary models in economics: a survey of methods and building blocks." *Journal of Evolutionary Economics* **20(3)**: pp. 329–373.

Sathya, S. & M. Radhika (2013): "Convergence of nomadic genetic algorithm on benchmark mathematical functions." *Applied Soft Computing* **13(5)**: pp. 2759–2766.

Sietsma, J. & R. Dow (1991): "Creating artificial neural networks that generalize." *Neural Networks* **4(1)**: pp. 67–79.

Srinivas, M. & L. Patnaik (1994): "Adaptive probabilities of crossover and mutation in genetic algorithms." *IEEE Transactions on Systems, Man and Cybernetics* **24(4)**: pp. 656–667.

Teng, C. & B. Wah (1996): "Automated learning for reducing the configuration of a feedforward neural network." *IEEE Transactions on Neural Networks* **7(5)**: pp. 1072–1085.

Verner, R. (2011): "Stock Markets Analysis Using New Genetic Annealed Neural Network." *Master thesis. Charles University in Prague, Faculty of Social Sciences, Institute of Economical Studies* .

Wang, Y.-H. (2009): "Using neural network to forecast stock index option price: a new hybrid GARCH approach." *Quality & Quantity* **43(5)**: pp. 833–843.

Yao, X. (1999): "Evolving artificial networks." *Proceedings IEEE* **87**: pp. 1423–1447.

Yim, J. (2002): "A comparison of neural networks with time series models for forecasting returns on a stock market index." *RMIT Business Working Paper Series, Working Paper No. 07/2002* .

Zhang, G. & D. Kline (2007): "Quarterly Time-Series Forecasting With Neural Networks." *IEEE Transactions on Neural Networks* **18(6)**: pp. 1800–1814.

Zhang, G., B. E. Patuwo, & M. Y. Hu (1998): "Forecasting with artificial neural networks." *International Journal of Forecasting* **14**: pp. 35–62.

Zhang, M., J. Russell, & R. Tsay (2001): "A nonlinear autoregressive conditional duration model with applications to financial transaction data." *Journal of Econometrics* **104(1)**: pp. 179–207.

Zikes, F., J. Barunik, & N. Shenai (2013): "Modeling and Forecasting Persistent Financial Durations." *Working Paper* .

Appendix A

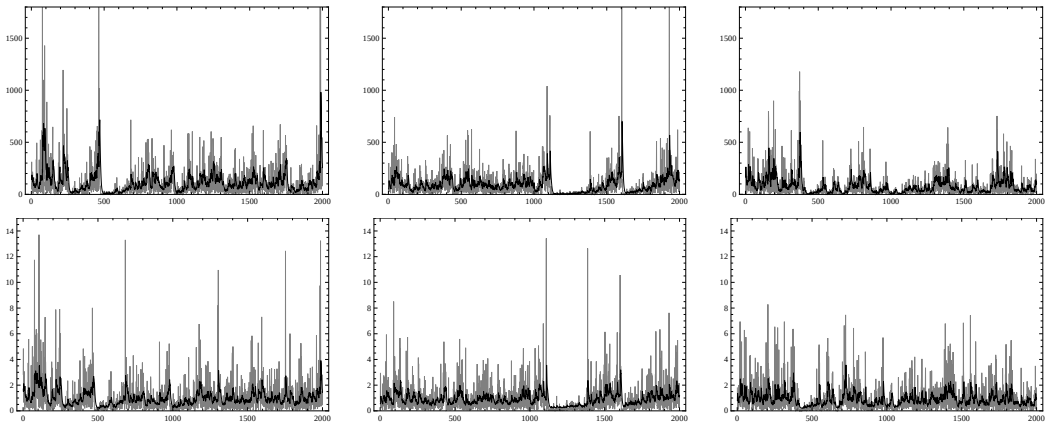
Tables and Figures

In the case of multiple figures in a line the set order is CHF, EUR, JPY. Unless specified otherwise.

Table A.1: ACD estimated parameters raw

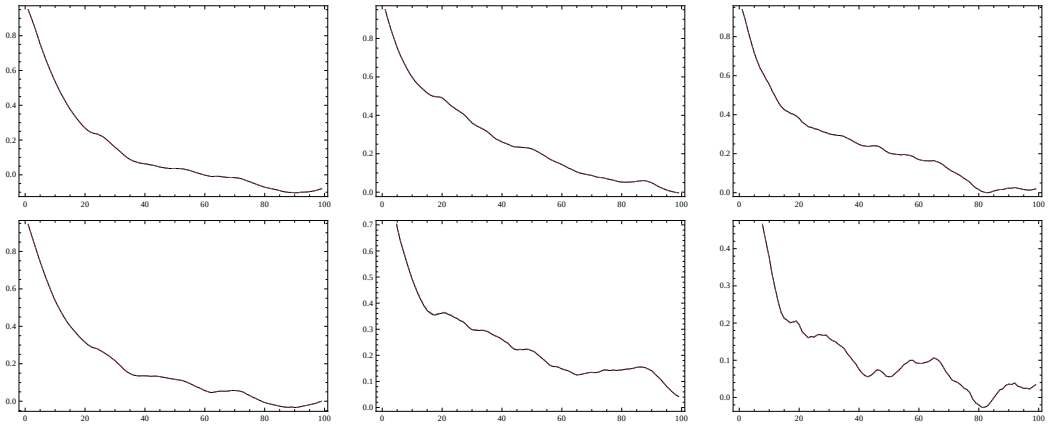
	CHF		EUR		JPY	
	exp	wbl	exp	wbl	exp	wbl
ω	-0.019	-0.019	-0.017	-0.017	-0.028	-0.028
α	0.202	0.202	0.196	0.195	0.240	0.241
β	0.816	0.817	0.822	0.823	0.788	0.786
κ	(-)	1.008	(-)	1.030	(-)	0.963
log likelihood	-6.566e+04	-6.566e+04	-6.420e+04	-6.419e+04	-6.105e+05	-6.104e+04

Figure A.1: ACD forecasts



The top row shows raw data one day ahead forecasts compared to the original time-series and the bottom row shows adjusted data one day ahead forecasts compared to the adjusted time-series. The forecast is depicted in black.

Figure A.2: ACD forecast ACF



The top row shows ACF from raw ACD forecasts and the bottom row shows ACF from adjusted ACD forecasts.

Appendix B

Content of Enclosed DVD

There is a DVD enclosed to this thesis which contains empirical data and PLSQL source codes.

- Folder 1: Source codes
- Folder 2: Empirical data