ADVANCES IN ECONOMETRICS
VOLUME 19

APPLICATIONS OF ARTIFICIAL INTELLIGENCE IN FINANCE AND ECONOMICS

JANE M. BINNER
GRAHAM KENDALL
SHU-HENG CHEN
Editors

APPLICATIONS OF ARTIFICIAL INTELLIGENCE IN FINANCE AND ECONOMICS

ADVANCES IN ECONOMETRICS

Series Editors: T. Fomby and R. Carter Hill

ADVANCES IN ECONOMETRICS VOLUME 19

APPLICATIONS OF ARTIFICIAL INTELLIGENCE IN FINANCE AND ECONOMICS

EDITED BY

JANE M. BINNER

Department of Strategic Management, Aston Business School, UK

GRAHAM KENDALL

School of Computer Science and Information Technology, University of Nottingham, UK

SHU-HENG CHEN

Department of Economics, National Chengchi University, Taiwan

2004

ELSEVIER

JAI

Amsterdam – Boston – Heidelberg – London – New York – Oxford
Paris – San Diego – San Francisco – Singapore – Sydney – Tokyo

ELSEVIER B.V.
Radarweg 29
P.O. Box 211
1000 AE Amsterdam
The Netherlands

ELSEVIER Inc.
525 B Street, Suite 1900
San Diego
CA 92101-4495
USA

ELSEVIER Ltd
The Boulevard, Langford
Lane, Kidlington
Oxford OX5 1GB
UK

ELSEVIER Ltd
84 Theobalds Road
London
WC1X 8RR
UK

First edition 2004

British Library Cataloguing in Publication Data
A catalogue record is available from the British Library.

ISBN: 0-7623-1150-9
ISSN: 0731-9053 (Series)

♾ The paper used in this publication meets the requirements of ANSI/NISO Z39.48-1992 (Permanence of Paper). Printed in The Netherlands.

CONTENTS

LIST OF CONTRIBUTORS *vii*

STATISTICAL ANALYSIS OF GENETIC ALGORITHMS IN DISCOVERING TECHNICAL TRADING STRATEGIES
Chueh-Yung Tsao and Shu-Heng Chen *1*

A GENETIC PROGRAMMING APPROACH TO MODEL INTERNATIONAL SHORT-TERM CAPITAL FLOW
Tina Yu, Shu-Heng Chen and Tzu-Wen Kuo *45*

TOOLS FOR NON-LINEAR TIME SERIES FORECASTING IN ECONOMICS – AN EMPIRICAL COMPARISON OF REGIME SWITCHING VECTOR AUTOREGRESSIVE MODELS AND RECURRENT NEURAL NETWORKS
Jane M. Binner, Thomas Elger, Birger Nilsson and Jonathan A. Tepper *71*

USING NON-PARAMETRIC SEARCH ALGORITHMS TO FORECAST DAILY EXCESS STOCK RETURNS
Nathan Lael Joseph, David S. Brée and Efstathios Kalyvas *93*

CO-EVOLVING NEURAL NETWORKS WITH EVOLUTIONARY STRATEGIES: A NEW APPLICATION TO DIVISIA MONEY
Jane M. Binner, Graham Kendall and Alicia Gazely *127*

FORECASTING THE EMU INFLATION RATE: LINEAR ECONOMETRIC VS. NON-LINEAR COMPUTATIONAL MODELS USING GENETIC NEURAL FUZZY SYSTEMS
Stefan Kooths, Timo Mitze and Eric Ringhut *145*

FINDING OR NOT FINDING RULES IN TIME SERIES
Jessica Lin and Eamonn Keogh 175

A COMPARISON OF VAR AND NEURAL NETWORKS WITH GENETIC ALGORITHM IN FORECASTING PRICE OF OIL
Sam Mirmirani and Hsi Cheng Li 203

SEARCHING FOR DIVISIA/INFLATION RELATIONSHIPS WITH THE AGGREGATE FEEDFORWARD NEURAL NETWORK
Vincent A. Schmidt and Jane M. Binner 225

PREDICTING HOUSING VALUE: GENETIC ALGORITHM ATTRIBUTE SELECTION AND DEPENDENCE MODELLING UTILISING THE GAMMA TEST
Ian D. Wilson, Antonia J. Jones, David H. Jenkins and J. A. Ware 243

LIST OF CONTRIBUTORS

Jane M. Binner	Department of Strategic Management, Aston Business School, Aston Triangle, Aston University, Birmingham, UK
David S. Brée	Computer Science Department, University of Manchester, UK
Shu-Heng Chen	AI-ECON Research Center, Department of Economics, National Chengchi University, Taiwan
Thomas Elger	Department of Economics, Lund University, Sweden
Alicia Gazely	Department of Information Systems and Management, Nottingham Trent University, UK
David. H. Jenkins	School of Technology, University of Glamorgan, UK
Antonia J. Jones	Department of Computer Science, Cardiff University, UK
Nathan Lael Joseph	Aston Business School, Aston University, Aston Triangle, Birmingham, UK
Efstathios Kalyvas	Computer Science Department, University of Manchester, UK
Graham Kendall	School of Computer Science and Information Technology, University of Nottingham, UK
Eamonn Keogh	Computer Science & Engineering Department, University of California – Riverside, USA
Stefan Kooths	Muenster Institute for Computational Economics, Germany

Tzu-Wen Kuo	AI-ECON Research Center, Department of Economics, National Chengchi University, Taiwan
Hsi Cheng Li	Bryant University, Smithfield, USA
Jessica Lin	Computer Science & Engineering Department, University of California – Riverside, USA
Sam Mirmirani	Bryant University, Smithfield, USA
Timo Mitze	Gesellschaft fuer Finanz- und Regionalanalysen (GEFRA), Germany
Birger Nilsson	Department of Economics, Lund University, Sweden
Eric Ringhut	Muenster Institute for Computational Economics, Germany
Vincent A. Schmidt	Wright State University, Dayton, OH, USA
Jonathan A. Tepper	School of Computing and Technology, Faculty of Construction, The Nottingham Trent University, Nottingham, UK
Chueh-Yung Tsao	Department of Business Administration, Chang-Gung University, Taiwan
Andrew Ware	School of Computing, University of Glamorgan, UK
Ian D. Wilson	School of Computing, University of Glamorgan, UK
Tina Yu	ChevronTexaco Information Technology Company, San Ramon, CA, USA

INTRODUCTION

Artificial intelligence is a consortium of data-driven methodologies which includes artificial neural networks, genetic algorithms, fuzzy logic, probabilistic belief networks and machine learning as its components. We have witnessed a phenomenal impact of this data-driven consortium of methodologies in many areas of studies, the economic and financial fields being of no exception. In particular, this volume of collected works will give examples of its impact on the field of economics and finance. This volume is the result of the selection of high-quality papers presented at a special session entitled "Applications of Artificial Intelligence in Economics and Finance" at the "2003 International Conference on Artificial Intelligence" (IC-AI '03) held at the Monte Carlo Resort, Las Vegas, NV, USA, June 23–26 2003. The special session, organised by Jane Binner, Graham Kendall and Shu-Heng Chen, was presented in order to draw attention to the tremendous diversity and richness of the applications of artificial intelligence to problems in Economics and Finance. This volume should appeal to economists interested in adopting an interdisciplinary approach to the study of economic problems, computer scientists who are looking for potential applications of artificial intelligence and practitioners who are looking for new perspectives on how to build models for everyday operations.

The structure of this volume is as follows; in the first chapter by Shu-Heng Chen and Chueh-Yung Tsao a statistical approach to testing the performance of GA-based trading strategies is proposed. The paper asks the question, what are the statistical properties which distinguish a successful application of GA from an unsuccessful one? The performance of ordinal GA-based trading strategies is evaluated under six classes of time series model, namely, the *linear ARMA model*, the *bilinear model*, the *ARCH model*, the *GARCH model*, the *threshold model* and the *chaotic model*. The performance criteria employed are the winning probability, accumulated returns, Sharpe ratio and luck coefficient. Asymptotic test statistics for these criteria are derived. The hypothesis as to the superiority of GA over a benchmark, say, buy-and-hold, is then be tested using Monte Carlo simulation. From this rigorously-established evaluation process, simple genetic algorithms are found to work very well in linear stochastic environments, and they are also found to work very well in nonlinear deterministic (chaotic) environments. However, they may perform much worse in pure nonlinear stochastic cases. These results shed

light on the superior performance of GA when it is applied to the two *tick-by-tick* time series of foreign exchange rates: *EUR/USD* and *USD/JPY*.

The second chapter by Tina Yu, Shu Heng Chen and Tzu-Wen Kuo models international short-term capital flow by identifying technical trading rules in short-term capital markets. Through the simulation, they investigate if there exist any trading strategies that are capable of predicting the capital inflow and out-flow, hence making investment profitable. The modelling and simulation were conducted using Genetic Programming (GP), a novel approach for this task. The simulation results suggest that the international short-term markets were quite efficient during the period of 1997–2002, with most GP generated trading strategies recommending buy-and-hold on one or two assets. The out-of-sample performance of GP trading strategies varies from year to year. However, many of the strategies are able to forecast Taiwan stock market down time and avoid making futile investment. Investigation of Automatically Defined Functions shows that they do not give advantages or disadvantages to the GP results.

The third chapter by Binner, Elger, Nilsson and Tepper contrasts the forecasting performance of two non-linear models, a regime-switching (RS) vector autoregressive model (VAR) and a recurrent neural network (RNN), to that of a linear benchmark VAR model. These models belong to different classes of non-linear models that are both econometrically challenging and therefore rarely compared. Evidence suggests that the RNN model and RS-VAR model outperform the VAR model for both monthly and annual forecast horizons. The RS-VAR and the RNN perform approximately on par over both forecast horizons. For the RS-VAR model, findings suggest that imposing the restriction that only the intercept is allowed to vary across regimes provides the best forecasts. For the RNN-model, the forecasting performance depends on the number of hidden units and thus free parameters included.

Nathan Joseph, David Brée and Efstathios Kalyvas ask the question, "Are the learning procedures of genetic algorithms (GAs) able to generate optimal architectures for artificial neural networks (ANNs) in high frequency data?" in paper four. The approach is in some respects similar in spirit to the use of bootstrapping to select suitable ANN structures. The architectures of the ANNs are also evaluated, both in- and out-of-sample, against a set of naïve RW models and the mean absolute error (MAE). The use of in-sample forecasts facilitates an assessment of the suitability of the chosen ANN configuration prior to implementation, while the use of RW models serves to evaluate the contribution of the ANNs to forecasting accuracy. No ANN architectures were able to outperform a random walk, despite the finding of non-linearity in the excess returns. This failure is attributed to the absence of suitable ANN structures and further implies that researchers need to be cautious when making inferences from ANN results that use high frequency data.

Chapter 5 uses the Artificial Intelligence techniques of evolutionary strategies and neural networks to evaluate the performance of simple sum monetary aggregates vis-à-vis their Divisia index counterparts in a simple inflation forecasting experiment. Jane Binner, Graham Kendall and Alicia Gazely find that that superior tracking of inflation is possible for models that employ a Divisia M2 measure of money that has been adjusted to incorporate a learning mechanism to allow individuals to gradually alter their perceptions of the increased productivity of money. Divisia measures of money outperform their simple sum counterparts as macroeconomic indicators. Co-evolution is found to compete very favourably with neural networks and has the potential to beat neural networks in terms of superior predictive performance when used to evolve neural networks. Artificial Intelligence techniques in general and co-evolution in particular are highly effective tools for predicting future movements in inflation and the paper concludes that there is tremendous scope for further research into the development of these methods as new macroeconomic forecasting models.

Stefan Kooths, Timo Mitze and Eric Ringhut seek to determine whether the predictive power of dynamic single-equation, linear econometric models outperform models based on a novel computational approach using genetic-neural fuzzy rule-bases when forecasting the EMU inflation rate in paper six. Evidence for the superiority of the computational approach based on genetic-neural fuzzy rule-bases (GENEFER) is found according to various different evaluation criteria. Especially striking is the ability of GENEFER models to predict turning points reliably. GENEFER models perform best within a 2-stage approach, where the disequilibrium (error-correction) terms from cointegration analysis are used as input variables. This result proposes a combination of econometric and computational techniques and calls for further research.

Given the recent explosion of interest in streaming data and online algorithms, clustering of time series subsequences has received much attention. In the seventh chapter, Jessica Lin and Eamonn Keogh make a surprising claim. Clustering of time series subsequences is completely meaningless. More concretely, clusters extracted from these time series are forced to obey a certain constraint that is pathologically unlikely to be satisfied by any dataset, and because of this, the clusters extracted by any clustering algorithm are essentially random. While this constraint can be intuitively demonstrated with a simple illustration and is simple to prove, it has never appeared in the literature. Jessica and Eamonn can justify calling their claim surprising, since it invalidates the contribution of dozens of previously published papers. They justify their claim with a theorem, illustrative examples, and a comprehensive set of experiments on reimplementations of previous work.

Sam Mirmirani and Hsi-Cheng Li apply VAR and ANN techniques to make ex-post forecast of U.S. oil price movements in the eighth chapter. The VAR-based forecast uses three endogenous variables: lagged oil price, lagged oil supply and lagged energy consumption. However, the VAR model suggests that the impacts of oil supply and energy consumption has limited impacts on oil price movement. The forecast of the genetic algorithm-based ANN model is made by using oil supply, energy consumption, and money supply (M1). Root mean squared error and mean absolute error have been used as the evaluation criteria. Their analysis suggests that the back-propagation network-GA model noticeably outperforms the VAR model.

Vincent Schmidt and Jane Binner demonstrate how neural network models (such as the Aggregate Feedforward Neural Network) provide beneficial information in the domain of discovering and describing the money-price relationship using Divisia component data in the ninth chapter. The AFFNN is demonstrated as being straightforward to design and use with encoded Divisia component and inflation data, and the model is able to effectively learn the relationships within the dataset. A simple decompositional rule extraction technique examines the learned knowledge and automatically generates a collection of if-then rules in terms of the original attribute values. These Divisia rules are suitable for examination by subject-matter experts (specifically, econometricians). The rules potentially provide interesting and useful insight into monetary aggregation theory, particularly when exploring the relationships between various monetary assets and the corresponding growth rate of prices. As an additional advantage, the resulting rules are expressed in well-documented computer code, capable of being executed for validation or used for forecasting purposes.

Ian Wilson, Antonia Jones, David Jenkins and Andrew Ware show, by means of an example of its application to the problem of house price forecasting, an approach to attribute selection and dependence modelling utilising the Gamma Test (GT), a non-linear analysis algorithm that is described. The GT is employed in a two-stage process: first the GT drives a Genetic Algorithm (GA) to select a useful subset of features from a large dataset that is developed from eight economic statistical series of historical measures that may impact upon house price movement. Next a predictive model is generated utilising an Artificial Neural Network (ANN) trained to the Mean Squared Error (MSE) estimated by the GT, which accurately forecasts changes in the House Price Index (HPI). They present a background to the problem domain and demonstrate, based on results of this methodology, that the GT was of great utility in facilitating a GA based approach to extracting a sound predictive model from a large number of inputs in a data-point sparse real-world application.

There are still many important Artificial Intelligence disciplines yet to be covered. Among them are the methodologies of independent component analysis,

reinforcement learning, inductive logical programming, classifier systems and Bayesian networks, not to mention many ongoing and highly fascinating hybrid systems. A way to make up for their omission is to visit this subject again later. We certainly hope that we can do so in the near future with another volume of "Applications of Artificial Intelligence in Economics and Finance."

STATISTICAL ANALYSIS OF GENETIC ALGORITHMS IN DISCOVERING TECHNICAL TRADING STRATEGIES

Chueh-Yung Tsao and Shu-Heng Chen

ABSTRACT

In this study, the performance of ordinal GA-based trading strategies is evaluated under six classes of time series model, namely, the linear ARMA model*, the* bilinear model*, the* ARCH model*, the* GARCH model*, the* threshold model *and the* chaotic model*. The performance criteria employed are the winning probability, accumulated returns, Sharpe ratio and luck coefficient. Asymptotic test statistics for these criteria are derived. The hypothesis as to the superiority of GA over a benchmark, say, buy-and-hold, can then be tested using Monte Carlo simulation. From this rigorously-established evaluation process, we find that simple genetic algorithms can work very well in linear stochastic environments, and that they also work very well in nonlinear deterministic (chaotic) environments. However, they may perform much worse in pure nonlinear stochastic cases. These results shed light on the superior performance of GA when it is applied to the two* tick-by-tick *time series of foreign exchange rates:* EUR/USD *and* USD/JPY.

1. INTRODUCTION

Genetic algorithms (GAs) have been developed by Holland (1975) to mimic some of the processes observed in natural evolution. They are based on the

Applications of Artificial Intelligence in Finance and Economics
Advances in Econometrics, Volume 19, 1–43

ISSN: 0731-9053/doi:10.1016/S0731-9053(04)19001-4

genetic processes of natural selection which have become widely known as the "survival of the fittest" since Darwin's celebrated work. In recent years, GAs have been successfully applied to find good solutions to real-world problems whose search space is complex, such as the traveling salesman problem, the knapsack problem, large scheduling problems, graph partitioning problems, and engineering problems, too.[1]

In finance, Bauer (1994) provides the first application of GAs to discover trading strategies. Since then, GAs have gradually become a standard tool for enhancing investment decisions.[2] While many studies have supported the effectiveness of GAs in investment decisions; however, the foundation of these applications has not been well established. The thing that concerns us, therefore, is the *robustness* of these empirical results. For example, if GAs are effective for the investment in one market at one time, would the same result apply to the same market or different markets at different times? It is for the purpose of pursuing this generality, that we see the necessity of building a solid foundation upon which a rigorous evaluation can be made.

In this paper, a statistical approach to testing the performance of GA-based trading strategies is proposed. Instead of testing the performance of GAs in specific markets as a number of conventional studies already have, we are interested in a market-independence issue: *what makes GAs successful and what makes them not*? Since the data to which GAs are applied consist of financial time series, the question can be rephrased as follows: what are the *statistical properties* which distinguish a successful application of GA from an unsuccessful one? One way to think of the question is to consider two markets following different stochastic processes. One market follows stochastic process A, and the other stochastic process B. If GAs can work well with stochastic process A, but not B, then the successful experience of GAs in the first market is certainly not anticipated in the second market.

Having said that, this paper follows the following research methodology. First, some financially-related stochastic processes are singled out as the standard scenarios (testbeds) to test the performance of GA. Second, appropriate *performance criteria* are used to evaluate the performance of the GA over these testbeds. Third, the associated *asymptotic statistical tests* are applied to examine whether the GAs perform significantly differently as opposed to a familiar *benchmark*. By this procedure, we may be able to distinguish the processes in which the GA has competence from others in which it does not. Once the critical properties are grasped, we can then apply the GA to the financial time series whose stochastic properties are well-known, and test whether the GA behaves consistently with what we have learned from the previous statistical analysis.

By means of the procedure established in this paper, we hope to push forward the current applications of GAs or, more generally, computational intelligence (CI),

toward a more mature status. After all, whether GA will work has been asked too intensely in the literature. The very mixed results seem to suggest that we look at the same question at a finer level and start to inquire why it works or why it doesn't. We believe that there are other ways to do something similar to what we propose in this paper.[3] We do not exclude these possibilities. In fact, little by little, these efforts will eventually enable GA or CI tools to rid themselves of their notoriety for being *blackboxes*.

The rest of the paper is organized as follows. Section 2 introduces a specific version of GA, referred as to the ordinary GA (OGA), used in this paper. Section 3 will detail the classes of stochastic processes considered in this paper and the reasons for this choice. Section 4 reviews the four performance criteria and establishes their associated asymptotic test. Section 5 sets up the Monte Carlo simulation procedure. Section 6 summarizes and discusses the actual performance of the GA over the artificial data, whereas the counterpart over the real data is given in Section 7. Section 8 concludes this paper.

2. TRADING WITH GAS

A *trading strategy* g can be formally defined as a mapping:

$$g: \Omega \rightarrow \{0, 1\}. \tag{1}$$

In this paper, Ω is assumed to be a collection of *finite-length binary strings*. This simplification can be justified by the *data-preprocessing procedure* which *transforms* the raw data into *binary strings*. The *range* of the mapping g is simplified as a 0–1 action space. In terms of simple *market-timing* strategy, "1" means to "*act*" and "0" means to "*wait*." Here, for simplicity, we are only interested in *day trading*. So, "*act*" means to buy it at the opening time and sell it at the closing time.

Like all financial applications of GA, the start-off question is the *representation* issue. In our case, it is about how to effectively characterize the mapping g by a *finite-length binary string*, also known as a *chromosome* in GA. Research on this issue is very much motivated by the format of existing trading strategies, and there are generally two approaches to this issue. The first approach, called the *decision tree* approach, was pioneered by Bauer (1994). In this approach each trading strategy is represented by a decision tree. Bauer used bit strings to encode these decision tress, and generated and evolved them with genetic algorithms. The second approach, called the *combinatoric* approach, was first seen in Palmer et al. (1994). The combinatoric approach treats each trading strategy as one realization from $\binom{n}{k}$ combinations, where $l \leq k \leq n$, and n is the total number of given trading rules. Using GAs, one can encode the *inclusion* or *exclusion* of a

specific trading rule as a bit and the whole trading strategy as a bit string (chromosome).

Both approaches have very limited expression power. While various enhancements are possible, they all lead to non-standard GAs in the sense that their representations are not based on finite-length binary strings. Since the main focus of this paper is to illustrate a statistical foundation of the GA, we try to avoid all unnecessary complications, including the use of those non-standard representations. In other words, at this initial stage, we only make the illustration with the ordinary genetic algorithm (OGA), and, for that reason, Bauer's simple decision-tree representation is employed. However, it is clear that the statistical foundation presented in this paper is also applicable to GAs with different representations.

Bauer's decision-tree representation corresponds to the following general form of *trading strategies*

(**IF** (CONDS)
THEN (BUY AND SELL [**DAY TRADING**])
ELSE (WAIT)).

The CONDS appearing in the trading strategy is a *predicate*. CONDS itself is a logical composition of several primitive predicates. In this paper, all CONDSs are composed of three primitive predicates. Each primitive predicate can be represented as:

$$\text{Cond}(Z) = \begin{cases} 1(\text{True}), & \text{if } Z \oplus a, \\ 0(\text{False}), & \text{if } Z \ominus a \end{cases} \tag{2}$$

where Z, in our application, can be considered as a time series of returns indexed by t, e.g. r_{t-1}, r_{t-2}, etc., and a can be regarded as a *threshold* or *critical value* ($a \in \aleph$, a set of integers). $\oplus \in \{\geq, <\}$ and $\ominus = \{\geq, <\} - \oplus$. An example of CONDS with three primitive predicates is

$$\text{CONDS}(r_{t-1}, r_{t-2}, r_{t-3}) = \text{Cond}(r_{t-1}) \vee (\text{Cond}(r_{t-2}) \wedge \text{Cond}(r_{t-3})), \tag{3}$$

where "$\vee$" refers to the logic operator "OR," and "$\wedge$" refers to "AND."

Following Bauer, we use a 21-bit string to encode a trading strategy of this kind. Details can be found in the Appendix (Section A.1). Let G be the collection of all trading strategies encoded as above. Then the cardinality of G is 2^{21} ($\#(G) = 2^{21}$), which is more than 2 million. The search over the space G can be interpreted as a *numerical* algorithm as well as a *machine learning* algorithm for solving a mathematical optimization problem. Without losing generality, consider

the trading strategy with only *one* primitive predicate,

$$\text{Cond}(Z) = \begin{cases} 1(\text{True}), & \text{if } r_{t-1} \geq a, \\ 0(\text{False}), & \text{if } r_{t-1} < a. \end{cases} \tag{4}$$

Suppose the stochastic process of r_t is *strictly stationary* and denote the joint density of r_{t-1} and r_t by $f(r_{t-1}, r_t)$. In this simplest case, a trading strategy is parameterized by a single parameter a. Denote it by g_a. Then the optimal strategy g_{a^*} can be regarded as a solution to the optimization problem

$$\max_a E\left(\ln(\pi_n)\right), \tag{5}$$

where

$$\pi_n = \prod_{t=1}^{n}(1 + r_t) \tag{6}$$

is the accumulated returns of g_a over n consecutive periods. It can be shown that the solution to the problem (5) is

$$a^* = F^{-1}(0), \quad \text{if } F^{-1}(0) \text{ exists.} \tag{7}$$

where

$$F(a) = \int_{-\infty}^{\infty} \ln(1 + r_t) f(a, r_t)\, \mathrm{d}r_t \tag{8}$$

To solve Eq. (7), one has to know the density function of $f(r_{t-1}, r_t)$, which can only be inferred from the historical data. In this case, GAs are used as a *machine learning* tool to obtain an estimate of this joint density. Also, to arrive at a value for a^*, we have to know the inverse function of $F(a)$, which in general can only be solved numerically. In this case, GAs are used as a *numerical technique* to solve this problem. Therefore, in the trading-strategy problem, GAs are used simultaneously as a *numerical* technique and a *machine learning* tool to determine the critical parameter a^*. In the general case when CONDS has more than one predicate, the mathematical formulation of the problem can become very complicated, but the dual role of GAs remains unchanged. This interpretation justifies the mathematical significance of using GAs to discover the trading strategies.

The GA employed in this paper is a very basic version, which we shall call the ordinary genetic algorithm (*OGA*). In this study, we only focus on the OGA. Nonetheless, in a further study, it will be interesting to see whether a better result can be expected from advanced versions of GAs. The technical details of the OGA are given in the Appendix (Section A.2).

3. TESTBEDS

There are six stochastic processes used to evaluate the performance of GAs. They are:

(1) the linear stationary time series (also known as the Auto-Regressive and Moving-Average (*ARMA*) processes),
(2) the bilinear processes,
(3) the Auto-Regressive Conditional Heteroskedasticity (*ARCH*) processes,
(4) the Generalized ARCH (*GARCH*) processes,
(5) the threshold bilinear processes, and
(6) the chaotic processes.

All of the six classes have been frequently applied to modeling financial time series. Linear ARMA processes are found to be quite useful in *high-frequency financial data* (Campbell et al., 1997; Roll, 1984). Bilinear processes are often used to model the nonlinear dependence in both low- and high-frequency data (Drunat et al., 1998; Granger & Andersen, 1978). The ARCH processes are the most popular econometric tools for capturing the nonlinear dependence in the form of the second moment (Bollerslev et al., 1992). The threshold processes are good for asymmetric series and bursts (Tong, 1990). Finally, chaotic time series have been a topic of interest in finance over the last decade (Brock et al., 1991). Some details of these classes of processes are briefly reviewed from Sections 3.1 to 3.6.

These six processes are general enough to cover three important classes of dynamic processes, namely, linear stochastic processes, nonlinear stochastic processes, and nonlinear deterministic processes. This enables us to analyze the GA's performance in terms of some generic properties. For example, would it be easier for the GA to perform better with the linear (stochastic) process than with the nonlinear (stochastic) process, and with the deterministic (nonlinear) processes than with the stochastic (nonlinear) processes? The answers to these questions can certainly help us to delineate the effectiveness of GAs.

3.1. Linear Time Series

The linear time series model, also known as the *Auto-Regressive and Moving-Average* (*ARMA*(*p*,*q*)) model, was initiated by Box and Jenkings (1976). It has the following general form:

$$r_t = \mu + \sum_{i=1}^{p} \phi_i r_{t-i} + \sum_{j=1}^{q} \theta_j \varepsilon_{t-j} + \varepsilon_t, \tag{9}$$

Table 1. Data Generating Processes – ARMA

Code	Model	Parameters			
		ϕ_1	ϕ_2	θ_1	θ_2
L-1	ARMA(1,0)	0.3	0	0	0
L-2	ARMA(1,0)	0.6	0	0	0
L-3	ARMA(2,0)	0.3	−0.6	0	0
L-4	ARMA(2,0)	0.6	−0.3	0	0
L-5	ARMA(0,1)	0	0	0.3	0
L-6	ARMA(0,1)	0	0	0.6	0
L-7	ARMA(0,2)	0	0	0.3	−0.6
L-8	ARMA(0,2)	0	0	0.6	−0.3
L-9	ARMA(1,1)	0.3	0	−0.6	0
L-10	ARMA(1,1)	0.6	0	−0.3	0
L-11	ARMA(2,2)	0.4	−0.4	0.4	0.4
L-12	ARMA(2,2)	0.6	−0.3	−0.3	−0.6
L-13	White Noise	*Gaussian* (0, 0.1)			

where $\varepsilon_t \overset{\text{iid}}{\sim} N(0, \sigma^2)$. In all Monte Carlo simulations conducted in this paper, μ is set to 0 and σ^2 is set to 0.01. Thirteen ARMA(p,q) models were tested. The parameters of these thirteen ARMA(p,q) models are detailed in Table 1. Among these thirteen models, there are four pure AR models (L1–L4), four pure MA models (L5–L8), and four mixtures (L9–L12). The last one is simply *Gaussian white noise*.

3.2. Bilinear Process

The second class of stochastic processes considered in this paper is the *bilinear process* (BL), which was first studied by Granger and Anderson (1978), and subsequently by Subba-Rao (1981) and Subba-Rao and Gabr (1980). The BL process is constructed simply by adding the cross-product terms of r_{t-i} and ε_{t-j} to a linear ARMA process so it can be regarded as a second-order nonlinear time series model. In other words, if the parameters of all cross-product terms are zero, then the BL process can be reduced to the ARMA process.

The general form of a bilinear process, BL(p, q, u, v) is:

$$r_t = \mu + \sum_{i=1}^{p} \phi_i r_{t-i} + \sum_{j=1}^{q} \theta_j \varepsilon_{t-j} + \sum_{m=1}^{u} \sum_{n=1}^{v} \psi_{mn} r_{t-m} \varepsilon_{t-n} + \varepsilon_t, \qquad (10)$$

where $\varepsilon_t \overset{\text{iid}}{\sim} N(0, \sigma^2)$. Eight specific bilinear processes are employed for our Monte-Carlo simulation. In all of these processes, $\mu = 0$ and $\sigma^2 = 0.01$. Other parameters

Table 2. Data Generating Processes – Bilinear.

Code	Model	Parameters					
		ϕ_1	θ_1	ψ_{11}	ψ_{12}	ψ_{21}	ψ_{22}
BL-1	BL(0,0,1,1)	0	0	0.6	0	0	0
BL-2	BL(0,0,1,1)	0	0	0.3	0	0	0
BL-3	BL(0,1,1,2)	0	0.3	0	0.6	0	0
BL-4	BL(0,1,1,2)	0	0.6	0	0.3	0	0
BL-5	BL(1,0,2,1)	0.3	0	0	0	0.6	0
BL-6	BL(1,0,2,1)	0.6	0	0	0	0.3	0
BL-7	BL(1,1,2,2)	0.3	0.3	0	0	0	0.3
BL-8	BL(1,1,2,2)	0.3	0.3	0	0	0	0.6

are given in Table 2. Notice that the first two (BL-1, BL-2) do not have the linear component, and only the nonlinear cross-product terms are presented.

3.3. ARCH Processes

The third class of models considered is the *Auto-Regressive Conditional Heteroskedasticity* (ARCH) *process* introduced by Engle (1982), which has played a dominant role in the field of financial econometrics. The ARCH process is mainly used to replicate the three stylized facts of financial time series, namely, the fat-tailed marginal distribution of returns, the time-variant volatility of the returns, and clustering outliers. Consequently, unlike the ARMA process, ARCH mainly works only on the second moment, rather than the first moment. Nonetheless, by combining the two, one can attach an ARMA(p, q) process with an ARCH (q') process, called the ARMA(p, q)-ARCH(q') process. Its general form is

$$r_t = \mu + \sum_{i=1}^{p} \phi_i r_{t-i} + \sum_{j=1}^{q} \theta_j \varepsilon_{t-j} + \varepsilon_t \tag{11}$$

$$\varepsilon_t | \Omega_{t-1} \sim N(0, \sigma_t^2) \tag{12}$$

$$\sigma_t^2 = \omega + \sum_{m=1}^{q'} \alpha_m \varepsilon_{t-m}^2 \tag{13}$$

where $\omega > 0, \sigma_m \geq 0, m = 1, \ldots, q'$ and Ω_t denotes the information set available at time t.

Table 3. Data Generating Processes – ARCH.

Code	Model	Parameters				
		ω	α_1	α_2	ϕ_1	θ_1
AH-1	AR(0)-ARCH(1)	0.005	0.3	0	0	0
AH-2	AR(0)-ARCH(1)	0.005	0.6	0	0	0
AH-3	AR(0)-ARCH(2)	0.001	0.3	0.5	0	0
AH-4	AR(0)-ARCH(2)	0.001	0.5	0.3	0	0
AH-5	AR(1)-ARCH(1)	0.005	0.6	0	0.6	0
AH-6	AR(1)-ARCH(2)	0.001	0.5	0.3	0.6	0
AH-7	MA(1)-ARCH(1)	0.005	0.3	0	0	−0.6

Seven ARCH processes are included in this study. They share a common value of μ, which is 0. Values of other parameters are detailed in Table 3. Notice that the first four processes do not have linear signals ($\phi_1 = 0$, $\theta_1 = 0$), whereas the fifth and the sixth processes are associated with an AR(1) linear signal ($\phi_1 = 0.6$), and the last process has a MA(1) linear signal ($\theta_1 = -0.6$).

3.4. GARCH Processes

A generalized version of the ARCH process, known as the *generalized ARCH* (GARCH) process, was introduced by Bollerslev (1986). GARCH generalizes Engle's ARCH process by adding additional conditional autoregressive terms. An ARMA(p, q) process with a GARCH error term of order (p', q'), ARMA(p, q)-GARCH(p', q'), can be written as

$$r_t = \mu + \sum_{i=1}^{p} \phi_i r_{t-i} + \sum_{j=1}^{q} \theta_j \varepsilon_{t-j} + \varepsilon_t \tag{14}$$

$$\varepsilon_t | \Omega_{t-1} \sim N(0, \sigma_t^2) \tag{15}$$

$$\sigma_t^2 = \omega + \sum_{m=1}^{q'} \alpha_m \varepsilon_{t-m}^2 + \sum_{n=1}^{p'} \beta_n \sigma_{t-n}^2 \tag{16}$$

with $\omega > 0, \alpha_m = 0$ and $\beta_n \geq 0, m = 1, \ldots, q', n = 1, \ldots, p'$. Again, Ω_t denotes the information set available at time t.

Nine GARCH processes are attempted. In all cases, $\mu = 0$ and $\omega = 0.001$. Specifications of other parameters are given in Table 4. The 7th, 8th and 9th models (GH-7, GH-8, GH-9) are AR(1) processes combined with a GARCH error term,

Table 4. Data Generating Processes – GARCH.

Code	Model	Parameters					
		β_1	β_2	α_1	α_2	ϕ_1	θ_1
GH-1	AR(0)-GARCH(1,1)	0.3	0	0.5	0	0	0
GH-2	AR(0)-GARCH(1,1)	0.5	0	0.3	0	0	0
GH-3	AR(0)-GARCH(1,2)	0.2	0	0.2	0.4	0	0
GH-4	AR(0)-GARCH(1,2)	0.2	0	0.4	0.2	0	0
GH-5	AR(0)-GARCH(2,1)	0.2	0.4	0.2	0	0	0
GH-6	AR(0)-GARCH(2,1)	0.4	0.2	0.2	0	0	0
GH-7	AR(1)-GARCH(1,1)	0.5	0	0.3	0	0.6	0
GH-8	AR(1)-GARCH(1,2)	0.2	0	0.4	0.2	0.6	0
GH-9	AR(1)-GARCH(2,1)	0.4	0.2	0.2	0	0.6	0
GH-10	MA(1)-GARCH(1,1)	0.3	0	0.5	0	0	−0.6

whereas the last model (GH-10) is a MA(1) process plus a GARCH error term. For the remaining six, there are no linear signals but just pure GARCH processes.

3.5. Threshold Processes

Tong (1983) proposed a *threshold autoregressive* (TAR) model which is of the form,

$$r_t = \mu^{(l)} + \sum_{i=1}^{p} \phi_i^{(l)} r_{t-i} + \varepsilon_t \tag{17}$$

if $r_{t-d} \in \Omega_1$ $(l = 1, 2, \ldots, k)$, where $\Omega_i \cap \Omega_j = \varnothing$ $(i, j = 1, \ldots, k)$ if $i \neq j$ and $\cup_{l=1}^{k} \Omega_l = \Re$. The parameter k represents the number of thresholds and d is called the threshold lag (or delay parameter). Producing various limit cycles is one of the important features of the threshold models, and the TAR process can be applied to the time series which has an asymmetric cyclical form.

The threshold idea can be used as a module to add and to extend other processes. Here, we apply the threshold idea to the bilinear process (10), and extend it to a threshold bilinear (TBL) process. Let us denote a bilinear process (BL(p, q, u, v)) with k-thresholds by TBL(k, p, q, u, v), which can be written as

$$r_t = \mu^{(l)} + \sum_{i=1}^{p} \phi_i^{(l)} r_{t-i} + \sum_{j=1}^{q} \theta_j^{(l)} \varepsilon_{t-j} + \sum_{m=1}^{u} \sum_{n=1}^{v} \psi_{mn}^{(l)} r_{t-m} \varepsilon_{t-n} + \varepsilon_t \tag{18}$$

Table 5. Data Generating Processes – Threshold Processes.

Code	Model	Parameters						
		$\phi_1^{(1)}$	$\phi_2^{(1)}$	$\theta_1^{(1)}$	$\psi_{11}^{(1)}$	$\psi_{12}^{(1)}$	$\psi_{21}^{(1)}$	$\psi_{22}^{(1)}$
		$\phi_1^{(2)}$	$\phi_2^{(2)}$	$\theta_1^{(2)}$	$\psi_{11}^{(2)}$	$\psi_{12}^{(2)}$	$\psi_{21}^{(2)}$	$\psi_{22}^{(2)}$
TH-1	TBL(2;1,0,0,0)	0.3	0	0	0	0	0	0
		0.6	0	0	0	0	0	0
TH-2	TBL(2;1,1,0,0)	0.3	0	0.6	0	0	0	0
		0.6	0	0.3	0	0	0	0
TH-3	TBL(2;0,0,1,1)	0	0	0	0.3	0	0	0
		0	0	0	0.6	0	0	0
TH-4	TBL(2;1,1,2,2)	0.3	0	0	0	0	0.6	0
		0	0	0.3	0	0.6	0	0
TH-5	TBL(2;2,0,2,2)	0	0	0	0.3	0	0	−0.6
		0.3	–0.6	0	0	0	0	0

Note: The lag period d is set to 1 and $\mu^{(1)} = \mu^{(2)} = 0$ in all of the models. In addition, $\Omega_1 \equiv \{r_{t-d} \,|r_{t-d} \geq 0\}$ and $\Omega_2 \equiv \{r_{t-d} \,|r_{t-d} < 0\}$.

It is trivial to show that TBL can be reduced to a threshold ARMA if $\psi_{mm}^{(l)} = 0$ for all m, n and l. Table 5 lists the five TBL processes considered in this paper. The motives for choosing these five series will become clear when we come to Section 6.4.

3.6. Chaotic Processes

All of the above-mentioned processes are stochastic. However, the time series that appear to be random does not necessary imply that they are generated from a stochastic process. Chaotic time series as an alternative description of this seemingly random phenomenon was a popular econometrics topic in the 1990s. While it is hard to believe that a financial time series is just a deterministic chaotic time series, the chaotic process can still be an important module for the working of a nonlinear time series. Five chaotic processes are employed in this study.

C-1: Logistic Map

$$r_t = 4r_{t-1}(1 - r_{t-1}), \quad r_t \in [0, 1] \quad \forall t \tag{19}$$

C-2: Henon Map

$$r_t = 1 + 0.3r_{t-2} - 1.4r_{t-1}^2, \quad r_{-1}, r_0 \in [-1, 1] \tag{20}$$

C-3: Tent Map

$$\begin{cases} r_t = 2r_{t-1}, & \text{if } 0 \le r_{t-1} < 0.5, \\ r_t = 2(1 - r_{t-1}), & \text{if } 0.5 \le r_{t-1} \le 1. \end{cases} \tag{21}$$

C-4: Poly. 3

$$r_t = 4r_{t-1}^3 - 3r_{t-1}, \quad r_t \in [-1, 1] \quad \forall t \tag{22}$$

C-5: Poly. 4

$$r_t = 8r_{t-1}^4 - 8r_{t-1}^2 + 1, \quad r_t \in [-1, 1] \quad \forall t \tag{23}$$

The series generated by all these stochastic processes (from Sections 3.1 to 3.6) may have a range which does not fit the range of the normal return series. For example, the process (19) is always positive. As a result, a contracting or a dilating map is needed. We, therefore, contract or dilate all series linearly and monotonically into an acceptable range, which is $(-0.3, 0.3)$ in this paper.

4. PERFORMANCE CRITERIA AND STATISTICAL TESTS

Basic performance metrics to evaluate the performance of trading strategies have long existed in the literature. Following Refenes (1995), we consider the following four main criteria: *returns*, the *winning probability*, the *Sharpe ratio* and the *luck coefficient*. In this paper, the performance of the trading strategies generated by the ordinal genetic algorithm (OGA) is compared with that using a benchmark based on these four criteria. To make the evaluation process rigorous, performance differences between the OGA-based trading strategies and the benchmark are tested *statistically*. Tests for returns and winning probability are straightforward. Tests for the Sharpe ratio are available in the literature (see, for example, Jobson and Korkie (1981) and Arnold (1990)). However, tests for the luck coefficient are more demanding, and it has not been derived in the literature. In this paper, we develop asymptotic tests for the luck coefficient.

4.1. Returns

Let X and Y be the accumulated returns of an one-dollar investment by applying OGA-based trading strategies and the benchmark strategy, say, the buy-and-hold (B&H) strategy, respectively. Assume that $E(X) = \mu$ and $E(Y) = \nu$. Let us estimate

the μ and ν by the respective *sample average* $\bar{\pi}^2$ and $\bar{\pi}^1$ via the Monte Carlo simulation. Then one can test the null

$$H_0 : \mu - \nu \leq 0, \tag{24}$$

with the following test statistic

$$Z_\pi = \frac{\sqrt{n}(\bar{\pi}^2 - \bar{\pi}^1)}{(\hat{\sigma}^2 + \hat{\tau}^2 - 2\hat{\rho}\hat{\sigma}\hat{\tau})^{1/2}}, \tag{25}$$

where $\hat{\sigma}^2$ and $\hat{\tau}^2$ are the sample variances of X and Y, $\hat{\rho}$ is the sample correlation coefficient of X and Y, and n is the sample size (the number of ensembles generated during the Monte Carlo simulation). By using the central limit theorem, it is straightforward to show that Z_π is an asymptotically standard normal test.

While testing the difference between $\bar{\pi}^2$ and $\bar{\pi}^1$ can tell us the performance of the GA as opposed to a benchmark, it provides us with nothing more than a point evaluation. In some cases, we may also wish to know whether the superiority, if shown, can extend to a large class of trading strategies. A common way to address this question is to introduce an *omniscient* trader. Let us denote the respective accumulated returns earned by this omniscient trader as $\bar{\pi}^*$.[4] Now, subtracting $\bar{\pi}^1$ from $\bar{\pi}^*$ gives us the total unrealized gain, if we only know the benchmark. Then, the ratio, also called the *exploitation ratio*,

$$\tilde{\pi} \equiv \frac{\bar{\pi}^2 - \bar{\pi}^1}{\bar{\pi}^* - \bar{\pi}^1} \tag{26}$$

is a measure of the size of those unrealized gains which can be exploited by using a GA. Based on its formulation, $\tilde{\pi}$ may be positive, negative or zero, but has one as its maximum. If $\tilde{\pi}$ is not only positive, but is also close to one, then its superiority is not just restricted to the benchmark, but may also have global significance.

In addition to the accumulated gross returns, one can also base the comparison on the excess return by simply subtracting one from the accumulated gross returns. A relative superiority measure of the GA as opposed to the benchmark can be defined accordingly as

$$\dot{\pi} \equiv \frac{(\bar{\pi}^2 - 1) - (\bar{\pi}^1 - 1)}{|\bar{\pi}^1 - 1|} = \frac{\bar{\pi}^2 - \bar{\pi}^1}{|\bar{\pi}^1 - 1|}. \tag{27}$$

4.2. Winning Probability

The mean return can sometimes be sensitive to outliers. Therefore, it is also desirable to base our performance criterion on some robust statistics, and the

winning probability is one of this kind. The winning probability basically tells us, by randomly picking up an ensemble from one stochastic process, the probability that the GA will win. Formally, let (X, Y) be a random vector with the joint density function $h(x, y)$. Then p_w, defined as follows, is called the *winning probability*.

$$p_w = \Pr(X > Y) = \int\int_{x>y} h(x, y)\, dx\, dy \tag{28}$$

Based on the winning probability, we can say that X is *superior* to Y if $p_w > 0.5$, and *inferior* to Y if $p_w < 0.5$, and *equivalent* to Y if $p_w = 0.5$. The null hypothesis to test is

$$H_0: p_w \leq 0.5 \tag{29}$$

The rejection of (29) shows the superiority of the GA over the benchmark. An asymptotic standard normal test of (29) can be derived as

$$Z_w = \frac{\sqrt{n}(\hat{p}_w - 0.5)}{\sqrt{\hat{p}_w(1 - \hat{p}_w)}} \tag{30}$$

where $\hat{p}_w$ is the sample counterpart of p_w.

4.3. Sharpe Ratio

One criterion which has been frequently ignored by machine learning people in finance is the *risk* associated with a trading rule. Normally, a higher profit known as the *risk premium* is expected when the associated risk is higher. Without taking the risk into account, we might exaggerate the profit performance of a highly risky trading rule. Therefore, to evaluate the performance of our GA-based trading rule on a risk-adjusted basis, we have employed the well-known *Sharpe ratio* as the third performance criterion (Sharpe, 1966). The Sharpe ratio s is defined as the excess return divided by a risk measure. The higher the Sharpe ratio, the higher the risk-adjusted return.

Formally, let $X \sim f(x)$ with $E(X) = \mu$ and $\mathrm{Var}(X) = \sigma^2$. Then the value

$$s = \frac{\mu - c}{\sigma} \tag{31}$$

is called the *Sharpe ratio* of X where c is one plus a risk-free rate. Furthermore, to compare the performance of two trading strategies in the Sharpe ratio, let $X \sim f(x)$ and $Y \sim g(y)$ with $E(X) = \mu$, $E(Y) = \nu$, $\mathrm{Var}(X) = \sigma^2$ and $\mathrm{Var}(Y) = \tau^2$. Then the difference

$$d = \frac{\mu - c}{\sigma} - \frac{\nu - c}{\tau} \tag{32}$$

is called the *Sharpe-ratio differential* between X and Y. Accordingly, X is said to have a *higher* (*lower*) Sharpe ratio relative to Y if $d > 0$ ($d < 0$). Otherwise, X and Y are said to be *identical* in terms of the Sharpe ratio.

Jobson and Korkie (1981) derive an asymptotic standard normal test for the Sharpe-ratio differential. However, we do not follow their Taylor expansion formulation. Instead, by applying *Slutzky's theorem*, the *Cramer* δ *theorem*, and the *multivariate central limit theorem*, a standard normal test for the null

$$H_0 : d \leq 0 \tag{33}$$

can be derived as follows:

$$Z_d = \frac{\sqrt{n}(\hat{d} - d)}{\hat{\omega}_1}, \tag{34}$$

where

$$\hat{d} = \frac{\bar{\pi}^2 - c}{\hat{\sigma}} - \frac{\bar{\pi}^1 - c}{\hat{\tau}}, \tag{35}$$

and

$$\begin{aligned}\hat{\omega}_1^2 = {} & 2(1 - \hat{\rho}) + \frac{(\bar{\pi}^2 - c)}{\hat{\sigma}}(\hat{\theta} - \hat{\delta}) + \frac{(\bar{\pi}^1 - c)}{\hat{\tau}}(\hat{\psi} - \hat{\xi}) \\ & - \frac{(\bar{\pi}^2 - c)(\bar{\pi}^1 - c)}{\hat{\sigma}\hat{\tau}}\frac{(\hat{\phi} - 1)}{2} + \frac{(\bar{\pi}^2 - c)^2}{\hat{\sigma}^2}\frac{(\hat{\gamma} - 1)}{4} + \frac{(\bar{\pi}^1 - c)^2}{\hat{\tau}}\frac{(\hat{\eta} - 1)}{4}\end{aligned} \tag{36}$$

$\hat{\delta}$, $\hat{\gamma}$, $\hat{\xi}$ and $\hat{\eta}$ are the corresponding sample third and fourth moments of X and Y, whereas $\hat{\rho}$, $\hat{\theta}$, $\hat{\psi}$, $\hat{\phi}$ are the corresponding sample mixed moments between X and Y (also expressed as Eq. (37)).

$$\begin{bmatrix} \dfrac{E(X-\mu)^3}{\sigma^3} \\ \dfrac{E(X-\mu)^4}{\sigma^4} \\ \dfrac{E(Y-\nu)^3}{\tau^3} \\ \dfrac{E(Y-\nu)^4}{\tau^4} \end{bmatrix} = \begin{bmatrix} \delta \\ \gamma \\ \xi \\ \eta \end{bmatrix}, \begin{bmatrix} \dfrac{E(X-\mu)E(X-\nu)}{\sigma\tau} \\ \dfrac{E(X-\mu)^2(Y-\nu)}{\sigma^2\tau} \\ \dfrac{E(X-\mu)(Y-\nu)^2}{\sigma\tau^2} \\ \dfrac{E(x-\mu)^2(Y-\nu)^2}{\sigma^2\tau^2} \end{bmatrix} = \begin{bmatrix} \rho \\ \theta \\ \psi \\ \phi \end{bmatrix} \tag{37}$$

4.4. Luck Coefficient

The largest positive trade can be very important if it makes a significant contribution towards skewing the average profit dramatically. When this happens, people can be severely misled by the sample mean. As a solution to this problem, the *trimmed mean* is often used in statistics. A similar idea in finance is known as the *luck coefficient*. The luck coefficient l_ε is defined as the sum of the largest $100\varepsilon\%$ returns, $\varepsilon \in (0, 1)$, divided by the sum of total returns. In a sense, the larger the luck coefficient, the weaker the reliability of the performance. The luck coefficient, as a performance statistic, is formally described below.

Let $\{X_1, X_2, \ldots, X_m\}$ be a random sample from $f(x)$ with $E(X) = \mu$. The order statistic of this random sample can be enumerated as $X_{(1)}, X_{(2)}, \ldots, X_{(m)}$, where $X_{(1)} \le X_{(2)} \le \cdots \le X_{(m)}$. Then, from the *order statistics*, it is well known that

$$X_{(m)} \sim g(x_{(m)}) = m[F(x_{(m)})]^{m-1} f(x_{(m)}) \tag{38}$$

where F is the *distribution function* of X. Furthermore, let $X_i \overset{\text{iid}}{\sim} f(x)$, $i = 1, 2, \ldots, m$ and $X_{(m)} \sim g(x_{(m)})$ as described above with $E(X_{(m)}) = \mu_\varepsilon$. Then the ratio

$$l_\varepsilon = \frac{\varepsilon \mu_\varepsilon}{\mu} \tag{39}$$

is called the *luck coefficient* of X where $\varepsilon = \frac{1}{m}$. In this study, ε is set to 0.05. Here we want to see how much of the contribution to mean returns comes from the largest 5% of trades.

For making a comparison between strategies, the *luck-coefficient ratio* is defined as follows. Let $X_i \overset{\text{iid}}{\sim} f_x(x)$ with $E(X) = \mu$, $Y_i \overset{\text{iid}}{\sim} f_y(y)$ with $E(Y) = \nu$, $i = 1, 2, \ldots, m$ and $X_{(m)} \sim g_x(x_{(m)})$ with $E(X_{(m)}) = \mu_\varepsilon$, $Y_{(m)} \sim g_y(y_{(m)})$ with $E(Y_{(m)}) = \nu_\varepsilon$. Then the ratio

$$r_\varepsilon = \frac{\varepsilon \nu_\varepsilon / \nu}{\varepsilon \mu_\varepsilon / \mu} = \frac{\mu \nu_\varepsilon}{\nu \mu_\varepsilon} \tag{40}$$

is called the *luck-coefficient ratio* of X relative to Y where $\varepsilon = \frac{1}{m}$. Based on this definition, X is said to have a *lower* (*higher*) luck coefficient relative to Y if $r_\varepsilon > 1$ ($r_\varepsilon < 1$). Otherwise, X and Y are said to be *identical* in terms of the luck coefficient. However, to the best of our knowledge, the respective asymptotic standard normal test for the null

$$H_0 : r_\varepsilon \le 1 \tag{41}$$

is not available in the literature. Nevertheless, similar to the derivation of the test of the Sharpe ratio (34), it is not hard to cook up such a test by using *Slutzky's theorem*, the *Cramer* δ *theorem*, and the *multivariate central limit theorem*, which

is given in Eq. (42)

$$Z_r = \frac{\sqrt{n}(\hat{r}_\varepsilon - r_\varepsilon)}{\hat{\omega}_2}, \tag{42}$$

where

$$\hat{r}_\varepsilon = \frac{\bar{\pi}^2 \bar{\pi}^1_m}{\bar{\pi}^1 \bar{\pi}^2_m}, \tag{43}$$

and

$$\begin{aligned}\hat{\omega}_2^2 = {} & \frac{\varepsilon(\bar{\pi}^1_m)^2}{(\bar{\pi}^1)^2(\bar{\pi}^2_m)^2}\left(\hat{\sigma}^2 + \frac{(\bar{\pi}^2)^2\hat{\tau}^2}{(\bar{\pi}^2)^2}\right) + \frac{(\bar{\pi}^2)^2}{(\bar{\pi}^1)^2(\bar{\pi}^2_m)^2}\left(\hat{\tau}^2_\varepsilon + \frac{(\bar{\pi}^1_m)^2\hat{\sigma}^2_\varepsilon}{(\bar{\pi}^2_m)^2}\right) \\ & - \frac{2\bar{\pi}^2\bar{\pi}^1_m\hat{\tau}}{(\bar{\pi}^1)^3(\bar{\pi}^2_m)^2}(\varepsilon\bar{\pi}^1_m\hat{\rho}\hat{\sigma} + \bar{\pi}^2\hat{\lambda}\hat{\tau}_\varepsilon) - \frac{2\bar{\pi}^2\bar{\pi}^1_m\hat{\sigma}_\varepsilon}{(\bar{\pi}^1)^2(\bar{\pi}^2_m)^3}(\bar{\pi}^1_m\hat{\sigma}\hat{\iota} + \bar{\pi}^2\hat{\tau}_\varepsilon\hat{\zeta}) \\ & + \frac{2\bar{\pi}^2\bar{\pi}^1_m}{(\bar{\pi}^1)^2(\bar{\pi}^2_m)^2}\left(\hat{\sigma}\hat{\kappa}\hat{\tau}_\varepsilon + \frac{\bar{\pi}^2\bar{\pi}^1_m\hat{\sigma}_\varepsilon\hat{o}\hat{\tau}}{\bar{\pi}^1\bar{\pi}^2_m}\right).\end{aligned} \tag{44}$$

$\bar{\pi}^1_m$ and $\bar{\pi}^2_m$ are the corresponding sample means of $Y_{(m)}$ and $X_{(m)}$. $\hat{\tau}^2_\varepsilon$ and $\hat{\sigma}^2_\varepsilon$ are the corresponding sample variances of $Y_{(m)}$ and $X_{(m)}$, and $\hat{\rho}$, $\hat{\zeta}$, $\hat{\kappa}$, $\hat{\iota}$, $\hat{\lambda}$, and $\hat{o}$ are the corresponding sample correlation coefficients as indicated in Eq. (45).

$$\begin{bmatrix} \mathrm{corr}(X_i, Y_i) \\ \mathrm{corr}(X_{(m)}, Y_{(m)}) \\ \mathrm{corr}(X_i, Y_{(m)}) \end{bmatrix} = \begin{bmatrix} \rho \\ \zeta \\ \kappa \end{bmatrix}, \begin{bmatrix} \mathrm{corr}(X_i, X_{(m)}) \\ \mathrm{corr}(Y_i, Y_{(m)}) \\ \mathrm{corr}(Y_i, X_{(m)}) \end{bmatrix} = \begin{bmatrix} \iota \\ \lambda \\ o \end{bmatrix} \tag{45}$$

5. MONTE CARLO SIMULATION

Since it is hard to obtain analytical results of the performance of the GA in relation to various stochastic processes, Monte Carlo simulation methodology is used in this study. Each stochastic process listed in Tables 1–5 and Eqs (19) to (23) is used to generate 1000 *independent* time series, each with 105 observations ($\{r_t\}_{t=1}^{105}$).[5] For each series, the first 70 observations ($\{r_t\}_{t=1}^{70}$) are taken as the training sample, and the last 35 observations ($\{r_t\}_{t=76}^{105}$) are used as the testing sample. The OGA are then employed to extract trading strategies from these training samples. These strategies are further tested by the testing samples, and the resulting accumulated returns (π) are calculated, i.e.

$$\pi = \prod_{t=76}^{105}(1 + r_t) \tag{46}$$

In the meantime, the accumulated returns of the benchmark are also calculated. In following convention, our choice of the benchmark is simply the buy-and-hold (B&H) strategy.

Let π_i^1 ($i = 1, 2, \ldots, 1000$) be the accumulated returns of the B&H strategy when tested on the ith ensemble of a stochastic process, and π_i^2 be the accumulated returns of the OGA when tested on the same ensemble. The issue which we shall address, given the set of observations $S(\equiv \{\pi_i^1, \pi_i^2\}_{i=1}^{1000})$, is to decide *whether the OGA-based trading strategies can statistically significantly outperform the B&H strategy under the stochastic process in question*. The answers are given in the next section.

6. TEST RESULTS

6.1. ARMA Processes

We start our analysis from the linear stochastic processes. Table 6 summarizes the statistics defined in Section 4. Several interesting features stand out. First, from the statistics $\hat{p}_w$ and z_w, it can be inferred that, in accumulated returns, the probability that the OGA-based trading strategies can beat the B&H strategy *is significantly greater than 0.5*. For the stochastic processes with linear signals (L-1–L-12), the winning probability $\hat{p}_w$ ranges from 0.713 (L-5) to 0.991 (L-12). What, however, seems a little puzzling is that, even in the case of *white noise* (L-13), the GA can also beat B&H statistically significantly, while with much lower winning probabilities p_w (0.606). This seemingly puzzling finding may be due to the fact that a pseudo-random generator can actually generate a series with signals *when the sample size is small*. For example, Chen and Tan (1999) show that, when the sample size is 50, the probability of having signals in a series generated from a pseudo-random generator is about 5%, while that probability can go to zero when the sample size is 1000. Therefore, by supposing that the OGA-based trading strategies can win in all these atypical ensembles and get even with the B&H strategy in other normal ensembles, then $\hat{p}_w$ can still be significantly greater than 0.5.

Second, by directly comparing $\bar{\pi}^1$ with $\bar{\pi}^2$, we can see that, except for the case of white noise, the OGA-based trading strategies unanimously outperform the B&H strategy *numerically* in all linear ARMA*(p, q)* processes. From the $\dot{\pi}$ statistic (27), we see that the triumph of GA over B&H extends from a low of 19% (L-10) to a high of 916% (L-3). The z_π statistic, ranging from 2.12 to 47.39, signifies the statistical significance of these differences. Third, to see how the GA effectively exploited the excess potential returns earned by the omniscient trader, $\tilde{\pi}$ is also included in Table 6. There it is observed that the GA exploited 2–31%

Table 6. Performance Statistics of the OGA and B&H – ARMA.

Code	Model	$\bar{\pi}^1$	$\bar{\pi}^2$	$\bar{\pi}^*$	z_π	$\tilde{\pi}$ (%)	$\dot{\pi}$ (%)	$\hat{p}_w$	z_w
L-1	ARMA(1,0)	1.198	1.355	4.388	6.33	4	20	0.732	16.56
L-2	ARMA(1,0)	1.992	2.868	6.658	13.67	19	88	0.859	32.62
L-3	ARMA(2,0)	0.845	2.265	5.480	42.98	31	916	0.976	98.35
L-4	ARMA(2,0)	1.123	1.185	5.170	27.08	2	50	0.896	41.02
L-5	ARMA(0,1)	1.103	1.269	4.241	7.63	5	161	0.713	14.89
L-6	ARMA(0,1)	1.199	1.775	5.166	20.61	15	289	0.861	32.99
L-7	ARMA(0,2)	0.853	1.633	5.104	39.97	18	531	0.926	51.46
L-8	ARMA(0,2)	1.065	1.522	5.285	21.58	11	703	0.848	30.65
L-9	ARMA(1,1)	0.898	1.229	4.128	24.55	10	325	0.812	25.25
L-10	ARMA(1,1)	1.452	1.538	4.783	2.12	3	19	0.721	15.58
L-11	ARMA(2,2)	1.306	2.588	6.957	30.43	23	419	0.927	51.90
L-12	ARMA(2,2)	0.721	2.167	6.189	47.39	26	518	0.991	164.40
L-13	ARMA(0,0)	0.983	0.993	3.881	0.67	0	59	0.606	6.85
Code	Model	$\hat{s}_1$	$\hat{s}_2$	$\hat{d}$	z_d	$\hat{l}^1_{0.05}$	$\hat{l}^2_{0.05}$	$\hat{r}_{0.05}$	z_r
L-1	ARMA(1,0)	0.166	0.438	0.272	11.74	0.179	0.126	1.416	3.32
L-2	ARMA(1,0)	0.236	0.526	0.290	8.40	0.310	0.214	1.450	1.75
L-3	ARMA(2,0)	−0.342	1.181	1.523	32.14	0.115	0.106	1.087	1.68
L-4	ARMA(2,0)	0.111	0.877	0.767	24.53	0.182	0.114	1.594	4.45
L-5	ARMA(0,1)	0.110	0.419	0.309	13.40	0.169	0.117	1.449	4.23
L-6	ARMA(0,1)	0.135	0.602	0.467	5.02	0.216	0.138	1.563	2.48
L-7	ARMA(0,2)	−0.353	0.948	1.301	27.67	0.108	0.099	1.092	1.68
L-8	ARMA(0,2)	0.065	0.624	0.559	18.18	0.181	0.120	1.509	4.18
L-9	ARMA(1,1)	−0.307	0.524	0.831	22.43	0.093	0.092	1.007	0.16
L-10	ARMA(1,1)	0.214	0.392	0.177	5.39	0.263	0.171	1.534	2.50
L-11	ARMA(2,2)	0.170	0.854	0.684	11.19	0.240	0.141	1.708	3.34
L-12	ARMA(2,2)	−1.363	1.224	2.587	36.46	0.083	0.105	0.795	−6.21
L-13	ARMA(0,0)	−0.025	−0.016	0.010	0.37	0.130	0.096	1.353	3.90

Note: $\bar{\pi}^1$, $\bar{\pi}^2$ and $\bar{\pi}^*$ are the respective sample mean return of OGA, B&H and the omniscient trader. $\tilde{\pi}$ is the exploitation ratio (Eq. (26)), and $\dot{\pi}$ is the relative superiority index (Eq. (27)). $\hat{p}_w$ is the sample winning probability of OGA over B&H (Eq. (28)). $\hat{s}_1$ and $\hat{s}_2$ are the corresponding sample Sharpe ratio of OGA and B&H (Eq. (31)). Their sample difference is $\hat{d}$ (Eq. (32)). $\hat{l}^1_{0.05}$ and $\hat{l}^2_{0.05}$ are the sample luck coefficient of OGA and B&H (Eq. (39)), and $\hat{r}_{0.05}$ is the sample luck coefficient ratio between the two (Eq. (40)). The z_π, z_w, z_d and z_r are the test statistics of the mean return difference, winning probability, Sharpe ratio differential, and luck coefficient ratio, respectively. The critical value of them is 1.28 at the 10% significance level, and is 1.64 at the 5% significance level.

of the potential excess returns. However, as we expect, it was to no avail when the scenario changed to white noise.

As mentioned earlier, we should not judge the performance of the GA solely by the profitability criterion. The risk is a major concern in business practice.

We, therefore, have also calculated the Sharpe ratio, a risk-adjusted profitability criterion. It is interesting to notice that in all cases the Sharpe-ratio differential ($\hat{d}$) is positive. In other words, the GA still outperforms B&H even after taking into account the risk. The test of this differential also lends support to its statistical significance.

Finally, we examine whether the GA wins just by *luck* in the sense that its return performance depends heavily on its best 5% trades. Based on the statistic of luck coefficient $\hat{r}_{0.05}$, it is found that in only one of the 13 cases, i.e. the case L-12, dose the GA have a higher luck coefficient; in the other 12 cases, the luck-coefficient ratios are larger than 1, meaning that the dominance of the GA over B&H cannot be attributed to the presence of a few abnormally large returns. From the test z_r, this result is again significant except for the case L-9. All in all, we can conclude that if the return follows a simple linear ARMA process, then the superior performance of the GA compared to B&H is expected.

6.2. Bilinear Processes

By moving into the bilinear processes, we are testing the effectiveness of the GA when the return series is *nonlinear*. Table 7 summarizes all the key statistics. Obviously, the performance of the GA is not as glamorous as before. Out of the eight battles, it loses twice (cases BL-1 and BL-2) to B&H (see z_π and z_w). Taking the risk into account would not help reverse the situation (see z_d). It is, however, interesting to notice a unique feature shared by BL-1 and BL-2. As mentioned in Section 3.2, the two stochastic processes do not have any linear component (all ϕ_i and θ_j in Eq. (10) or Table 2 are zero). In other words, these two cases are *pure nonlinear* (pure bilinear). If some linear components are added back to the series, then the significant dominance of the GA does come back. This is exactly what happens in the other six cases (BL-3 to BL-8), which all have the ARMA component as a part (Table 2).

Even for the six cases where the GA wins, we can still observe some adverse impacts of nonlinearity on the GA. Roughly speaking, Table 7 shows that the distribution of both $\dot{\pi}$ and $\tilde{\pi}$ becomes lower as opposed to those items observed in the linear stochastic processes. So, not only does the advantage of the GA relative to B&H shrink, but its disadvantage relative to the omniscient also becomes larger.

However, nonlinearity does not change many of the results in relation to the luck coefficients. The luck-coefficient ratios are all higher than 1, and most of the results are statistically significant, indicating the relative stability of the GA.

Table 7. Performance Statistics of the OGA and B&H – Bilinear.

Code	Model	$\bar{\pi}^1$	$\bar{\pi}^2$	$\bar{\pi}^*$	z_π	$\tilde{\pi}$ (%)	$\dot{\pi}$ (%)	$\hat{p}_w$	z_w
BL-1	BL(0,0,1,1)	1.253	1.126	4.398	−6.78	−4	−50	0.491	–0.57
BL-2	BL(0,0,1,1)	1.151	1.064	4.228	−4.66	−3	−58	0.517	1.08
BL-3	BL(0,1,1,2)	1.302	1.830	5.341	11.50	13	175	0.861	17.78
BL-4	BL(0,1,1,2)	1.186	1.356	4.449	6.95	5	91	0.745	17.78
BL-5	BL(1,0,2,1)	1.260	1.419	4.539	5.07	5	61	0.747	17.97
BL-6	BL(1,0,2,1)	2.292	3.143	7.226	9.89	17	66	0.877	36.30
BL-7	BL(1,1,2,2)	1.841	2.471	6.448	8.83	14	75	0.848	30.65
BL-8	BL(1,1,2,2)	1.602	2.287	5.894	19.57	16	114	0.870	34.79
Code	Model	$\hat{s}_1$	$\hat{s}_2$	$\hat{d}$	z_d	$\hat{l}^1_{0.05}$	$\hat{l}^2_{0.05}$	$\hat{r}_{0.05}$	z_r
BL-1	BL(0,0,1,1)	0.316	0.251	–0.065	–3.29	0.132	0.105	1.256	3.30
BL-2	BL(0,0,1,1)	0.190	0.144	–0.046	–2.21	0.144	0.101	1.427	4.14
BL-3	BL(0,1,1,2)	0.167	0.425	0.259	7.31	0.182	0.124	1.793	3.08
BL-4	BL(0,1,1,2)	0.162	0.724	0.562	16.32	0.232	0.129	1.465	3.22
BL-5	BL(1,0,2,1)	0.178	0.465	0.287	13.53	0.211	0.138	1.531	3.54
BL-6	BL(1,0,2,1)	0.251	0.539	0.289	10.38	0.346	0.226	1.534	2.05
BL-7	BL(1,1,2,2)	0.285	0.711	0.426	9.29	0.270	0.168	1.603	2.67
BL-8	BL(1,1,2,2)	0.179	0.386	0.207	2.52	0.272	0.182	1.494	1.14

Note: $\bar{\pi}^1$, $\bar{\pi}^2$ and $\bar{\pi}^*$ are the respective sample mean return of OGA, B&H and the omniscient trader. $\tilde{\pi}$ is the exploitation ratio (Eq. (26)), and $\dot{\pi}$ is the relative superiority index (Eq. (27)). $\hat{p}_w$ is the sample winning probability of OGA over B&H (Eq. (28)). $\hat{s}_1$ and $\hat{s}_2$ are the corresponding sample Sharpe ratio of OGA and B&H (Eq. (31)). Their sample difference is $\hat{d}$ (Eq. (32)). $\hat{l}^1_{0.05}$ and $\hat{l}^2_{0.05}$ are the sample luck coefficient of OGA and B&H (Eq. (39)), and $\hat{r}_{0.05}$ is the sample luck coefficient ratio between the two (Eq. (40)). The z_π, z_w, z_d and z_r are the test statistics of the mean return difference, winning probability, Sharpe ratio differential, and luck coefficient ratio, respectively. The critical value of them is 1.28 at the 10% significance level, and is 1.64 at the 5% significance level.

6.3. ARCH and GARCH Processes

As we have already seen from the bilinear processes, nonlinearity can have some adverse effects on the performance of the GA. It would be imperative to know whether this finding is just restricted to a specific class of nonlinear processes or can be generalized to other nonlinear processes. In this and the next two sections, we shall focus on this question, and briefly mention other details when we see the necessity.

Let us first take a look at the results of the other two nonlinear stochastic processes, namely, ARCH and GARCH. Just like what we saw in the bilinear processes, these two classes of processes can become pure nonlinear stochastic if some specific coefficient values are set to zero. This is basically what we do

Table 8. Performance Statistics of the OGA and B&H – ARCH.

Code	Model	$\bar{\pi}^1$	$\bar{\pi}^2$	$\bar{\pi}^*$	z_π	$\tilde{\pi}$ (%)	$\dot{\pi}$ (%)	$\hat{p}_w$	z_w
AH-1	AR(0)-ARCH(1)	1.038	1.013	3.195	−1.99	−1	−66	0.546	2.92
AH-2	AR(0)-ARCH(1)	1.001	1.005	4.251	0.19	0	400	0.592	5.92
AH-3	AR(0)-ARCH(2)	0.985	0.991	2.307	0.67	0	40	0.562	3.95
AH-4	AR(0)-ARCH(2)	1.007	0.997	2.268	−1.09	−1	−143	0.529	1.84
AH-5	AR(1)-ARCH(1)	1.175	1.509	2.187	22.88	33	191	0.862	33.19
AH-6	AR(1)-ARCH(2)	1.300	1.705	3.061	17.64	23	135	0.838	29.01
AH-7	MA(1)-ARCH(1)	0.869	1.551	3.602	44.12	25	521	0.959	73.20
Code	Model	$\hat{s}_1$	$\hat{s}_2$	$\hat{d}$	z_d	$\hat{l}^1_{0.05}$	$\hat{l}^2_{0.05}$	$\hat{r}_{0.05}$	z_r
AH-1	AR(0)-ARCH(1)	0.170	0.038	–0.032	−1.33	0.117	0.091	1.285	4.53
AH-2	AR(0)-ARCH(1)	0.001	0.010	0.009	0.34	0.149	0.105	1.411	3.19
AH-3	AR(0)-ARCH(2)	−0.038	−0.035	0.002	0.09	0.100	0.079	1.269	4.03
AH-4	AR(0)-ARCH(2)	0.017	−0.012	−0.030	−1.22	0.099	0.080	1.246	3.24
AH-5	AR(1)-ARCH(1)	0.211	0.774	0.563	15.42	0.145	0.109	1.331	3.43
AH-6	AR(1)-ARCH(2)	0.221	0.605	0.384	10.79	0.187	0.140	1.332	2.15
AH-7	MA(1)-ARCH(1)	−0.641	1.126	1.766	35.75	0.076	0.086	0.889	−3.44

Note: $\bar{\pi}^1$, $\bar{\pi}^2$ and $\bar{\pi}^*$ are the respective sample mean return of OGA, B&H and the omniscient trader. $\tilde{\pi}$ is the exploitation ratio (Eq. (26)), and $\dot{\pi}$ is the relative superiority index (Eq. 27). $\hat{p}_w$ is the sample winning probability of OGA over B&H (Eq. (28)). $\hat{s}_1$ and $\hat{s}_2$ are the corresponding sample Sharpe ratio of OGA and B&H (Eq. (31)). Their sample difference is $\hat{d}$ (Eq. (32)). $\hat{l}^1_{0.05}$ and $\hat{l}^2_{0.05}$ are the sample luck coefficient of OGA and B&H (Eq. (39)), and $\hat{r}_{0.05}$ is the sample luck coefficient ratio between the two (Eq. (40)). The z_π, z_w, z_d and z_r are the test statistics of the mean return difference, winning probability, Sharpe ratio differential, and luck coefficient ratio, respectively. The critical value of them is 1.28 at the 10% significance level, and is 1.64 at the 5% significance level.

in Tables 3 and 4. Notice that, based on these settings, AH-1 to AH-4 (ARCH) and GH-1 to GH-6 (GARCH) are all pure nonlinear stochastic processes, i.e. pure ARCH or pure GARCH without linear ARMA components. For the rest, they are a mixture of pure ARCH (GARCH) and linear ARMA processes. Tables 8 and 9 summarize the results of the two stochastic processes. A striking feature is that, in contrast to its performance in mixed processes, the GA performed dramatically worse in pure nonlinear ARCH and GARCH scenarios.

Let us take the ARCH processes as an illustration. In the mixed processes AH-5, AH-6 and AH-7, the GA has a probability of up to 80% or higher of beating B&H, and earned 135–521% more than B&H. The fact that these excess returns are not compensation for risk is further confirmed by the Sharpe-ratio differentials which are significantly positive. In addition, the GA exploited 23–33% of the potential returns earned by the omniscient trader. However, when coming to the pure

Table 9. Performance Statistics of the OGA and B&H – GARCH.

Code	Model	$\bar{\pi}^1$	$\bar{\pi}^2$	$\bar{\pi}^*$	z_π	$\tilde{\pi}$ (%)	$\dot{\pi}$ (%)	$\hat{p}_w$	z_w
GH-1	AR(0)-GARCH(1,1)	0.987	0.983	2.457	−0.42	0	−31	0.539	2.47
GH-2	AR(0)-GARCH(1,1)	0.968	0.979	2.580	1.19	1	34	0.554	3.44
GH-3	AR(0)-GARCH(1,2)	1.008	1.007	2.474	−0.04	0	−13	0.544	2.79
GH-4	AR(0)-GARCH(1,2)	0.998	1.007	2.434	0.90	1	450	0.572	4.60
GH-5	AR(0)-GARCH(2,1)	0.978	1.001	2.637	2.24	1	105	0.584	5.39
GH-6	AR(0)-GARCH(2,1)	0.982	0.997	2.595	1.50	1	83	0.563	4.02
GH-7	AR(1)-GARCH(1,1)	1.428	1.926	3.511	18.40	24	116	0.856	32.07
GH-8	AR(1)-GARCH(1,2)	1.356	1.747	3.298	12.58	20	110	0.841	29.49
GH-9	AR(1)-GARCH(2,1)	1.378	1.934	3.616	19.20	25	147	0.872	35.21
GH-10	MA(1)-GARCH(1,1)	0.911	1.376	2.769	36.44	25	521	0.949	64.54
Code	Model	$\hat{s}_1$	$\hat{s}_2$	$\hat{d}$	z_d	$\hat{l}^1_{0.05}$	$\hat{l}^2_{0.05}$	$\hat{r}_{0.05}$	z_r
GH-1	AR(0)-GARCH(1,1)	−0.030	−0.652	−0.035	−1.19	0.101	0.079	1.282	4.30
GH-2	AR(0)-GARCH(1,1)	−0.080	−0.076	0.004	0.17	0.098	0.081	1.202	4.08
GH-3	AR(0)-GARCH(1,2)	−0.005	0.020	0.024	1.05	0.094	0.081	1.166	3.32
GH-4	AR(0)-GARCH(1,2)	0.020	0.026	0.007	0.27	0.108	0.093	1.151	1.68
GH-5	AR(0)-GARCH(2,1)	−0.051	0.005	0.056	2.04	0.103	0.083	1.233	4.10
GH-6	AR(0)-GARCH(2,1)	−0.044	−0.012	0.032	1.23	0.097	0.083	1.178	3.50
GH-7	AR(1)-GARCH(1,1)	0.244	0.620	0.375	11.06	0.225	0.158	1.426	2.72
GH-8	AR(1)-GARCH(1,2)	0.231	0.614	0.383	14.52	0.201	0.143	1.405	2.59
GH-9	AR(1)-GARCH(2,1)	0.703	0.239	0.465	13.47	0.213	0.147	1.454	3.13
GH-10	MA(1)-GARCH(1,1)	−0.476	1.034	1.509	29.43	0.070	0.081	0.867	−3.90

Note: $\bar{\pi}^1$, $\bar{\pi}^2$ and $\bar{\pi}^*$ are the respective sample mean return of OGA, B&H and the omniscient trader. $\tilde{\pi}$ is the exploitation ratio (Eq. (26)), and $\dot{\pi}$ is the relative superiority index (Eq. (27)). $\hat{p}_w$ is the sample winning probability of OGA over B&H (Eq. (28)). $\hat{s}_1$ and $\hat{s}_2$ are the corresponding sample Sharpe ratio of OGA and B&H (Eq. (31)). Their sample difference is $\hat{d}$ (Eq. (32)). $\hat{l}^1_{0.05}$ and $\hat{l}^2_{0.05}$ are the sample luck coefficient of OGA and B&H (Eq. (39)), and $\hat{r}_{0.05}$ is the sample luck coefficient ratio between the two (Eq. (40)). The z_π, z_w, z_d and z_r are the test statistics of the mean return difference, winning probability, Sharpe ratio differential, and luck coefficient ratio, respectively. The critical value of them is 1.28 at the 10% significance level, and is 1.64 at the 5% significance level.

nonlinear processes AH-1 to AH-4, this dominance either disappears or becomes weaker. This can be easily shown by the sharp decline in the statistics z_π, z_w and z_d in Table 8 with an almost 0% exploitation ($\tilde{\pi}$) of the maximum potential returns.

This discernible pattern also extends to Table 9. The double-digit z_π, z_w, and z_d of the mixed processes (GH-7 to GH-10) distinguish themselves from the low, or even negative, single-digit ones of the pure nonlinear processes (GH-1 to GH-6). For the former, the GA has 84–95% chance of beating B&H and earned 110–521% more than B&H. Again, from z_d, we know that the high returns are more than compensation for risk. Very similar to the case of ARCH, 20–25% of the maximum potential returns can be exploited by the GA, but that value $\tilde{\pi}$ drops near to 0% when the underlying processes change to pure GARCH.

Despite the fact that pure nonlinear processes continue to deal the GA a hard blow, as far as the winning probability is concerned, its relative performance to B&H is overwhelmingly good. This can be reflected by the z_w statistics which are consistently significantly positive in all cases. A similar property holds for the luck coefficient (see z_r). The only two exceptions are the cases AH-7 and GH-10, which, however, are not pure nonlinear. In fact, they both have MA(1) as their linear component.

6.4. Threshold Processes

The threshold process leads to a different kind of nonlinear process. While its global behavior is nonlinear, within each local territory, characterized by Ω_i, it can be linear. TH-1 and TH-2 in Table 5 are exactly processes of this kind. The former is switching between two AR(1) processes, whereas the latter is switching between two ARMA(1,1) processes. Since the GA can work well with linear processes, it would be interesting to know whether its effectiveness will extend to these local linear processes. Our results are shown in Table 10. The four statistics z_π, z_w, z_d, and z_r all give positive results. The GA is seen to exploit 20–30% of the maximum potential returns, and the winning probabilities are greater than 90%.

TH-4 and TH-5 are another kind of complication. TH-4 switches between two mixed processes, while TH-5 switches between a pure nonlinear process and a linear process. From previous experiences, we already knew that the GA can work well with the mixed process. Now, from Table 10, it seems clear that it can survive these two complications as well.

Finally, we come to the most difficult one TH-5, i.e the one which switches between two pure nonlinear (bilinear) processes. Since the GA did not show its competence in the pure nonlinear process, at least from the perspective of the return criteria, one may conjecture that TH-5 will deal another hard blow to the

Table 10. Performance Statistics of the OGA and B&H – Threshold.

Code	Model	$\bar{\pi}^1$	$\bar{\pi}^2$	$\bar{\pi}^*$	z_π	$\tilde{\pi}$ (%)	$\dot{\pi}$ (%)	$\hat{p}_w$	z_w
TH-1	TBL(2;1,0,0,0)	0.612	1.233	3.372	24.89	23	160	0.910	45.30
TH-2	TBL(2;1,1,0,0)	1.262	2.743	6.361	21.15	29	565	0.931	53.77
TH-3	TBL(2;0,0,1,1)	1.161	1.074	4.207	–4.38	–3	–54	0.502	0.13
TH-4	TBL(2;1,1,2,2)	1.271	1.406	4.497	5.41	4	50	0.717	15.23
TH-5	TBL(2;2,0,2,2)	0.654	1.236	3.890	37.38	18	168	0.919	48.56
Code	Model	$\hat{s}_1$	$\hat{s}_2$	$\hat{d}$	z_d	$\hat{l}^1_{0.05}$	$\hat{l}^2_{0.05}$	$\hat{r}_{0.05}$	z_r
TH-1	TBL(2;1,0,0,0)	–0.398	0.374	0.772	9.33	0.267	0.119	2.252	4.30
TH-2	TBL(2;1,1,0,0)	0.093	0.727	0.634	11.86	0.329	0.163	2.012	2.95
TH-3	TBL(2;0,0,1,1)	0.208	0.176	–0.032	–1.42	0.136	0.098	1.394	3.72
TH-4	TBL(2;1,1,2,2)	0.208	0.426	0.219	10.41	0.192	0.140	1.379	2.97
TH-5	TBL(2;2,0,2,2)	–0.813	0.484	1.297	16.88	0.130	0.097	1.343	3.54

Note: $\bar{\pi}^1$, $\bar{\pi}^2$ and $\bar{\pi}^*$ are the respective sample mean return of OGA, B&H and the omniscient trader. $\tilde{\pi}$ is the exploitation ratio (Eq. (26)), and $\dot{\pi}$ is the relative superiority index (Eq. (27)). $\hat{p}_w$ is the sample winning probability of OGA over B&H (Eq. (28)). $\hat{s}_1$ and $\hat{s}_2$ are the corresponding sample Sharpe ratio of OGA and B&H (Eq. (31)). Their sample difference is $\hat{d}$ (Eq. (32)). $\hat{l}^1_{0.05}$ and $\hat{l}^2_{0.05}$ are the sample luck coefficient of OGA and B&H (Eq. (39)), and $\hat{r}_{0.05}$ is the sample luck coefficient ratio between the two (Eq. (40)). The z_π, z_w, z_d and z_r are the test statistics of the mean return difference, winning probability, Sharpe ratio differential, and luck coefficient ratio, respectively. The critical value of them is 1.28 at the 10% significance level, and is 1.64 at the 5% significance level.

GA. Both z_π and z_d in Table 10 confirm this conjecture. Not just the returns, but z_w shows that the winning probability is also not good, which is similar to what we experienced in BL-1 and BL-2. The only criterion that remains unaffected by this complication is the luck coefficient. Furthermore, it turns out that z_r seems to give the most stable performance across all kinds of processes considered so far, except the MA process.

6.5. Chaotic Processes

Chaotic processes are also nonlinear, but they differ from the previous four nonlinear processes in that they are *deterministic* rather than *stochastic*. These processes can behave quite erratically without any discernible pattern. Can the GA survive well with this type of nonlinear process? The answer is a resounding *yes*. All the statistics in Table 11 are sending us this message.

The winning probabilities are all higher than 85%. In the case of the Henon map (C-2), the GA even beats B&H in all of the 1000 trials. In addition, in this

Table 11. Performance Statistics of the OGA and B&H – Chaos.

Code	$\bar{\pi}^1$	$\bar{\pi}^2$	$\bar{\pi}^*$	z_π	$\tilde{\pi}$ (%)	$\dot{\pi}$ (%)	$\hat{p}_w$	z_w
C-1	1.019	5.664	21.876	31.15	22	24447	0.993	186.99
C-2	5.387	23.235	33.452	85.62	64	407	1.000	*
C-3	0.937	4.124	11.374	44.65	31	5059	0.990	352.49
C-4	1.188	3.066	25.563	22.91	8	999	0.950	65.29
C-5	0.928	1.790	23.172	17.18	4	1197	0.876	36.08
Code	$\hat{s}_1$	$\hat{s}_2$	$\hat{d}$	z_d	$\hat{l}^1_{0.05}$	$\hat{l}^2_{0.05}$	$\hat{r}_{0.05}$	z_r
C-1	0.009	0.832	0.824	16.59	0.297	0.184	1.615	2.28
C-2	1.600	2.502	0.901	23.56	0.112	0.090	1.252	4.39
C-3	−0.075	1.160	1.235	28.92	0.153	0.127	1.207	2.75
C-4	0.074	0.627	0.554	10.39	0.348	0.200	1.739	2.66
C-5	−0.045	0.518	0.563	14.45	0.279	0.169	1.649	2.88

Note: $\bar{\pi}^1$, $\bar{\pi}^2$ and $\bar{\pi}^*$ are the respective sample mean return of OGA, B&H and the omniscient trader. $\tilde{\pi}$ is the exploitation ratio (Eq. (26)), and $\dot{\pi}$ is the relative superiority index (Eq. (27)). $\hat{p}_w$ is the sample winning probability of OGA over B&H (Eq. (28)). $\hat{s}_1$ and $\hat{s}_2$ are the corresponding sample Sharpe ratio of OGA and B&H (Eq. (31)). Their sample difference is $\hat{d}$ (Eq. (32)). $\hat{l}^1_{0.05}$ and $\hat{l}^2_{0.05}$ are the sample luck coefficient of OGA and B&H (Eq. (39)), and $\hat{r}_{0.05}$ is the sample luck coefficient ratio between the two (Eq. (40)). The z_π, z_w, z_d and z_r are the test statistics of the mean return difference, winning probability, Sharpe ratio differential, and luck coefficient ratio, respectively. The critical value of them is 1.28 at the 10% significance level, and is 1.64 at the 5% significance level.

map, the GA is seen to exploited 64% of the potential excess returns earned by the omniscient trader, which is the highest of all the processes tested in this paper. One of the possible reasons why the GA can work well with these nonlinear deterministic processes is that they are not pure nonlinear. C-1, C-2 and C-4 have linear AR(1) or AR(2) components. C-3, like the threshold processes, switches between two linear processes. As already evidenced in Section 6.4, the GA can handle these types of processes effectively. So, the success is not totally unanticipated.

However, the explanation above does not apply to C-5, which has no linear component. Nonetheless, statistics such as z_π, $\tilde{\pi}$ and $\hat{p}_w$ all indicate that this process is not as easy as the other four. For example, only 4% of the potential excess returns are exploited in this process. Regardless of these weaknesses, the fact that the GA can dominate B&H in this case motivates us to ask the following question: *Can the GA work better for the pure nonlinear deterministic processes than the respective stochastic ones, and hence can it help distinguish the chaotic processes from the stochastic processes*? This is a question to pursue in the future.

6.6. Summary

The Monte Carlo simulation analysis conducted above provides us with an underpinning of the practical financial applications of the GA. It pinpoints the kinds of stochastic processes which we may like to see fruitful results. We have found that the GA can perform well with all kinds of stochastic processes which have a linear process (signal) as a part of them. Preliminary studies also suggest that it may also work well with chaotic processes. However, the class of nonlinear stochastic processes presents a severe limitation for the GA. In the next section, we shall see the empirical relevance of these results by actually applying OGA-based trading strategies to financial data.

7. EMPIRICAL ANALYSIS

7.1. Data Description and Analysis

The empirical counterpart of this paper is based on two sets of high-frequency time series data regarding foreign exchange rates, namely, the Euro dollar vs. the U.S. dollar *EUR/USD* and the U.S. dollar vs. the Japanese yen *USD/JPY*.[6] The data is from January 11, 1999 to April 17, 1999. Data within this period are further divided into 12 sub-periods with roughly equal numbers of observations. Table 12 gives the details.

Let $P^U_{i,t}(P^P_{i,t})$ denote the tth ($t = 1, 2, \ldots, n_i$) observation of the ith sub-period ($i = A, B, \ldots, L$) of the EUR/USD (USD/JPY) forex series. The price series is transformed into the return series by the usual logarithmic formulation,

$$r^j_{i,t} = \ln(P^j_{i,t}) - \ln(P^j_{i,t-1}) \tag{47}$$

where $j = U, P$. Tables 13 and 14 give some basic statistics of the returns of each sub-period.

Both return series share some common features. From Tables 13 and 14, the mean, median and skewness of these two return series are all close to zero. The kurtosis is much higher than 3, featuring the well-known *fat-tail* property. The Jarque-Bera (1980) test further confirms that these forex returns do not follow the normal distribution, and that is true for each sub-period. In addition, the series is not independent due to its significant negative first-order serial correlation ρ_1. However, there is no evidence of serial correlation in higher orders.[7]

To apply what we learned from the Monte Carlo simulation to predict the effectiveness of the GA over these series, we must first gauge their likely stochastic processes. Here we follow a standard procedure frequently used in econometrics

Table 12. Data Quotations – EUR/USD and USD/JPY.

Sub-Period	A	B	C	D	E	F
EUR/USD						
Number	12000	12000	12000	12000	12000	12000
From (GMT)	2/25 7:59	3/1 0:59	3/3 15:36	3/8 6:43	3/10 6:53	3/12 7:26
To (GMT)	2/26 8:22	3/2 7:17	3/5 3:04	3/9 1:08	3/11 7:12	3/15 1:16
Sub-Period	G	H	I	J	K	L
Number	12000	12000	12000	12000	12000	12000
From (GMT)	3/17 7:36	3/19 0:19	3/24 15:06	3/26 15:46	3/31 7:32	4/15 6:14
To (GMT)	3/18 6:12	3/22 2:01	3/26 2:12	3/30 6:23	4/02 1:14	4/17 0:37
Sub-Period	A	B	C	D	E	F
USD/JPY						
Number	12000	12000	12000	12000	12000	10808
From (GMT)	1/11 6:11	1/15 0:00	1/27 15:14	2/04 8:47	2/17 7:20	2/23 6:10
To (GMT)	1/14 8:11	1/21 0:00	2/03 3:24	2/11 2:43	2/23 6:09	2/26 21:48
Sub-Period	G	H	I	J	K	L
Number	12000	12000	11026	12000	12000	12000
From (GMT)	2/28 18:15	3/04 10:02	3/09 21:52	3/15 5:25	3/18 6:07	3/24 13:00
To (GMT)	3/04 10:01	3/09 21:52	3/15 1:21	3/18 6:06	3/24 13:00	3/30 10:41

Note: GMT: Greenwich Mean Time.

(Chen & Lu, 1999). First, notice that all series used in our Monte Carlo simulation are *stationary*. To make sure that the forex returns are stationary, the *Augmented Dickey-Fuller* (*ADF*) test is applied (Dickey & Fuller, 1979). From Table 15, the null hypothesis that $r^j_{i,t}$ contains a unit root is rejected at the 1% significance level, meaning that the $r^j_{i,t}$ are stationary.

Second, since our Monte Carlo simulations demonstrate the effectiveness of the GA over the linear stochastic processes, it is important to know whether the forex returns have a linear component. To do so, the famous Rissanen's predictive stochastic complexity (*PSC*) as a linear filter is taken.[8] Table 15 gives the ARMA(p, q) process extracted from the forex return series. A MA(1) linear process is founded for both forex returns in each sub-period. In fact, it re-confirms the early finding that the high-frequency forex returns follow a MA(1) process (Moody & Wu, 1997; Zhou, 1996).

Third, it should be not surprising if none of these series is just linear. To see whether nonlinear dependence exists, one of the most frequently used statistics, the *BDS* test, is applied to the residuals filtered through the PSC filter.[9] There

Table 13. Basic Statistics of the Return Series – EUR/USD.

Sub-Period	A	B	C	D	E	F
Mean	−2.56E−07	−8.13E−07	−7.37E−07	5.39E−07	5.63E−07	−7.49E−07
Median	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
Std. Dev.	0.000252	0.000252	0.000213	0.000191	0.000238	0.000264
Skewness	−0.015831	0.007214	−0.034436	0.002017	−0.001071	−0.009908
Kurtosis	5.606484	5.558600	5.636056	5.976148	6.136196	5.757020
Jarque-Bera	3397.10	3273.05	3476.48	4428.37	4917.45	3800.46
P-value	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
ρ_1	−0.513935	−0.503725	−0.494695	−0.504014	−0.486925	−0.509612
Sub-Period	G	H	I	J	K	L
Mean	3.81E−07	−8.00E−07	−7.48E−07	−5.64E−08	2.37E−07	−1.13E−06
Median	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
Std. Dev.	0.000225	0.000217	0.000184	0.000241	0.000292	0.000219
Skewness	0.011155	−0.050369	−0.119412	0.007646	−0.021431	−0.203838
Kurtosis	6.512019	5.435495	6.226714	5.337107	8.780986	10.97326
Jarque-Bera	6166.88	2970.40	5233.92	2730.92	16708.03	31861.55
P-value	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
ρ_1	−0.493223	−0.505528	−0.480500	−0.498232	−0.475452	−0.464571

Note: ρ_1 is the first-order autocorrelation coefficient. Jarque-Bera statistic converges to a chi-square distribution with two degrees of freedom under the normality assumption.

Table 14. Basic Statistics of the Return Series – USD/JPY.

Sub-Period	A	B	C	D	E	F
Mean	3.97E−07	−5.16E−07	−2.01E−06	2.54E−07	1.69E−06	−1.44E−06
Median	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
Std. Dev.	0.000413	0.002108	0.001853	0.000332	0.000311	0.000363
Skewness	0.008135	0.080038	−0.018340	−0.057694	0.022959	−0.003358
Kurtosis	6.769064	6.711594	6.854310	7.170642	6.757800	6.374525
Jarque-Bera	7091.806	6898.478	7426.049	8700.883	7059.230	5123.885
P-value	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
ρ_1	−0.343317	−0.338790	−0.370748	−0.362052	−0.360786	−0.335953
Sub-Period	G	H	I	J	K	L
Mean	2.53E−06	−1.09E−06	−2.54E−06	−2.75E−07	−7.87E−07	1.90E−06
Median	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
Std. Dev.	0.000301	0.000279	0.000322	0.000287	0.000265	0.000247
Skewness	0.080100	0.019734	0.079313	0.002414	−0.019244	0.213584
Kurtosis	5.597214	6.763973	6.747828	8.198238	7.650768	6.701801
Jarque-Bera	3385.029	7083.936	6459.934	13508.60	10811.96	6941.746
P-value	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
ρ_1	−0.436860	−0.396329	−0.344660	−0.348622	−0.361993	−0.364189

Note: ρ_1 is the first-order autocorrelation coefficient. Jarque-Bera statistic converges to a chi-square distribution with two degrees of freedom under the normality assumption.

Table 15. Basic Econometric Properties of the Return Series – EUR/USD and USD/JPY.

Sub-Period	A	B	C	D	E	F
EUR/USD						
ADF	−74.9502	−76.4264	−74.0755	−76.6226	−77.4292	−79.1714
Critical Value	−3.4341	−3.4341	−3.4341	−3.4341	−3.4341	−3.4341
PSC	(0,1)	(0,1)	(0,1)	(0,1)	(0,1)	(0,1)
	G	H	I	J	K	L
ADF	−74.7427	−74.7053	−68.8254	−73.4958	−72.3726	−67.6148
Critical Value	−3.4341	−3.4341	−3.4341	−3.4341	−3.4341	−3.4341
PSC	(0,1)	(0,1)	(0,1)	(0,1)	(0,1)	(0.1)
	A	B	C	D	E	F
USD/JPY						
ADF	−57.1573	−55.2394	−56.0518	−56.8433	−55.0202	−51.1507
Critical Value	−2.5660	−2.5660	−2.5660	−3.4341	−3.4341	−3.4342
PSC	(0,1)	(0,1)	(0,1)	(0,1)	(0,1)	(0,1)
	G	H	I	J	K	L
ADF	−59.3422	−57.4123	−55.5809	−58.0822	−57.5485	−59.5623
Critical Value	−3.4341	−3.4341	−3.4341	−3.4341	−3.4341	−3.4341
PSC	(0,1)	(0,1)	(0,1)	(0,1)	(0,1)	(0,1)

Note: The "Critical Value" indicates the critical value of the ADF test that is taken from the table provided by Dickey and Fuller at the 1% significance level.

are two parameters used to conduct the BDS test. One is the distance measure (ε standard deviations), and the other is the embedding dimension. The parameter "ε" considered here is equal to one standard deviation. (In fact, other are also tried, but the results are not sensitive to the choice of ε.) The embedding dimensions considered range from 2 to 5. Following Barnett et al. (1997), if the absolute values of all BDS statistics under various embedding dimensions are greater than 1.96, the null hypothesis of an identical independent distribution (IID) is rejected. From Table 16, the BDS statistics for the EUR/USD and USD/JPY are all large enough to reject the null hypothesis, i.e. nonlinear dependence is detected.

Fourth, given the existence of the nonlinear dependence, the next step is to identify its possible form, i.e. by modeling nonlinearity. While there is no standard answer as to how this can be done, the voluminous (G)ARCH literature over the past two decades has proposed a second-moment connection (Bollerslev et al., 1992). In order to see whether (G)ARCH can successfully capture nonlinear signals, we

Table 16. The BDS Test of the PSC-filtered Return Series – EUR/USD and USD/JPY.

Sub-Period	A		B		C		D		E		F	
Part	I	II	I	II	I	II	I	II	I	II	I	II
EUR/USD												
DIM = 2	20.47	26.82	22.58	26.56	13.60	20.25	17.15	14.66	18.23	18.09	18.03	19.37
DIM = 3	27.57	34.17	30.61	34.72	19.44	26.84	22.50	20.12	22.78	23.48	24.63	26.43
DIM = 4	33.60	40.03	37.25	40.81	23.80	31.27	26.80	24.22	25.68	27.63	30.21	32.09
DIM = 5	38.50	45.80	43.40	46.75	27.43	35.23	30.38	27.40	28.54	31.23	35.26	37.94
	G		H		I		J		K		L	
	I	II	I	II	I	II	I	II	I	II	I	II
DIM = 2	12.04	16.97	23.90	19.45	13.06	12.40	20.13	13.41	35.69	19.74	8.18	22.23
DIM = 3	17.84	22.20	30.02	25.59	17.30	17.31	26.84	18.79	46.83	24.39	10.98	27.08
DIM = 4	21.09	26.34	34.39	30.41	20.35	20.57	31.24	22.98	56.42	27.22	12.97	30.22
DIM = 5	24.08	30.18	39.31	35.47	23.29	23.40	35.39	26.48	66.58	29.79	14.20	33.13
	A		B		C		D		E		F	
	I	II	I	II	I	II	I	II	I	II	I	II
USD/JPY												
DIM = 2	15.36	23.15	15.68	13.41	12.00	16.63	14.76	20.44	12.98	17.84	17.88	16.61
DIM = 3	17.89	28.38	18.83	16.04	14.54	20.02	17.11	23.15	16.08	20.87	21.35	18.94
DIM = 4	20.03	31.37	20.17	17.89	15.32	22.24	18.72	24.27	17.49	22.82	23.35	20.44
DIM = 5	22.30	34.58	21.57	19.13	16.07	24.42	20.28	25.43	18.52	24.56	24.43	22.16
	G		H		I		J		K		L	
	I	II	I	II	I	II	I	II	I	II	I	II
DIM = 2	15.65	11.34	15.56	16.84	16.44	15.51	20.98	17.79	19.41	15.51	15.28	15.61
DIM = 3	17.64	13.92	18.57	18.91	18.50	18.68	25.07	21.84	21.94	16.84	16.32	17.87
DIM = 4	19.30	15.35	20.86	19.45	19.78	21.02	27.72	24.43	23.23	17.52	17.21	19.34
DIM = 5	20.82	16.49	23.10	19.73	20.95	22.76	30.10	26.45	24.15	18.56	18.14	20.62

Note: Due to the size of the data which is beyond the affordable limit of the software computing the BDS statistics, each sub-period was divided into two parts before the BDS test was applied. The BDS statistic follows an asymptotically standard normal distribution.

carry out the *Lagrange Multiplier* (*LM*) test for the presence of ARCH effects. The LM test for ARCH effects is a test based on the following model:

$$\sigma_t^2 = h(\alpha_0 + \alpha_1 \varepsilon_{t-1}^2 + \cdots + \alpha_p \varepsilon_{t-p}^2), \tag{48}$$

where h is a differential function. The null hypothesis that the ARCH effect does not exist is

$$\alpha_1 = \cdots = \alpha_p = 0. \tag{49}$$

By taking $p = 1, 2, \ldots, 4$, the LM test results are given in Table 17. It is found that the ARCH effect does exist in both return series.

Table 17. The LM Test of the ARCH Effect in the Return Series – EUR/USD and USD/JPY.

Sub-Period	A	B	C	D	E	F
EUR/USD						
$p = 1$	1029.94	821.665	681.92	560.27	463.98	401.08
$p = 2$	1572.34	1191.26	998.22	1094.72	960.83	585.88
$p = 3$	2030.32	1501.74	1202.15	1320.58	1052.54	705.17
$p = 4$	2169.98	1731.33	1295.77	1471.40	1195.93	871.73
Sub-Period	G	H	I	J	K	L
$p = 1$	275.07	797.26	411.61	390.94	1584.30	1571.04
$p = 2$	423.33	1168.19	689.02	553.11	1668.88	1587.53
$p = 3$	493.11	1262.87	1001.22	678.90	1714.39	1640.60
$p = 4$	551.99	1354.28	1050.53	715.68	2036.42	1641.41
Sub-Period	A	B	C	D	E	F
USD/JPY						
$p = 1$	533.15	411.35	479.80	769.49	550.15	685.34
$p = 2$	639.75	490.58	6018.02	849.31	604.18	752.71
$p = 3$	677.49	531.78	667.50	854.11	614.26	821.85
$p = 4$	709.00	559.97	687.09	923.01	636.99	854.71
Sub-Period	G	H	I	J	K	L
$p = 1$	600.528	545.791	696.185	749.650	883.107	795.762
$p = 2$	648.101	656.653	758.918	1094.82	926.127	929.618
$p = 3$	695.639	727.043	811.000	1101.78	939.221	1059.00
$p = 4$	726.942	764.836	844.766	1103.08	951.489	1109.23

Note: The LM test is asymptotically distributed as χ^2 with p degrees of freedom when the null hypothesis is true. There is no need to report the p values here because they are all 0.0000.

After these series of statistical tests, we may conclude that basically both *the EUR/USD and the USD/JPY return series have MA(1) as a linear component and ARCH as a part of its nonlinear components*. In Section 6.3, the Monte Carlo simulation analysis already indicated that the GA can work well with MA(1) plus (G)ARCH processes. To see the empirical relevance of the simulation study, in the next sections, the GA is applied to the two return series.

7.2. Experimental Design

In order to compare the empirical results with our earlier simulation analysis, the experiments are designed in a similar fashion to the one which our Monte Carlo

simulation follows. Specifically, many "*ensembles*" are generated from the original series to evaluate the performance of the GA. Of course, rigorously speaking, they are not the "ensembles" defined in the stochastic process. They are just subseries taken from the original return series. Each subseries has 105 observations. The first 70 observations are treated as the training sample, and the last 35 observations are used as the testing sample.

Nonetheless, to make the tests we developed in Section 4 applicable, we cannot just continuously chop the return series into subseries, because doing so will not make the sampling process independent, and hence will violate the fundamental assumption required for the central limit theorem. One solution to this problem is to leave an interval between any two consecutive subseries so that they are not immediately connected. The purpose in doing this is hopefully to make them independent of each other as if they were sampled independently. However, how large an interval would suffice? To answer this question, we take a subsequence with a fixed number of lags, say, $\{r^j_{i,t}, r^j_{i,t+k}, r^j_{i,t+2k}, \ldots\}$ from the original return series, where k varies from 40, 60, ..., to 300. We then apply the BDS test to each of these subsequences.

Table 18 summarizes the BDS test results. For the EUR/USD case, it is found that when k is greater than 100, the null hypothesis that the subsequence $\{r^j_{i,t}, r^j_{i,t+k}, r^j_{i,t+2k}, \ldots\}$ is IID is not rejected. In other words, leaving an interval of 100 observations between each of two consecutive subseries would suffice. For the EUR/USD case, k can even be smaller than 60. To ensure the quality of the sampling process, we, however, take an even larger number of lags, i.e. $k = 200$. This choice leaves us with a total of 720 subseries from the EUR/USD and 709 subseries from the USD/JPY.

The GA is then employed to extract trading strategies from the training samples of these subseries, and the strategies extracted are further applied to the respective testing samples. The resulting accumulated returns (π) are then compared with that of the B&H strategy.

7.3. Results of the Experiments

Since the analysis of the data shows that the two forex returns are mixtures of MA(1) and (G)ARCH processes, our previous results of Monte Carlo simulations may provide a good reference for what one can expect from such empirical applications. Both Tables 8 and 9 indicate the superior performance of the GA over B&H, except in relation to the criterion for the luck coefficient, when the underlying stochastic processes are MA plus (G)ARCH. Will the dominance carry over?

Table 18. The BDS Test of the Lag Period in the Return Series – EUR/USD and USD/JPY.

Lag	DIM = 2	DIM = 3	DIM = 4	DIM = 5
EUR/USD				
40	2.94	3.45	3.86	4.18
60	0.72	1.20	1.27	1.38
80	1.11	1.21	1.38	1.50
100	0.66	0.66	0.69	0.69
120	0.61	0.66	0.79	0.88
140	0.45	0.52	0.54	0.58
160	0.30	0.43	0.46	0.54
180	0.21	0.30	0.42	0.49
200	−0.01	0.08	0.12	0.11
220	0.11	0.14	0.13	0.13
240	0.25	0.24	0.27	0.24
260	−0.02	−0.04	−0.04	−0.01
280	0.10	0.11	0.14	0.14
300	0.06	0.07	0.05	0.01
USD/JPY				
40	1.39	1.50	1.50	1.57
60	0.53	0.69	0.75	0.89
80	0.56	0.63	0.72	0.80
100	−0.08	−0.12	−0.12	−0.16
120	0.13	0.22	0.19	0.20
140	0.01	−0.13	−0.14	−0.09
160	0.05	0.09	0.09	0.12
180	−0.01	−0.07	0.01	0.06
200	−0.04	−0.08	−0.08	−0.06
220	0.21	0.29	0.30	0.32
240	0.15	0.13	0.11	0.12
260	0.05	0.12	0.09	0.07
280	−0.14	−0.09	−0.11	−0.10
300	0.06	0.02	0.05	0.04

Note: The BDS statistic follows an asymptotically standard normal distribution.

Table 19 is the kind of table which we have presented many times in Section 6. All the key statistics z_π, z_w, and z_d are consistent with those of AH-7 (Table 8) and GH-10 (Table 9). So, in both forex return series, the dominance of the GA over B&H is statistically significant. The consistency continues even to a finer level of the results: $\bar{\pi}^1 < 1$ and $\bar{\pi}^2 > 1$. As already seen, B&H earned negative profits in both of the cases AH-7 and GH-10, while the GA earned positive profits in both cases. In addition, both the winning probability and the exploitation ratio are also

Table 19. Performance Statistics of the OGA and B&H – EUR/USD and USD/JPY.

	$\bar{\pi}^1$	$\bar{\pi}^2$	$\bar{\pi}^*$	z_π	$\tilde{\pi}$ (%)	$\dot{\pi}$ (%)	$\hat{p}_w$	z_w
EUR/USD	0.9999	1.0012	1.0028	38.58	43	9257	0.972	77.10
USD/JPY	0.9999	1.0010	1.0039	23.70	27	11462	0.850	26.17
	$\hat{s}_1$	$\hat{s}_2$	$\hat{d}$	z_d	$\hat{l}^1_{0.05}$	$\hat{l}^2_{0.05}$	$\hat{r}_{0.05}$	z_r
EUR/USD	−0.0338	1.4193	1.4532	18.32	0.0812	0.0933	0.8710	−1.69
USD/JPY	−0.0086	0.8786	0.8873	20.64	0.0826	0.0948	0.8713	−1.66

Note: $\bar{\pi}^1$, $\bar{\pi}^2$ and $\bar{\pi}^*$ are the respective sample mean return of OGA, B&H and the omniscient trader. $\tilde{\pi}$ is the exploitation ratio (Eq. (26)), and $\dot{\pi}$ is the relative superiority index (Eq. (27)). $\hat{p}_w$ is the sample winning probability of OGA over B&H (Eq. (28)). $\hat{s}_1$ and $\hat{s}_2$ are the corresponding sample Sharpe ratio of OGA and B&H (Eq. (31)). Their sample difference is $\hat{d}$ (Eq. (32)). $\hat{l}^1_{0.05}$ and $\hat{l}^2_{0.05}$ are the sample luck coefficient of OGA and B&H (Eq. (39)), and $\hat{r}_{0.05}$ is the sample luck coefficient ratio between the two (Eq. (40)). The z_π, z_w, z_d and z_r are the test statistics of the mean return difference, winning probability, Sharpe ratio differential, and luck coefficient ratio, respectively. The critical value of them is 1.28 at the 10% significance level, and is 1.64 at the 5% significance level.

comparable. $\hat{p}_w$ is around 95% for both AH-7 and GH-10, and $\tilde{\pi}$ is about 25%. The value of $\hat{p}_w$ remains as high for the EUR/USD series, while it drops a little to 85% for the USD/JPY series. As to $\tilde{\pi}$, it is also about 25% for the USD/JPY series, but is greater than 40% for the EUR/USD series.

Notice that our earlier simulation result already indicated that, for some reason unknown to us, the MA component when combined with the ARCH or GARCH component may bring a negative impact to the luck coefficient. This has been already shown in the cases AH-7 and GH-10. What interests us here is that this observation repeats itself in our empirical results. The statistic z_r is statistically negative in both return series. As a result, to a large extent, what we have found from the early Monte Carlo simulations applies quite well to the real data. Hence, the GA can be useful in extracting information to develop trading strategies involving these high-frequency financial data because the underlying stochastic process, based on the Monte Carlo simulation analysis, is not a hard one for the GA.

8. CONCLUDING REMARKS

The literature on financial data mining, driven by the rapid development and applications of computational intelligence tools, are frequently clothed with a "magic house" notoriety. Unlike in mainstream econometrics, users are usually

not well informed of the stochastic properties of these tools, which in turn makes it difficult to grasp the significance of the result obtained from one specific application, be it positive or negative. An essential question is how we can know that what happens in one specific application can or cannot extend to the other one. Will we still be so "lucky" next time?

By using the Monte Carlo simulation methodology, a statistical foundation for using the GA in market-timing strategies is *initiated*. This foundation would allow us to evaluate how likely the GA will work given a time series whose underlying stochastic process is known. This helps us to distinguish the *luck* from *normal expectations*. We believe that this is a major step toward lightening the black box. We emphasize that this work provides *a* statistical foundation, not *the* statistical foundation, because there are many other ways of enriching the current framework and of making it more empirically relevant.

First, different benchmarks may replace the B&H strategy. This is particularly so given a series of articles showing that simple technical analysis can beat B&H. However, since we can never run out of interesting benchmarks, the exploitation ratio $\tilde{\pi}$ introduced in this paper will always be a good reference. For example, in this paper, we can hardly have a $\tilde{\pi}$ of 30% or higher. Consequently, the 70% left there may motivate us to try more advanced version of the GA or different computational intelligence algorithms.

Second, financial time series are not just restricted to the six stochastic processes considered in this paper, but introducing new stochastic processes causes no problems for the current framework. Third, different motivations may define different evaluation criteria. The four criteria used in this paper are by no means exhausted. For example, the downside risk or VaR (Value at Risk) frequently used in current risk management can be another interesting criterion. However, again, it is straightforward to add more criteria to the current framework as long as one is not bothered by deriving the corresponding statistical tests. Fourth, the focus of this paper is to initiate a statistical foundation. Little has been addressed regarding the practical trading behavior or constraints. Things like transaction costs, non-synchronous trading, etc., can be introduced to this framework quite easily. Fifth, our framework is also not restricted to just the ordinary GA, for the general methodology applies to other machine learning tools, including the more advanced versions of the GA.

Finally, while, in this paper, we are only interested in the statistical foundation, we do not exclude the possibilities of having other foundations. As a matter of fact, we believe that a firm statistical foundation can show us where to ask the crucial questions, and that will help build a more general mathematical foundation. For example, in this paper, we have been already well motivated by the question as to why the GA performed quite poorly in the pure nonlinear stochastic processes, but

performed well in the chaotic processes. Of course, this statistical finding alone may need more work before coming to its maturity. However, the point here is that theoretical questions regarding the GA's performance cannot be meaningfully answered unless we have firmly grasped their behavior in a statistical way.

NOTES

1. The interested reader can obtain more spread applications in the fields of research from Goldberg (1989).

2. A bibliographic list of financial applications of genetic algorithms and genetic programming can be found in Chen and Kuo (2002) and Chen and Kuo (2003). For a general coverage of this subject, interested readers are referred to Chen (1998a), Chen (2002) and Chen and Wang (2003). As opposed to the conventional technical analysis, the advantages of using GAs and GP are well discussed in Allen and Karjalainen (1999), and is also briefly reviewed in another paper of this special issue. (Yu et al., 2004).

3. For example, Chen (1998b) sorted out three *stochastic properties* which may impinge upon the performance of GAs in financial data mining. These are the *no-free-lunch property*, the *well-ordered property* and the *existence of temporal correlation*. Several tests of these properties are then proposed and an a *priori* evaluation of the potential of GAs can be made based on these proposed tests.

4. $\bar{\pi}^*$ is a sample average of π_i^*, which is the accumulated return earned by the omniscient trader in the ith ensemble of the Monte Carlo simulation.

5. Doing this enables us to apply the central limit theorem to derive the asymptotic distribution of the various test statistics mentioned in Section 4.

6. The main source of this dataset is the interbank spot prices published by Dow Jones in a multiple contributors page (the TELERATE page). This covers markets worldwide 24 hours a day. These prices are quotations of the average prices of bid and ask and not actual trading prices. Furthermore, they are irregularly sampled and therefore termed as *tick-by-tick* prices.

7. The clear cut-off pattern appearing at the first lag suggests that these series involve a MA(1) process. Later on, from more rigorous statistics, we will see that indeed it is the case.

8. The detailed description can be found in Chen and Tan (1996).

9. Once the linear signals are filtered out, any signals left in the residual series must be nonlinear. "BDS" stands for "Brock, Dechert and Scheinkman" see Brock et al. (1996).

ACKNOWLEDGMENTS

An earlier version of this paper has been presented at the 2003 International Conference on Artificial Intelligence (IC-AI03). The first author is grateful for the research support from NSC grant No. NSC 91-2415-H-004-005. We also thank the two anonymous referees for their comments and suggestions and Bruce Stewart for proof reading this paper.

REFERENCES

Allen, F., & Karjalainen, R. (1999). Using generic algorithms to find technical trading rules. *Journal of Financial Economics*, *51*, 245–271.

Arnold, S. F. (1990). *Mathematical statistics*. New Jersey: Prentice-Hall.

Barnett, W. A., Gallant, A. R., Hinich, M. J., Jungeilges, J. A., Kaplan, D. T., & Jensen, M. J. (1997). A single-blind controlled competition among tests for nonlinearity and chaos. Paper presented at the 1997 Far Eastern Meeting of the Econometric Society (FEMES'97), Hong Kong, July 24–26, 1997 (Session 4A).

Bauer, R. J. (1994). *Genetic algorithms and investment strategies*. Wiley.

Bollerslev, T. P. (1986). Generalized autoregressive conditional heteroskedasticity. *Journal of Econometrics*, *31*, 307–327.

Bollerslev, T., Chou, R. Y., & Kroner, K. F. (1992). ARCH modeling on finance: A review of the theory and empirical evidence. *Journal of Econometrics*, *52*, 5–59.

Box, G. E. P., & Jenkings, G. M. (1976). *Time series analysis: Forecasting and control*. San Fransisco: Holden-Day.

Brock, W. A., Dechert, W. D., Scheinkman, J., & LeBaron, B. (1996). A test for independence based on the correlation dimension. *Econometric Reviews*, *15*, 197–235.

Brock, W. A., Hsieh, D., & LeBaron, B. (1991). *Nonlinear dynamics, chaos and instability*. Cambridge, MA: MIT Press.

Campbell, J. Y., Lo, A. W., & MacKinlay, A. C. (1997). *The econometrics of financial markets*. Princeton University Press.

Chen, S.-H. (1998a). Evolutionary computation in financial engineering: A road map of GAs and GP. Financial Engineering News 2, No. 4. Also available from the website: http://www.fenews.com/1998/v2n4/chen.pdf.

Chen, S.-H. (1998b). Can we believe that genetic algorithms would help without actually seeing them work in financial data mining? In: L. Xu, L. W. Chan, I. King & A. Fu (Eds), *Intelligent Data Engineering and Learning: Perspectives on Financial Engineering and Data Mining* (Part I, The Foundations, pp. 81–87). Singapore: Springer-Verlag.

Chen, S.-H. (Ed.) (2002). *Genetic algorithms and genetic programming in computational finance*. Kluwer.

Chen, S.-H., & Kuo, T.-W. (2002). Evolutionary computation in economics and finance: A bibliography. In: S.-H. Chen (Ed.), *Evolutionary Computation in Economics and Finance* (pp. 419–455). Physica-Verlag.

Chen, S.-H., & Kuo, T.-W. (2003). Discovering hidden patterns with genetic programming. In: S.-H. Chen & P. P. Wang (Eds), *Computational Intelligence in Economics and Finance* (pp. 329–347). Springer-Verlag.

Chen, S.-H., & Lu, C.-F. (1999). Would evolutionary computation help for designs of artificial neural nets in financial applications? In: Proceedings of 1999 Congress on Evolutionary Computation. IEEE Press.

Chen, S.-H., & Tan, C.-W. (1996). Measuring randomness by rissanen's stochastic complexity: Applications to the financial data. In: D. L. Dowe, K. B. Korb & J. J. Oliver (Eds), *ISIS: Information, Statistics and Induction in Science* (pp. 200–211). Singapore: World Scientific.

Chen, S.-H., & Tan, C.-W. (1999). Estimating the complexity function of financial time series: An estimation based on predictive stochastic complexity. *Journal of Management and Economics*, *3*.

Chen, S.-H., & Wang, P. P. (2003). Computational intelligence in economics and finance. In: S.-H. Chen & P. P. Wang (Eds), *Computational Intelligence in Economics and Finance* (pp. 3–55). Springer-Verlag.

Dickey, D. A., & Fuller, W. A. (1979). Distribution of the estimators for autoregressive time series with a unit root. *Journal of the American Statistical Association*, *74*, 427–431.

Drunat, J., Dufrénot, G., & Mathieu, L. (1998). Modelling burst phenomena: Bilinear and autogressive exponential models. In: C. Dunis & B. Zhou (Eds), *Nonlinear Modelling of High Frequency Financial Time Series* (pp. 201–221). Wiley.

Engle, R. F. (1982). Autoregressive conditional heteroscedasticity with estimates of the variance of U.K. inflation. *Econometrica*, *50*, 987–1008.

Goldberg, D. E. (1989). *Genetic algorithms in search, optimization and machine learning*. Addison-Wesley.

Granger, D. W. J., & Anderson, A. P. (1978). *An introduction to bilinear time series models*. Gottingen and Zurich: Vandenhoech & Ruprecht.

Holland, J. H. (1975). *Adaptation in natural and artificial systems*. Ann Arbor, MI: University of Michigan Press.

Jarque, C. M., & Bera, A. K. (1980). Efficient tests for normality, homoscedasticity and serial independence of regression residuals. *Economic Letters*, *6*, 255–259.

Jobson, J. D., & Korkie, B. M. (1981). Performance hypothesis testing with the Sharpe and Treynor measures. *Journal of Finance*, *36*(4), 889–908.

Moody, J., & Wu, L. (1997). What is the "true price"? – state space models for high frequency FX data. Proceedings of the Conference on Computational Intelligence for Financial Engineering. IEEE Press.

Palmer, R. G., Arthur, W. B., Holland, J. H., LeBaron, B., & Taylor, P. (1994). Artificial economic life: A simple model of a stockmarket. *Physica D*, *75*, 264–274.

Refenes, A.-P. (1995). Testing strategies and metrics. In: A.-P. Refenes (Ed.), *Neural Networks in the Capital Markets* (pp. 67–76). New York: Wiley.

Roll, R. (1984). A simple implicit measure of the effective bid-ask spread in an efficient market. *Journal of Finance*, *39*, 1127–1139.

Sharpe, W. F. (1966). Mutual fund performance. *Journal of Business*, *39*(1), 119–138.

Subba-Rao, T. (1981). On the theory of bilinear time series models. *Journal of the Royal Statistical Society, Series B*, *43*, 244–255.

Subba-Rao, T., & Gabr, M. M. (1980). An introduction to bispectral analysis and bilinear time series models. In: *Lecture Notes in Statistics* (Vol. 24). New York: Springer-Verlag.

Tong, H. (1983). Threshold models in nonlinear time series analysis. In: *Lecture Notes in Statistics* (Vol. 21). Heidelberg: Springer-Verlag.

Tong, H. (1990). *Non-linear time series: A dynamical system approach*. New York: Oxford University Press.

Yu, T., Chen, S.-H., & Kuo, T.-W. (2004). A genetic programming approach to model international short-term capital flow. *Advances in Econometrics* (special issue of 'Applications of AI in Finance & Economics').

Zhou, B. (1996). High-frequency data and volatility in foreign-exchange rates. *Journal of Business and Economic Statistics*, *14*(1), 45–52.

APPENDIX A

A.1. Coding Trading Strategies

Based on the trading formulation (3), to encode a trading strategy, we only need to encode the CONDS with three primitive predicates, which means the following three parts:

- $\vec{a} = (a_1, a_2, a_3)$,
- $\vec{\oplus} = (\oplus_1, \oplus_2, \oplus_3)$,
- the logical combination of the three predicates $\text{Cond}(r_{t-i})$ $(i = 1, 2, 3)$.

To encode $\vec{a}$, we first transform the range of the variable Z, $[Z_{\min}, Z_{\max}]$, into a fixed interval, say [0, 31].

$$Z^* = \frac{Z - Z_{\min}}{Z_{\max} - Z_{\min}} \times 32 \tag{A.1}$$

Then Z^* will be further transformed by Eq. (A.2).

$$Z^{**} = \begin{cases} n, & \text{if } n \le Z^* < n+1 \\ 31 & \text{if } Z^* = 32 \end{cases} \tag{A.2}$$

Since there are only 32 cutoff values, each a_i can be encoded by a 5-bit string. Hence the vector $\vec{a}$ can be encoded by a 15-bit binary string. To encode $\vec{\oplus}$, notice that each $\oplus$ has only two possibilities: $\ge$ or $<$. Therefore, a $\vec{\oplus}$ can be encoded by a 3-bit binary string (Table A.1). Finally, there are a total of totally 8 logical combinations for three predicates and they can be encoded by 3-bit strings (Table A.2).

In sum, a CONDS can be encoded by a 21-bit string (3 for logical combinations, 3 for inequalities, and 15 for the three thresholds). Therefore, each trading strategy can be represented by a 21-bit string.

Table A.1. Binary Codes for Inequality Relation.

Code	$\oplus_1$	$\oplus_2$	$\oplus_3$
0(000)	$\ge$	$\ge$	$\ge$
1(001)	$<$	$\ge$	$\ge$
2(010)	$\ge$	$<$	$\ge$
3(011)	$\ge$	$\ge$	$<$
4(100)	$<$	$<$	$\ge$
5(101)	$<$	$\ge$	$<$
6(110)	$\ge$	$<$	$<$
7(111)	$<$	$<$	$<$

Table A.2. Binary Codes for Logical Combinations.

Logic Code	Logical Combination of Predicates
0(000)	Cond 1 OR (Cond 2 AND Cond 3)
1(001)	Cond 1 AND (Cond 2 OR Cond 3)
2(010)	(Cond 1 OR Cond 2) AND Cond 3
3(011)	(Cond 1 AND Cond 2) OR Cond 3
4(100)	(Cond 1 OR Cond 3) AND Cond 2
5(101)	(Cond 1 AND Cond 3) OR Cond 2
6(110)	Cond 1 OR Cond 2 OR Cond 3
7(111)	Cond 1 AND Cond 2 AND Cond 3

A.2. Ordinary Genetic Algorithms

The GA described below is a very basic version of a GA, and is referred to as the ordinary genetic algorithm (OGA). More precisely, it is very similar to the GA employed in Bauer (1994).

- The genetic algorithm maintains *a population of individuals*,

$$P_i = \{g_1^i, \ldots, g_n^i\} \tag{A.3}$$

  for iteration i, where n is *population size*. Usually, n is treated as fixed during the whole evolution. Clearly, $P_i \subset G$.
- *Evaluation step:* Each individual g_j^i represents a trading strategy at the ith iteration (population). It can be implemented with the *historical data* r_{t-1}, r_{t-2}, and r_{t-3} by means of Eq. (2). A specific example is given in Eq. (3). Each trading strategy g_j^i is evaluated by a *fitness* function, say Eq. (6).
- *Selection step:* Then, a new generation of population (iteration $i + 1$) is formed by randomly selecting individuals from P_i in accordance with a *selection scheme*, which, in this paper, is the *roulette-wheel selection scheme*.

$$M_i = P_s(P_i) = (s_1(P_i), s_2(P_i), \ldots, s_n(P_i)) \tag{A.4}$$

  where

$$s_k : \left\{ \binom{G}{n} \right\} \to G, \tag{A.5}$$

  $k = 1, 2, \ldots, n$, and $\left\{ \binom{G}{n} \right\}$ is the set of all populations whose population size is n. The set M_i is also called the *mating pool*.
- *Alteration step:* Some members of the new population undergo transformations by means of *genetic operators* to form new solutions.

- *Crossover:* We use *two-point crossover* c_k, which create new individuals by combining parts from two individuals.

$$O_i = P_c(M_i) = (c_1(M_i), c_2(M_i), \ldots, c_{n/2}(M_i)) \tag{A.6}$$

where

$$c_k: \left\{ \binom{G}{n} \right\} \to G \times G, \tag{A.7}$$

$k = 1, 2, \ldots, n/2$. O_i is known as the set of *offspring* in the GA.

- *Mutation:* We use *bit-by-bit mutation* m_k, which creates new individuals by flipping, with a small probability, each bit of each individual of O_i.

$$P_{i+1} = P_m(O_i) = (m_1(O_i), m_2(O_i), \ldots, m_n(O_i)) \tag{A.8}$$

where

$$m_k: \left\{ \binom{G}{n} \right\} \to G \tag{A.9}$$

$k = 1, 2, \ldots, n$.

- After the evaluation, selection and alteration steps, the new population P_{i+1} is generated. Then we proceed with the three steps with P_{i+1}, and the loop goes over and over again until a termination criterion is met. The control parameters employed to run the OGA are given in Table A.3.

Table A.3. Control Parameters of OGA.

Number of generations	100
Population size (n)	100
Selection scheme	Roulette-wheel
Fitness function	Accumulated returns
Elitist strategy	Yes
Rank min	0.75
Crossover style	Two-Point
Crossover rate	0.6
Mutation rate	0.001

A GENETIC PROGRAMMING APPROACH TO MODEL INTERNATIONAL SHORT-TERM CAPITAL FLOW

Tina Yu, Shu-Heng Chen and Tzu-Wen Kuo

ABSTRACT

We model international short-term capital flow by identifying technical trading rules in short-term capital markets using Genetic Programming (GP). The simulation results suggest that the international short-term markets was quite efficient during the period of 1997–2002, with most GP generated trading strategies recommending buy-and-hold on one or two assets. The out-of-sample performance of GP trading strategies varies from year to year. However, many of the strategies are able to forecast Taiwan stock market down time and avoid making futile investment. Investigation of Automatically Defined Functions shows that they do not give advantages or disadvantages to the GP results.

1. INTRODUCTION

Hot money, or speculative capital, is raising some concerns in Chinese economy. During the first half of this year (2003), about US$25 billion in short-term speculative funds sneaked into China as investors bet on possible sharp appreciation

Applications of Artificial Intelligence in Finance and Economics
Advances in Econometrics, Volume 19, 45–70

ISSN: 0731-9053/doi:10.1016/S0731-9053(04)19002-6

of the local currency *Renminbi*. Speculative capital mostly flows into areas with high liquidity, such as the security and bonds markets, as it is for short-term investments. Without being invested in industries, this money usually does not damage the overall economy once it is withdrawn.[1] Nevertheless, Chinese government has to take heed of possible longer-term fallout from speculation.

Unlike the normal direct investment, speculative capital moves very quickly among international capital markets, sometimes with very huge amount (as the Asian Crisis has demonstrated). Therefore, it can be always a potential threat for macroeconomic stability. If we can predicate the short-term capital movements, it becomes possible to control and to stabilize the economy under the influence of hot money.

In short-term international capital movements, technical trading rules play an important role as they reveal investors' behavior. This work models international short-term capital flow by identifying technical trading rules in short-term capital markets. Through the simulation, we investigate if there exists trading strategies that are capable of predicting the capital inflow and outflow, hence make profitable investment. The modeling and simulation were conducted using Genetic Programming (GP) (Koza, 1992), a novel approach for this task. Its effectiveness will be analyzed and discussed.

As a first step, we use Taiwan as the host country and model the short-term capital flow between Taiwan and four other foreign countries: United States, Hong Kong, Japan and United Kingdom. In other words, the speculator resides in Taiwan, investing Taiwan currency to other foreign assets to pursue the highest returns. The two types of short-term assets considered here are currency and stocks, whose

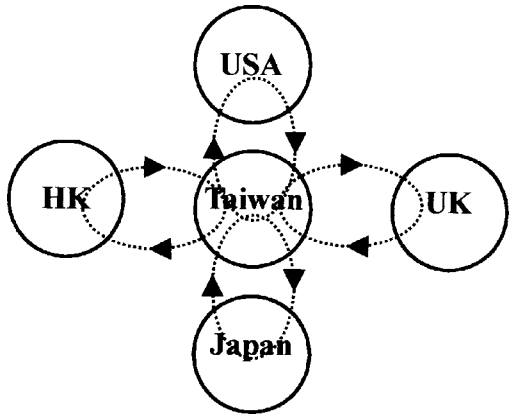


Fig. 1. The Global Short-term Capital Flow Model.

transactions are governed by stock markets and foreign exchange markets. This overall model gives a global picture of the short-term capital inflow and outflow between Taiwan and four foreign countries (see Fig. 1).

The paper is organized as follows. Section 2 gives the background of this work. It explains technical analysis in financial markets and surveys the applications of GP to model financial trading strategies. Section 3 describes the capital flow model representation and Section 4 gives the GP trading strategy structure. The financial data used for modeling and simulation are explained in Section 5. Section 6 gives the GP experimental setup. In Section 7, the benchmark used to evaluate GP trading strategies is explained. Section 8 presents the experimental results. The analysis of GP trading strategies is presented in Section 9. Finally, Section 10 gives the concluding remarks and outlines the direction of future work.

2. BACKGROUND

One driving force of short-term capital movement is the opportunities of profit. The prediction of short-term capital flow can therefore be viewed as the forecast of positive investment returns. One empirical approach to identify profitable capital trading is technical analysis. This approach uses historical price information to study price trends. This technique was originated from the work of Charles Dow in the late 1800 and is now widely used by investment professionals as inputs for trading decisions (Pring, 1991).

Based on technical analysis techniques, various trading rules have been developed. Examples include *moving average, filter* and *trading-range break* (see Section 4.2 for more explanation). In (Brock et al., 1992), they reported that *moving average* and *trading-range break* give significant positive returns on Dow Jones Index from 1897 to 1986. Similarly, Cooper (1999) showed that *filter* strategy can out-perform buy-and-hold under relatively low transaction cost on NYSE and AMEX stocks for the 1962–1993 period. These studies are encouraging evidences indicating that it is possible to devise profitable trading rules for financial markets.

However, one concern toward these studies is that the investigated trading rules are decided *ex post*. It is possible that the selected trading rule is favored by the tested time periods. If the investor has to make a choice about what rule or combination of rules to use at the beginning of the sample period, the reported returns may have not occurred. In order to obtain true out-of-sample performance, GP has been used to derive the trading rules for analysis (Allen & Karjalainen, 1999; Neely et al., 1997; Neely & Weller, 1999; Wang, 2000).

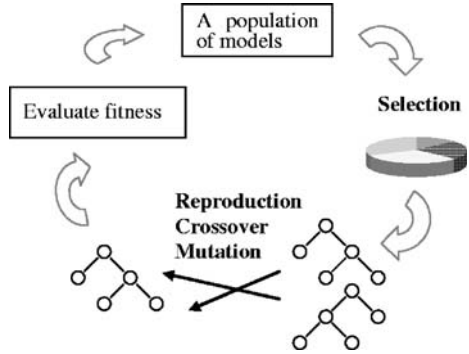


Fig. 2. Genetic Programming Cycle.

2.1. Genetic Programming

GP is a population-based search algorithm developed by John Koza (1992). It mimics the process of natural evolution to search for optimal solutions of a given problem. Figure 2 depicts the GP process cycle. Initially, a population of models is randomly created. Based on their fitness, better models are selected for reproduction. Using alteration operations, such as crossover and mutation, new offspring are generated to form a new generation. This process of selection, alteration and fitness evaluation continues until the specified termination criterion is met. The best model at the end of the process is the final model.

Various representations, selection and alteration schemes have been proposed to suit different applications. In this work, the model is represented as a parse tree that is evaluated to give trading decisions. The financial return after executing the trading decisions becomes the fitness of the model. Section 4 gives more details on the structure of GP trading strategies.

2.2. Related Works

Targeted toward different financial markets, different researchers have applied GP to generate trading rules and to analyze their profitability. For example, Allen and Karjalainen (1999) studied S&P 500 index from 1928 to 1995. They reported that the evolved GP trading rules do not earn consistent excess returns over buy-and-hold after the transaction costs. In contrast, Neely et al. (1997) reported that their GP trading rules for foreign exchange markets were able to gain excess returns for six exchange rates over the period of 1981–1995. Wang

(2000) suggested that this conflicting result might be due to the fact that foreign exchange markets have a lower transaction cost than the trading cost in the S&P index stock market. Another reason Wang suggested is that Neely et al. did not use the rolling forward approach to test their results for different time periods while Allen and Karjalainen did (see Section 5 for the explanation of rolling forward approach). Finally, Wang pointed out that these two works used different benchmarks to assess their GP trading rules: Allen and Karjalainen used the return from buy-and-hold while Neely et al. used zero return, because there is no well-defined buy-and-hold strategy in the foreign exchange markets.

Using a similar GP setup as that of Allen and Karjalainen (1999), Wang (2000) also investigated GP rules to trade in S&P 500 futures markets alone and to trade in both S&P 500 spot and futures markets simultaneously. He reported that GP trading rules are not able to beat buy-and-hold in both cases. Additionally, he also incorporated Automatically Defined Function (ADF) (Koza, 1994) in his GP experiments. He reported that ADFs made the representation of the trading rules simpler by avoiding duplication of the same branches. In his work, Wang did not compare the results from GP with the results from ADF-GP.

Similar to the trading model of Wang, our short-term capital flow model allows trading in two kinds of financial markets (stock and foreign exchange) simultaneously. Moreover, we also included ADFs in our GP implementations. However, the implementations of our ADFs have more variation than that of Wang's. We also used a different data transformation method to normalize time series. Consequently, the evolved GP trading rules have different interpretations (see Section 9.1).

There are other works using Genetic Algorithms (GA) and/or Neural Network (NN) to make investment decisions. For example, Kassicieh et al. (1997) applied GA to determine the time to trade in different financial markets by selecting a subset of 10 given economic indicator time series. Baba et al. (2000) applied GA/NN hybrid to devise their decision support system for trading in Tokyo stock markets. Although GA and NN are powerful modeling tools, we find GP more suitable for our work because it has a natural representation (S-expression) for modeling trading rules. If we use GA or NN, there is an inevitable extra task of mapping the GA and NN structures to the technical trading rules.

3. MODEL REPRESENTATION

The representation of our short-term capital flow model between Taiwan and a foreign country is a directed graph. Each node in the graph represents an asset. For example, Fig. 3 gives the capital flow model between Taiwan and United States.

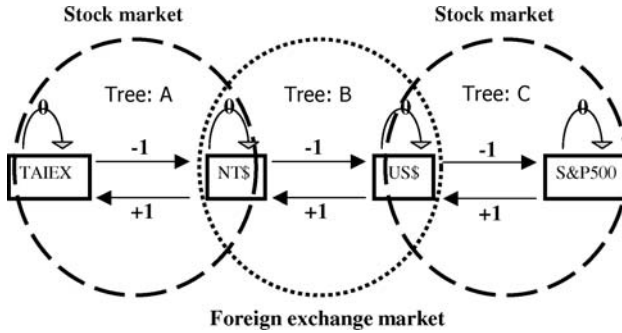


Fig. 3. The Short-term Capital Flow Model Represented as a Directed Graph.

From the left to the right, the four nodes represent Taiwan stock market (TAIEX), Taiwan currency (NT$), United States currency (US$), and United States stock market (S&P 500). This model encompasses three capital markets: Taiwan stock market, U.S. stock market and Taiwan-U.S. currency exchange market.

Funds in one asset can be transferred into one or more other assets, through the transactions in the related capital markets. For example, funds in NT$ may be changed to US$ by trading in Taiwan foreign exchange market. One can also use them to buy Taiwan stocks in the Taiwan stock market.

Initially, the fund is placed in foreign currency. At each time step, the fund may be reallocated to other assets, according to the trading decisions made for the three capital markets. These three decisions made up the trading strategies to be carried out by an investor. More details on GP trading strategies are given in the following section.

A trading decision may be to buy an asset, to sell an asset or to do nothing. For the purpose of generality, we structure a financial market with two assets, one on left and one on right. When the decision is to transfer a fund from the asset on the right to the asset on the left, a "+1" is signaled. When the decision is to transfer a fund from the asset on the left to the one on the right, a "−1" is signaled. Signal "0" means do nothing. Table 1 gives the 27 possible combinations of trading decisions. Assuming at time t, the fund in TAIEX is A, in NT$ is B, in US$ is C and in S&P500 is D, the table gives the fund allocations at time $t + 1$.

When the decision is to trade (+1 or −1), half of the current fund is transferred to the designated asset. For example, if the trading strategy is $\{-1, -1, -1\}$, half of the TAIEX funds (A) will be moved to NT$; half of the original NT$ fund (B) will be moved to US$ and half of the original US$ fund (C) will be moved to S&P500. A trading strategy may cause an original fund to be completely transferred out, e.g. $\{1, -1, -1\}$ trades all NT$ with TAIEX and US$. However, the maximum amount

Table 1. Trading Decisions and Their Funds Reallocation Results.

TSM	CEM	FSM	TAIEX$_{t+1}$	NT$\$_{t+1}$	U.S.$\$_{t+1}$	S&P500$_{t+1}$
−1	−1	−1	0.5A	0.5A + 0.5B	0.5B + 0.5C	0.5C + D
−1	−1	0	0.5A	0.5A + 0.5B	0.5B + C	D
−1	−1	1	0.5A	0.5A + 0.5B	0.5B + C + 0.5D	0.5D
−1	0	−1	0.5A	0.5A + B	0.5C	0.5C + D
−1	0	0	0.5A	0.5A + B	C	D
−1	0	1	0.5A	0.5A + B	C + 0.5D	0.5D
−1	1	−1	0.5A	0.5A + B + 0.5C	0	0.5C + D
−1	1	0	0.5A	0.5A + B + 0.5C	0.5C	D
−1	1	1	0.5A	0.5A + B + 0.5C	0.5C + 0.5D	0.5D
0	−1	−1	A	0.5B	0.5B + 0.5C	0.5C + D
0	−1	0	A	0.5B	0.5B + C	D
0	−1	1	A	0.5B	0.5B + C + 0.5D	0.5D
0	0	−1	A	B	0.5C	0.5C + D
0	0	0	A	B	C	D
0	0	1	A	B	C + 0.5D	0.5D
0	1	−1	A	B + 0.5C	0	0.5C + D
0	1	0	A	B + 0.5C	0.5C	D
0	1	1	A	B + 0.5C	0.5C + 0.5D	0.5D
1	−1	−1	A + 0.5B	0	0.5B + 0.5C	0.5C + D
1	−1	0	A + 0.5B	0	0.5B + C	D
1	−1	1	A + 0.5B	0	0.5B + C + 0.5D	0.5D
1	0	−1	A + 0.5B	0.5B	0.5C	0.5C + D
1	0	0	A + 0.5B	0.5B	C	D
1	0	1	A + 0.5B	0.5B	C + 0.5D	0.5D
1	1	−1	A + 0.5B	0.5B + 0.5C	0	0.5C + D
1	1	0	A + 0.5B	0.5B + 0.5C	0.5C	D
1	1	1	A + 0.5B	0.5B + 0.5C	0.5C + 0.5D	0.5D

Note: TSM: Taiwan Stock Market; CEM: Currency Exchange Market; FSM: Foreign Stock Market. The table is simplified in that no transaction cost is considered. The modeling process, however, does take transaction cost into account.

of fund that one asset can acquire is half of its two neighboring assets. For example, the trading strategy $\{0, -1, 1\}$ leads to an increase of US$ by half of the original NT$ fund and half of the original S&P500 fund. This is a rather conservative setup. We therefore adjust the modeling and simulation procedure to execute a transaction multiple (10) times in one time step,[2] with the transaction cost charged once only. This leads to almost 100% of the original fund in one asset to be transferred to the designated asset in one time step. Nevertheless, to reallocate all the funds $(A + B + C + D)$ into single asset, it still requires at least three time steps.

To be close to the reality, the model does not allow direct capital flow between international stocks. In the real world, the trading between stocks in two different

Table 2. Trading Decision Table.

Rule 1 Recommendation	Rule 2 Recommendation	Final Decision
True	False	+1
False	True	−1
True	True	0
False	False	0

countries requires an intermediate step of currency exchange. For example, to trade a Taiwan stock with a U.S. stock, the Taiwan stock has to be cashed into Taiwan currency, which is exchanged to U.S. currency, which is then used to purchase the U.S. stock.

4. GP TRADING STRATEGIES

A GP trading strategy consists of three trading decisions made for the three financial markets. Each trading decision (+1, −1 or 0) is determined by a pair of GP rules. The first rule decides whether to move funds from the right asset to the left asset (True) or not (False). The second rule decides whether to move funds from the left asset to the right asset (True) or not (False). The final decision is derived according to Table 2.

A GP rule has a tree structure. Figure 4 gives a trading rule example. It says, "If the 15-day moving average is greater than the 250-day moving average, then trade. Otherwise, if the closing exchange rate has risen by more than 1% above its minimum over the previous 10 days, then trade. Otherwise, do not trade."

With three trading decisions, each is determined by two rules; a GP trading strategy consists of six GP trees. Figure 5 gives the structure of a GP trading strategy. Note that the labels Tree-A, Tree-B and Tree-C correspond to those in Fig. 3.

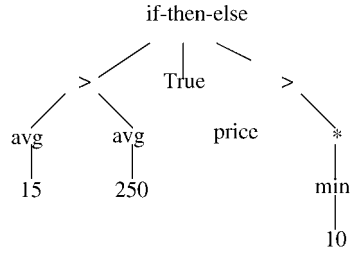


Fig. 4. A GP Trading Rules Example.

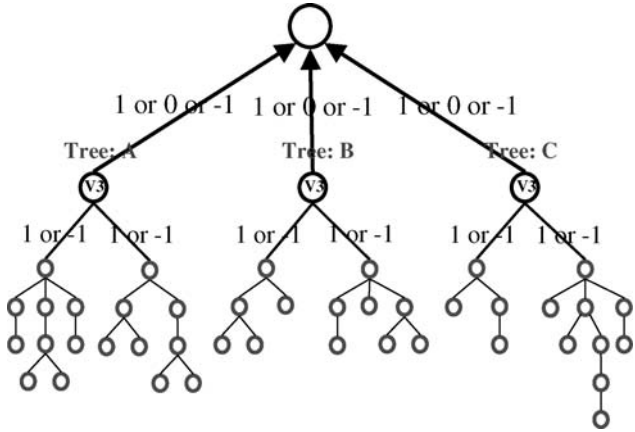


Fig. 5. The GP Trading Strategy Structure.

The following functions are provided to construct the internal nodes of a GP tree:

- Boolean function: *and, or, not*, $<$, $>$, *if-then-else*
- Numerical function: $+$, $-$, $\times$, $\div$, *average, max, min, norm, lag*

The function *average* computes the moving average of a variable in a time window specified by the integer argument. For example, *average* (x, 250) at time t is the arithmetic mean of $x_{t-1}, x_{t-2}, \ldots, x_{t-250}$. The function *max* returns the largest value of a variable during a time window specified by the integer argument. For example, *max* (y, 3) at time t is equivalent to $max(y_{t-1}, y_{t-2}, y_{t-3})$. Similarly, the function *min* returns the smallest value of a variable during a time window specified by the integer argument. The function *norm* computes the absolute value of the given real number. The function *lag* returns the value of a variable lagged by a number of days specified by the integer argument. For example, *lag* (z, 3) at time t is z_{t-3}. These functions are commonly used by financial traders to decide their trading strategies, hence are reasonable building blocks for GP to construct trading rules.

GP tree leaf nodes can be a value from the following three types of terminals:

- Input variables: TW_{IR}, TW_{SI}, FC_{IR}, FC_{SI}, NTD/FD
- Numerical constants: 100 constants randomly generated between 0.0 and 10.0
- Boolean constants: True, False

Input variables include: *interest rate* in Taiwan (TW_{IR}) and the foreign country (FC_{IR}); *stock index* in Taiwan stock market (TW_{SI}) and the foreign country stock market (FC_{SI}); the *exchange rate* between Taiwan and the foreign country (NTD/FD). These financial time series will be explained with more details in

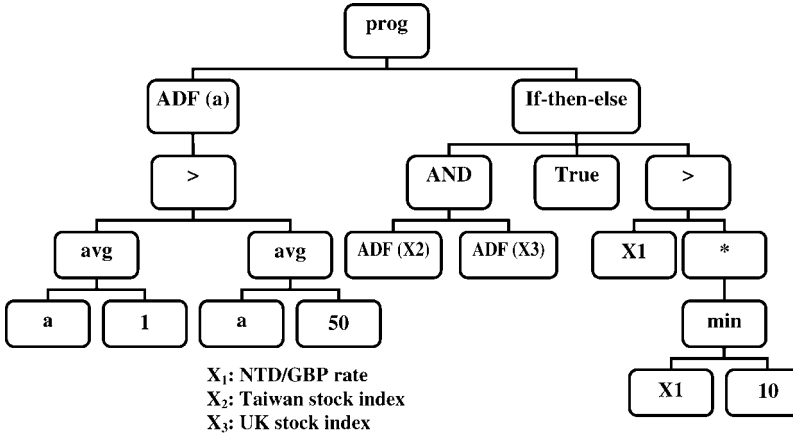


Fig. 6. An ADF-GP Trading Strategy Example.

Section 5. Real-valued constants may be truncated into integer value if they are passed over to time series functions, such as *lag*.

4.1. Automatically Defined Functions

Automatically Defined Function (ADF) is a mechanism devised by Koza to extend GP ability to solve problems with regularity, symmetry and homogeneity (Koza, 1994). ADFs are subroutines that are simultaneously evolved with the GP main programs. Figure 6 gives an example GP trading rule with one ADF. The left branch of the tree is an ADF while the right branch is the main trading rule. The ADF takes one argument (a time series variable) and checks if its 1-day moving average is greater than its 50-day moving average. This ADF is called twice in the GP main trading rule: ADF($\times 2$) takes Taiwan stock index as the argument while ADF($\times 3$) takes UK stock index as the argument.

An ADF is evolved simultaneously with the GP main trading rule. If a trading rule contains patterns, ADF-GP may discover and extract them as ADFs, which are then called from the GP main trading rule. We implemented ADFs in three different ways for three different purposes:

- One ADF is included in each trading strategy. This investigates whether regularity exists in profitable trading strategies and as to whether GP is able to discover them. Since the time series are transformed by dividing them by 250-day moving average (see Section 5), ADF is used to identify patterns in the *change of trend* that provide profitable trading.

- One partially defined ADF is included in each trading strategy. The ADF is initially seeded with one of the commonly used technical trading rules (see Section 4.2). They are then evolved during the GP runs. With the transformed time series, this implementation is to discover if the provided technical trading rules (and their variations) are effective on the transformed time series data.
- Three partially defined ADFs are included in each trading strategy. This is the same as the above except three, instead of one, ADFs are used.

The function and terminal sets used to evolve ADFs are the same as that used to evolve the GP main program. For ADF-GP, an extra function (the name of the ADF) is included in the GP main program function set.

4.2. Technical Trading Rules

Two types of technical trading rules are provided for GP to initialize its ADFs: *moving average rules* and *filter rules*. Moving average rules include a class of rules where the trading signals are decided by comparing a short-run with a long-run moving average in the same time series, producing a "buy" signal when the short-run moving average cuts the long-run moving average from below. This rule can be implemented in many different ways by specifying different short and long periods. We have included the following five implementations: (1–50), (1–150), (5–150), (1–200), and (2–200), where the first number is the short while the second number indicates the long. We also implemented a band moving average rule, where the band is 0.01, i.e. signal "buy" if the short-run moving average exceeds the long-run moving average by 1%.

Filter rules include a class of trading rules where the trading signals are decided by comparing the current price with its local low or with its local high over a past period of time. We select three time lengths (50, 150, 200) to implement this class of rules. We also implemented two band filter rules, one with band 0.01 and the other with band -0.01. In the first case, a "buy" signal is generated if the current price exceeds the local high by 1%. In the second scenario, a "sell" signal is generated if the current price is below the local low by 1%.

Since these predefined ADFs are evolved, the final versions have different semantics and are not to be called the same names anymore.

5. DATA SET

We have acquired financial time series data for five countries (Taiwan, United States, United Kingdom, Japan and Hong Kong) between January 1, 1992 and

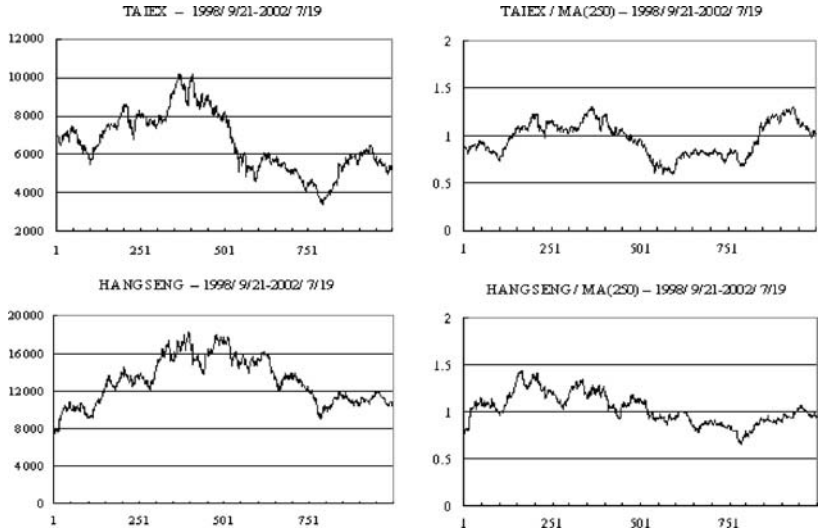
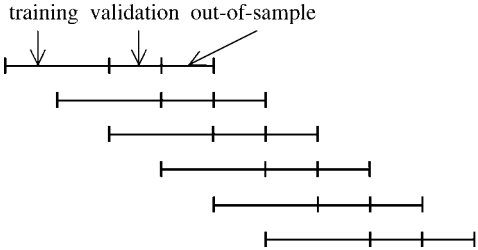


Fig. 7. Time Series Data Before and After Normalization.

December 31, 2002 from *Datastream*. The time series include: TW_{IR}, TW_{SI}, US_{IR}, US_{SI}, UK_{IR}, UK_{SI}, HK_{IR}, HK_{SI}, JP_{IR}, JP_{SI}, NTD/USD, NTD/GBP, NTD/JPY, NTD/HKD. Five times series are used to build one model. For example, TW_{IR}, TW_{SI}, US_{IR}, US_{SI} and NTD/USD are used to model Taiwan-U.S. capital flow.

Since the original time series are non-stationary, we transform them by dividing the daily data by a 250-day moving average. This is the method used by Allen and Karjalainen (1999) and Neely et al. (1997). The adjusted data oscillate around 1 and make the modeling task easier. Figure 7 gives two examples. On the left side are the two original series while on the right are the transformed ones. While the transformed series are used for modeling, the computation of GP trading strategies returns is based on the original time series. One implication of this data transformation is that GP is searching for patterns exhibited in the *change of trends* that give profitable trading strategies.

Over-fitting is an issue faced by all data modeling techniques. GP is no exception. When constructing/optimizing the trading strategies, GP tends to make the strategies producing maximum returns for the training period, which may contain noise that do not represent the overall series pattern. In order to construct trading strategies that generalize beyond the training data, we adopt two methods to run the GP experiments. The first one is to enforce parsimony pressures on the trading strategies structures, which will be discussed in Section 6. The second one is splitting the series into training, validation and out-of-sample periods. This is



1. 1993/12/06 – 1997/10/03
2. 1994/11/21 – 1998/09/18
3. 1995/11/06 – 1999/09/03
4. 1996/10/21 – 2000/08/18
5. 1997/10/06 – 2001/08/03
6. 1998/09/21 – 2002/07/19

Fig. 8. Six Sequences of Time Series Data.

a commonly used approach in machine learning and data mining. We adopt the rolling forward approach first proposed by Pesaran and Timmermann (1995) and also used by Allen and Karjalainen (1999) and Wang (2000).

To start, the first 500 data (250 used to transform raw data and 250 reserved to be referred by time series function such as *lag*) were removed. This leaves 2500 data in each time series. To guard against potential data snooping in the choice of time periods, the series are organized into 6 sequences, each with 1000 data points. Among them, 500 are for training, 250 are for validation and 250 are for out-of-sample testing. The data in one series may overlap with that in other series. As shown in Fig. 8, the second half of the training period and the entire validation period at the first series are the training period at the second series. The out-of-sample testing period at the first series is the validation period at the second series. With this setup, each out-of-sample testing period is one-year (short-term) and covers a different time period.

For each data series, 20 GP runs were made. The three data periods are used in the following manner:

(1) The best trading strategy against the training period at the initial population is selected and evaluated against the validation period. This is the initial "best strategy."
(2) A new generation of trading strategies is created by recombining/modifying parts of relatively fit strategies in the previous generation.
(3) The best trading strategies against the training period at the current population is selected and evaluated against the validation period.
(4) If this strategy has a better validation fitness than the previous "best strategy," then this is considered to be the new "best strategy."
(5) Go to step 2 until the maximum number of generation is reached or there is no fitter strategy is found after a certain number of generations (a controllable parameter).
(6) The last "best strategy" is tested against the out-of-sample period. This is what we use to evaluate the performance of GP trading strategies.

In summary, the training period is used to construct/optimize GP trading strategies while the validation period is used to select the GP trading strategies, which are then applied on the out-of-sample period to give the performance of the strategies. The analysis and evaluation are based on results from the out-of-sample period.

6. EXPERIMENTAL SETUP

The control parameters used to run GP experiments are given in Table 3. We experimented with different population size (200, 500 and 1000) to run for different number of generations (100 and 200). This setup is motivated by an observation reported by Chen and Kuo (2002, 2003) that population size and the number of generations have impact on GP search efficiency when modeling chaotic time series. These results will be compared in Section 8.

The GP system is generation-based, i.e. parents do not compete with offspring for selection and reproduction. This is a less aggressive search method compared to the steady-state-based GP where the offspring are used to replace less fit individuals in the population (Syswerda, 1991). Although steady-state-based GP has the advantage that fit offspring become available for reproduction right away, there are possibilities that the population becomes converged too fast hence leads to sub-optimal solutions.

We used tournament of size 2 to select winners. This means that two individuals are randomly selected and the one with a better fitness is the winner. For crossover operation, two winners are selected. For mutation or copy operation, only one winner is needed. The new population is generated with 70% of the individuals from crossover, 10% from point mutation, 10% from tree mutation and 10% from copy operation. The best individual in the current population is always copied over to the new generation.

Table 3. Control Parameters for GP Experiments.

Parameter	Value
Population size	200, 500, 1000
Maximum generation	100, 200
Crossover rate	70%
Point mutation rate	10%
Tree mutation rate	10%
Reproduction (copy)	10%
Elite	1
Maximum tree node	50
Maximum tree depth	17

The maximum tree depth of 17 is a hard constraint that cannot be violated. A GP strategy with tree depth larger than 17 is discarded. This is necessary to accommodate the computer resources. In contrast, the maximum number of tree node (50) is a soft constraint, which is handled using penalty explained in the following section (see Yu & Bentley, 1998) for more constraint handling methods.

As mentioned in Section 5, the best rule for a training period in each generation is evaluated against validation period. If the rule has a validation fitness that is better than the previous best rule has, it is saved as the new best rule. A GP run stops if no new best rule appears for 1/4 of the specified maximum number of generations or when the maximum number of generations is reached.

The fitness of an evolved GP trading strategy is the *gross return* (R) of the investment it generates. Initially, an investment of 1 unit is made in foreign currency. At the end of the time period, its final value is the *gross return*.

To determine the fitness of a GP trading strategy, it is applied on the normalized time series to produce a series of trading decisions for the three financial markets. This decision series are executed 10 times in each time step until the end of the time period. Every time a trading decision is executed, the amount of funds in each of the four assets may change (see Table 1). Let the amount of fund transferred from A to B be $\vec{A}$, from B to C be $\vec{B}$, from C to D be $\vec{C}$, from D to C be $\overleftarrow{D}$, from C to B be $\overleftarrow{C}$, from B to A be $\overleftarrow{B}$. Also, the associated one-way transaction costs are Cost_{AB}, Cost_{BC}, Cost_{CD}, Cost_{DC}, Cost_{CB} and Cost_{BA}. TW_{SI} is the Taiwan stock index and FC_{SI} is the foreign stock index. TW_{IR} is the Taiwan currency interest rate and FC_{IR} is the foreign currency interest rate. E is the exchange rate between the two currencies. At time $t+1$, the funds in each asset is given by:

$$
\begin{aligned}
A_{t+1} &= A_t - \vec{A} + \frac{\overleftarrow{B}}{\text{TW}_{\text{SI}(t)} \times (1 + \text{Cost}_{\text{BA}})} \\
B_{t+1} &= (B_t - \vec{B} - \overleftarrow{B}) \times (1 + \text{TW}_{\text{IR}(t)}) + \vec{A} \times \text{TW}_{\text{SI}(t)} \\
&\quad \times (1 - \text{Cost}_{\text{AB}}) + \overleftarrow{C} \times E_t \times (1 - \text{Cost}_{\text{CB}}) \\
C_{t+1} &= (C_t - \vec{C} - \overleftarrow{C}) \times (1 + \text{FC}_{\text{IR}(t)}) + \frac{\vec{B}}{E_t \times (1 + \text{Cost}_{\text{BC}})} + \overleftarrow{D} \\
&\quad \times \text{FC}_{\text{SI(t)}} \times (1 - \text{Cost}_{\text{DC}}) \\
D_{t+1} &= D_t - \overleftarrow{D} + \frac{\vec{C}}{\text{FC}_{\text{SI}(t)} \times (1 + \text{Cost}_{\text{CD}})}
\end{aligned}
$$

Different financial markets have different transaction costs. Moreover, within the same financial market, a transaction from asset A to asset B may have a different cost from that of a transaction from asset B to asset A. Table 4 gives the transaction cost implemented in this work. The costs associated with Taiwan stock market and

Table 4. Transaction Cost.

Transaction Type	Rate (%)
Cost_{AB}	0.4425
Cost_{BC}	0.2**
Cost_{CD}	0.1*
Cost_{DC}	0.43*
Cost_{CB}	0.2**
Cost_{BA}	0.1425

*Allen et al. (1999) used 0.1, 0.25 and 0.5% as the one-way transaction cost for S&P500 index market, while Wang (2000) used 0.12% for the same market.
**Neely et al. (1997) used 0.05% as the one-way transaction cost for foreign exchange markets.

Taiwan foreign exchange market are actual values. The costs associated with foreign country stock markets are estimated based on the fixed transaction tax charged to international investment and an estimated handling charge of 0.1%. Compared to the transaction cost for S&P500 stock market used by Allen and Karjalainen (1999) (0.1, 0.25 & 0.5%) and by Wang (2000) (0.12%), we have a higher transaction cost. Also, we have a higher transaction cost for foreign exchange market than that used by Neely et al. (1997) (0.05%). Normally, higher transaction costs discourage trades and reduces the number of transactions. This work intents to reflect the actual market operations, hence adapts the actual financial costs in the markets for modeling, in spite of the fact that they are higher than those used in other studies.

At the end of the time period (T), all assets are converted into the foreign currency:

$$B_{T+1} = B_T + A_T \times \text{TW}_{\text{SI}(T)} \times (1 - \text{Cost}_{\text{AB}})$$

$$C_{T+1} = C_T + \frac{B_{T+1}}{E_T \times (1 + \text{Cost}_{\text{BC}})} + D_T \times \text{FC}_{\text{SI}(T)} \times (1 - \text{Cost}_{\text{DC}})$$

The gross return is:

$$R = C_{T+1}$$

There is a penalty toward GP strategies that exceed the maximum number of 50 nodes. This soft constraint approach allows fitter strategies with a larger number of nodes to survive. Yet, it discourages tree size growth to avoid over-fitting, since trees with a large number of nodes tend to fit the training data so well that they lose their generality. The final fitness of a GP trading strategy is given in the following equation (Seshadri, 2003):

$$F = R \frac{50}{\max(\text{tree\_size},\ 50)}$$

7. BENCHMARK

The buy-and-hold (B&H) strategy is the most commonly used benchmark to evaluate financial trading strategies. With B&H, an investment made on one asset

Table 5. Return for the Buy-and-Hold Strategy.

Year	TW-U.S. Model	TW-HK Model	TW-JP Model	TW-UK Model
1997				
B&H(A)	1.2618	1.2627	1.3645	1.2431
B&H(B)	1.0187	1.0194	1.1016	1.0035
B&H(C)	1.0492	1.0536	1.0047	1.0594
B&H(D)	1.3523	1.2029	0.8240	1.3019
$R_{\text{B\& H}}$	**1.1705**	**1.1346**	**1.0737**	**1.1520**
1998				
B&H(A)	0.6792	0.6805	0.7364	0.6513
B&H(B)	0.8805	0.8822	0.9546	0.8444
B&H(C)	1.0492	1.0626	1.0042	1.0688
B&H(D)	1.0432	0.5012	0.7803	0.9488
$R_{\text{B\& H}}$	**0.9130**	**0.7816**	**0.8689**	**0.8783**
1999				
B&H(A)	1.2342	1.2362	1.0198	1.2862
B&H(B)	1.1340	1.1358	0.9370	1.1818
B&H(C)	1.0449	1.0492	1.0012	1.0558
B&H(D)	1.3186	1.8282	1.2897	1.2622
$R_{\text{B\& H}}$	**1.1829**	**1.3124**	**1.0619**	**1.1965**
2000				
B&H(A)	1.0192	1.0249	1.0139	1.0948
B&H(B)	1.0669	1.0728	1.0612	1.1460
B&H(C)	1.0527	1.0514	1.0003	1.0541
B&H(D)	1.0933	1.2960	0.9120	1.0209
$R_{\text{B\& H}}$	**1.0580**	**1.1113**	**0.9968**	**1.0790**
2001				
B&H(A)	0.4889	0.4893	0.5593	0.5101
B&H(B)	0.9351	0.9358	1.0698	0.9757
B&H(C)	1.0485	1.0494	1.0015	1.0555
B&H(D)	0.8056	0.6973	0.7592	0.8435
$R_{\text{B\& H}}$	**0.8195**	**0.7930**	**0.8474**	**0.8462**
2002				
B&H(A)	1.1983	1.1991	1.1205	1.0807
B&H(B)	1.0691	1.0698	0.9997	0.9642
B&H(C)	1.0158	1.0200	1.0000	1.0406
B&H(D)	0.7024	0.8454	0.8288	0.7377
$R_{\text{B\& H}}$	**0.9964**	**1.0336**	**0.9873**	**0.9558**

stays there until the end of time period. Since there are four assets in a model, the B&H strategy can be applied in four different ways: buy TAIEX and hold, buy NT$ and hold, . . ., etc. We therefore apply B&H over these four different assets. The average of their returns is used as the benchmark. Table 5 gives the B&H returns for the four different models.

8. RESULTS

For each of the four foreign countries modeled, we obtain 36 GP trading returns. These GP strategies are evolved based on 6 different data sequences using 3 different population sizes to run for 2 different numbers of generations. Each of the 36 results is the average of 20 trials. Table 6 gives the percentage of the GP trading strategies that out-performs the B&H strategy.

In TW-US, TW-JP and TW-UK models, most GP trading strategies out-perform B&H. In contrast, TW-HK model has a less number of GP trading strategies that give better returns than B&H. The number of statistically significant GP returns is given inside the parenthesis.

Different population sizes and number of generations make little difference on the GP results. For the small number of cases where they produce different results, there is not a consistent pattern showing larger (smaller) population size and/or longer (shorter) runs give better results. We checked the log files and found that most of the runs stop before generation 50 when no improved strategy on validation period was found.

Moreover, ADFs, in various form, provide no improvement in performance than the standard or "vanilla" GP model we used. For those runs where vanilla GP produces better returns than B&H, ADF-GP also gives better returns. Similarly, those runs where vanilla GP produces worse returns than the B&H method, the ADF-GP performs even worse. We will analyze the ADF-GP trading strategies and give explanation of this outcome in Section 9.1.

In this section, we analyze GP trading strategies based on the vanilla GP out-of-sample results, which are summarized in Table 7. In the table, six sets of data are

Table 6. Percentage of GP Trading Strategies Results that Out-performs B&H.

GP Implementation	TW-U.S. Model	TW-HK Model	TW-JP Model	TW-UK Model
Vanilla GP	29(19)/36	8(5)/36	19(11)/36	28(21)/36
GP with 1 ADF	27(20)/36	9(3)/36	22(9)/36	26(22)/36
GP with 1 partially defined ADF	30(21)/36	13(4)/36	23(11)/36	26(20)/36
GP with 3 partially defined ADFs	29(15)/36	11(7)/36	25(8)/36	26(12)/36

Table 7. Summary of Vanilla GP Trading Strategies Results.

Year	TW-U.S. Model			TW-HK Model		
	μ	σ	*t*	μ	σ	*t*
1997	**1.2295**	0.1292	**2.0409**	1.0815	0.0753	−3.1553
	1.1764	0.0998	0.2638	1.0865	0.1015	−2.1181
	1.1397	0.1171	−1.1759	1.0784	0.0652	−3.8543
	1.1680	0.1305	−0.0846	1.1228	0.1136	−0.4636
	1.2070	0.1086	1.5040	1.1077	0.1016	−1.1834
	1.2001	0.0907	1.4604	1.0784	0.0754	−3.3333
1998	**1.0489**	0.0190	**32.0570**	0.5356	0.0755	−14.5672
	1.0287	0.0355	**14.5564**	0.5102	0.0401	−30.2427
	1.0293	0.0488	**10.6501**	0.5058	0.0200	−61.5146
	1.0437	0.0014	**428.904**	0.5035	0.0102	−121.807
	1.0243	0.0553	**9.0039**	0.5927	0.1645	−5.1336
	1.0457	0.0328	**18.0693**	0.5429	0.1481	−7.2061
1999	**1.2033**	0.0708	1.2920	1.1947	0.0765	−6.8797
	1.2344	0.0877	**2.6265**	1.2110	0.1184	−3.8310
	1.2183	0.0875	**1.8076**	1.2720	0.1424	−1.2689
	1.2146	0.0936	1.5137	1.2728	0.1225	−1.4475
	1.1845	0.1195	0.0600	1.2494	0.1283	−2.1974
	1.2585	0.1160	**2.9160**	1.2419	0.1315	−2.3980
2000	**1.0980**	0.0178	**10.0229**	1.0739	0.0688	−2.4330
	1.0915	0.0162	**9.2525**	1.0808	0.0806	−1.6928
	1.0914	0.0132	**11.3239**	1.0967	0.0842	−0.7770
	1.0779	0.0660	1.3476	1.0540	0.0508	−5.0403
	1.0821	0.0475	**2.2684**	**1.1580**	0.1255	1.6630
	1.0974	0.0336	**5.2471**	**1.1167**	0.1222	0.1965
2001	**0.8227**	0.0313	0.4588	**0.9045**	0.1477	**3.3785**
	0.8538	0.0622	**2.4679**	**0.9207**	0.1558	**3.6642**
	0.8374	0.0284	**2.8263**	**0.8779**	0.1472	**2.5781**
	0.8430	0.0655	1.6014	**0.8370**	0.1385	1.4203
	0.8425	0.0267	**3.8532**	**0.8774**	0.1741	**2.1677**
	0.8509	0.0350	**4.0088**	**0.8755**	0.1569	**2.3513**
2002	0.9516	0.1376	−1.4553	0.8649	0.0440	−17.1482
	0.9677	0.1412	−0.9081	0.8639	0.0326	−23.2644
	0.9008	0.1383	−3.0940	0.8688	0.0441	−16.7151
	0.9416	0.1589	−1.5428	0.9015	0.1078	−5.4767
	0.9203	0.1665	−2.0433	0.8860	0.0662	−9.9759
	1.0118	0.1295	0.5315	0.9195	0.0908	−5.6226

given for each of the 6 out-of-sample periods (1997–2002). Each set contains data obtained from vanilla GP runs using different combinations of population size and number of generations. The average return of 20 trials is μ; the standard deviation is σ; the *t*-statistics is *t*. Those μ values in bold are average returns which are better than the returns of B&H (see Table 5 for B&H returns). Those *t* values in bold indicate the difference between μ and the B&H return is significant at the 5% level.

As shown, the performance of GP strategies varies in different out-of-sample periods. For example, in sequence 5 period, all GP strategies out-perform B&H while in sequence 6 period, B&H gives higher returns in most of the cases. We examined time series in sequence 6 and found that both Taiwan stock and the foreign stock indices (the two most influential trading decision factors) fluctuate widely. For example, during the training period (1999 and 2000), both Taiwan stock and Hang Seng indices declined. During the validation period (2001), the markets gradually improved. However, the markets rallied during the out-of-sample testing period (2002) (see Fig. 7). As a result, the strategies trained using 1999 and 2000 periods and selected based on 2001 period are not able to perform well on 2002 period. This is a shortcoming of all machine learning techniques, including GP.

In contrast, the stock indices for training, validation and out-of-sample periods in sequence 5 have a similar pattern: the stock markets generally went down. Consequently, the strategies evolved on training period were able to perform well on the out-of-sample period. Another interesting observation is that although all markets decline in this period (with Taiwan stock market having the worst decline of 50%) and cause B&H to have low returns (see Table 5), GP strategies were able to make profitable trading decisions. Figure 9 gives two such examples. In Fig. 9(a), the 1 US$ was kept until day 151 and then invested in S&P500 when the index started rising. As a result, it has a return of 1.1131, which is better than holding it until the end of the time period (1.0485). Figure 9(b) gives a different GP strategy, which invested in S&P index stock too early and cause a negative return at the beginning. However, as the index started improving on day 151, the return

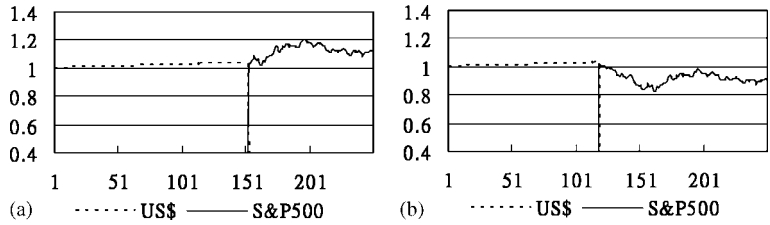


Fig. 9. Capital Flows of Two GP Trading Strategies Applied on Out-of-sample Period of 2001.

became positive. At the end of the period, the return is 0.9036, which is better than the B&H return. One important observation is that none of the GP trading strategies trained in this time sequence entered into Taiwan stock market, the worst asset to invest. This suggests GP strategies have some forecasting abilities in the sense of avoiding money-losing assets all the way to the end of the period.

The transaction frequencies in out-of-sample testing periods are mostly low: no more than 3 times in the whole year. The majority of GP strategies recommend to buy-and-hold on one or two assets. For example, for out-of-sample period 1999, GP trading strategies in TW-JP model either invest in Taiwan stock market or in Japan stock market. The first decision gives a higher return than the second decision does. There are also many strategies give zero transaction: hold the foreign currency all the way to the end of the period. Consequently, most of the GP trading strategies give returns that are close to the returns of B&H (see Tables 7 and 5). This indicates that international short-term financial markets are reasonably efficient during the years between 1997 and 2002.

Overall, the out-of-sample performance of GP trading strategies are not consistently better than that of B&H, an outcome that is consistent with the finding of Allen and Karjalainen (1999) and Wang (2000).

9. ANALYSIS OF GP TRADING STRATEGIES

9.1. Vanilla-GP Trading Strategies

Using both hard and soft constraints to enforce parsimony, the evolved GP trading strategies are not as complex as what we have expected. As mentioned in the previous section, many of them are evaluated into a simple B&H on one or two assets. These strategies either have other options blocked by constant "do nothing" decisions or recommending trading using assets which have no available fund. Overall, the decisions of GP strategies are not difficult to derive, although their financial meaning is challenging to interpret.

We have found one particular GP strategy derived from TW-UK model that provides financially meaningful recommendations. Figure 10 gives the GP tree. Without being very rigorous, the left tree can be interpreted as:

> When UK stock market bulls (suggested by a higher FTSE100 index 250-moving average than US/GBP 250-moving average exchange rate), sell Taiwan stocks to obtain Taiwan currency.

The middle tree can be interpreted as:

> Trade British Ponds with Taiwan Dollars when the exchange rate is less than the 250-day moving average.

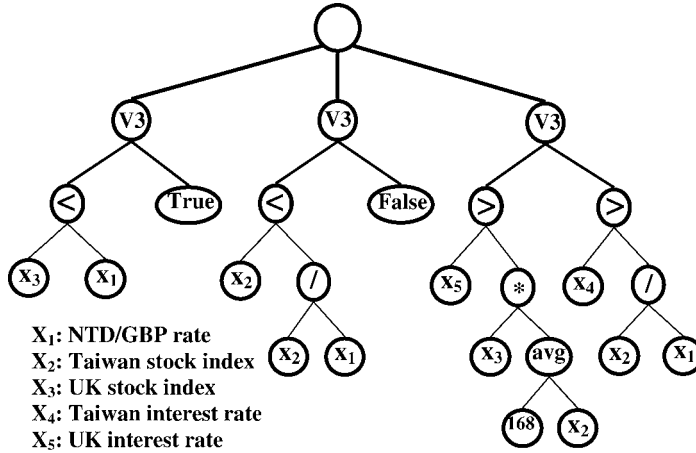


Fig. 10. An Evolved GP Trading Strategy.

The right tree can be interpreted as:

> When Taiwan stock market bulls (suggested by the high Taiwan Stock index), move the funds from UK stock market to UK currency.

The two stock market trading rules recommend allocating funds toward that with indication of higher returns. It advises cashing stocks as a preparation of purchasing foreign stocks, when the foreign stock index looks promising.

9.2. ADF-GP Trading Strategies

ADFs were incorporated for GP to identify possible regularity in profitable trading strategies. However, ADF-GP results are not better than vanilla GP results. We examined those strategies where ADFs were created and evolved by GP and found that most of the ADFs have a constant value of either "*True*" or "*False*." In other words, they are not functions but serve as constants in the trading rules. It is not surprising that this implementation of ADFs give similar returns as the vanilla GP does. As mentioned in Section 5, the time series have been transformed by dividing the daily data with a 250-day moving average. This result indicates that either there is no regularity in the *change of trend* that provides profitable trading or GP is not able to identify such regularity.

Provided with ADFs that are initiated with commonly used technical trading rules, GP still cannot find strategies that give better returns. This indicates that those technical trading rules (and their variations) are not effective on the transformed time series data.

This raises a question about whether the data transformation method used is appropriate for this modeling task or not. In Nikolaev and Iba (2002), they have reported that GP gives different results when the time series are normalized using different data transformation methods. Kassicieh et al. (1998) also reported a similar result when using a GA to make investment decisions. We have normalized the time series by dividing the daily data with 250-day moving average. With such data, GP is searching for patterns in the *change of trend* that give profitable trading. In other words, GP rules can exploit patterns in financial market indices, just like commonly used moving average and filter rules do. When ADFs are incorporated, GP becomes capable of exploiting higher-order complexity, i.e. an ADF gives the first-order pattern while the GP main program calling such ADF defines higher-order complexity (Li & Vitanyi, 1997). We are not certain if higher-order complexity exists in the *change of trend* financial time series. The ADF-GP results do not support this proposition. However, this does not preclude the possibility that such complexity can exist in time series that are normalized using different methods.

We have compared our approach with another work using ADF to find trading strategies in S&P stock index markets and found the author used a different data transformation method in his work: stock indices are divided by 100 while interest rates are divided by 10,000 (Wang, 2000). Similar to our ADF-GP results, Wang's ADF-GP did not discover trading strategies that out-perform B&H in S&P500 spot and future markets. However, his work did not acknowledge ADF-GP is capable of identifying higher-order complexity in the time series. Nor did it mention about the evolved ADF-GP strategies exhibit such complexity. We are inclined to believe that there exist patterns in profitable trading strategies when the time series are applied with appropriate data transformation method. We are currently investigating this hypothesis.

10. CONCLUDING REMARKS

The hot money issue occurred in China has triggered our interest in modeling short-term capital flow in international financial markets. If it is possible to predict such capital inflow and outflow, appropriate measures can be imposed before hand to stabilize global economy. Unfortunately, our finding using GP to simulate a simplified international markets model indicates that such task cannot be accomplished. The devised GP trading strategies do not consistently generate better returns than the buy-and-hold strategy, suggesting that they do not have the ability to predict capital inflow and outflow. Many of the GP strategies recommend the buy-and-hold approach on one or two assets. This indicates that the international

short-term capital markets are reasonably efficient, a finding which is similar to that reported by Allen and Karjalainen (1999) and Wang (2000).

However, many GP strategies are able to forecast Taiwan stock market down time and avoid making futile investments. This indicates that GP has the ability to learn from historical data to make profitable trading decisions. Moreover, during market down time when buy-and-hold gives poor returns, many GP strategies are able to identify opportunities and produce better returns than buy-and-hold.

Our investigation of ADF-GP trading strategies does not support the proposition that profitable strategies contain regularity. Nor does it endorse the idea that commonly used technical trading rules are effective on the *change of trend* time series. This seems to counter our intuitions since it is not uncommon for the real-world technical traders apply a combination of technical trading rules to make trading decisions. We are puzzled by this result and have started looking into reasons that have led to such a conclusion. One issue we have identified is the transformation of time series which might have changed the modeling space and time series correlation. Another aspect is the existence of higher-order complexity in financial time series that can be captured by ADF-GP. We are currently investigating different modular GP techniques (Yu et al., 2004), in addition to ADFs, and different data normalization methods in order to improve our understanding of regularity in profitable trading strategies.

NOTES

1. Of course, as long as the central government doesn't intervene in the foreign exchange market correspondingly, there are always the direct balance of payments effects, be the foreign investment speculative or not.
2. We have also experimented with the setup where each transaction is executed once in each time step. The preliminary results, however, show very little differences from that of executing a transaction 10 times in a time step. This suggests that for these time series data, trading strategies are not sensitive to the amount of capital flow. In other words, under such time series, a trading strategy gives similar return regardless of the amount of fund transferred in each time step.

ACKNOWLEDGMENTS

An earlier version of the paper has been presented at the 2003 International Conference on Artificial Intelligence (IC-AI03), the 9th International Conference on Computing in Economics and Finance (CEF 2003), and the 3rd International Workshop on Computational Intelligence in Economics and Finance (CIEF'2003).

The second author is grateful for the research support from NSC grant No. NSC 90-2415-H-004-018. We would like to thank members of AI-ECON Research Center, particularly Chueh-Yung Tsao and Bin-Tzong Chie, for spending time to discuss financial trading with us during the writing of this paper. We also thank the reviewers for their comments and suggestions and Ingrid Peterson for proof reading this paper.

REFERENCES

Allen, F., & Karjalainen, R. (1999). Using genetic algorithms to find technical trading rules. *Journal of Financial Economics*, *51*(2), 245–271.

Baba, N., Inoue, N., & Asakawa, H. (2000). Utilization of neural networks & GAs for constructing reliable decision support systems to deal stocks. Working Paper of Information Science, Osaka-Kyoiku University, Asahiga-Oka, 4–698–1, Kashiwara City, Osaka Prefecture, Japan.

Brock, W., Lakonishok, J., & LeBaron, B. (1992). Simple technical trading rules and the stochastic properties of stock returns. *Journal of Finance*, *47*, 1731–1764.

Chen, S.-H., & Kuo, T.-W. (2002). Genetic programming: A tutorial with the software simple GP. In: S.-H. Chen (Ed.), *Genetic Algorithms and Genetic Programming in Computational Finance* (pp. 55–77). Kluwer Academic Publishers.

Chen, S.-H., & Kuo, T.-W. (2003). Discovering hidden patterns with genetic programming. In: S.-H. Chen & P. P. Wang (Eds), *Computational Intelligence in Economics and Finance* (pp. 329–347). Springer-Verlag.

Kassicieh, S. K., Paez, T. L., & Vora, G. (1997). Investment decisions using genetic algorithms. In: *Proceedings of the Thirtieth Hawaii International Conference on System Sciences* (pp. 484–490).

Kassicieh, S. K., Paez, T. L., & Vora, G. (1998). Data transformation methods for genetic-algorithm-based investment decisions. In: *Proceedings of the Thirty-First Hawaii International Conference on System Sciences* (pp. 122–127).

Koza, J. (1992). *Genetic programming: On the programming of computers by means of natural selection*. Cambridge, MA: MIT Press.

Koza, J. (1994). *Genetic programming II: Automatic discovery of reusable programs*. Cambridge, MA: MIT Press.

Li, M., & Vitanyi, P. (1997). *An introduction to Kolmogorov complexity and its applications*. Springer Verlag.

Neely, C., & Weller, P. (1999). Technical trading rules in the European monetary system. *Journal of International Money and Finance*, *18*, 429–458.

Neely, C., Weller, P., & Dittmar, R. (1997). Is technical analysis in the foreign exchange market profitable? A genetic programming approach. *Journal of Financial and Quantitative Analysis*, *32*(4), 405–426.

Nikolaev, N., & Iba, H. (2002). Genetic programming of polynomial models for financial forecasting. In: S.-H. Chen (Ed.), *Genetic Algorithms and Genetic Programming in Computational Finance* (pp. 103–123). Kluwer Academic Publishers.

Pesaran, M., & Timmermann, A. (1995). Predictability of stock returns: Robustness and economic significance. *Journal of Finance*, *50*, 1201–1228.

Pring, M. J. (1991). *Technical analysis explained* (2nd ed.). New York: McGraw-Hill.

Seshadri, M. (2003). *Comprehensibility, overfitting and co-evolution in genetic programming for technical trading rules*. Master's Thesis. Worcester Polytechnic Institute.

Syswerda, G. (1991). A study of reproduction in generational and steady-state genetic algorithms. In: G. J. E. Rawlins (Ed.), *Foundations of Genetic Algorithms* (pp. 94–112). Morgan Kaufmann.

Wang, J. (2000). Trading and hedging in S&P 500 spot and futures markets using genetic programming. *Journal of Futures Markets*, *20*(10), 911–942.

Yu, T., & Bentley, P. (1998). Methods to evolve legal phenotypes. In: *Proceedings of the Fifth International Conference on Parallel Problem Solving From Nature* (pp. 911–942). Springer.

Yu, T., Chen, S.-H., & Kuo, T.-W. (2004). Discovering financial technical trading rules using genetic programming with lambda abstraction. In: U.-M. O'Reilly, T. Yu, R. Riolo & B. Worzel (Eds), *Genetic Programming Theory and Practice* (Vol. II, pp. 11–30). Kluwer Academic.

TOOLS FOR NON-LINEAR TIME SERIES FORECASTING IN ECONOMICS – AN EMPIRICAL COMPARISON OF REGIME SWITCHING VECTOR AUTOREGRESSIVE MODELS AND RECURRENT NEURAL NETWORKS

Jane M. Binner, Thomas Elger, Birger Nilsson
and Jonathan A. Tepper

ABSTRACT

The purpose of this study is to contrast the forecasting performance of two non-linear models, a regime-switching vector autoregressive model (RS-VAR) and a recurrent neural network (RNN), to that of a linear benchmark VAR model. Our specific forecasting experiment is U.K. inflation and we utilize monthly data from 1969 to 2003. The RS-VAR and the RNN perform approximately on par over both monthly and annual forecast horizons. Both non-linear models perform significantly better than the VAR model.

Applications of Artificial Intelligence in Finance and Economics
Advances in Econometrics, Volume 19, 71–91

ISSN: 0731-9053/doi:10.1016/S0731-9053(04)19003-8

1. INTRODUCTION

Non-linear models for economics and time series modeling have gained in popularity over recent years. The main reason for this is the failure of linear models to capture non-linear dynamic relationships embedded in real-world data. Econometric developments in combination with increases in computing power have further spurred the use of non-linear models. The purpose of this study is to contrast the forecasting performance of two non-linear models, a regime-switching (RS) vector autoregressive model (VAR) and a recurrent neural network (RNN), to that of a linear benchmark VAR model. These models belong to different classes of non-linear models that are both econometrically challenging and therefore rarely compared.

Our specific forecasting experiment is U.K. inflation over the period 1969–2003. For this purpose, we obtain monthly observations of the retail price index, M0 and industrial production. The first part of the data set is used for estimation (training). The last five years of the data is used for out-of-sample forecasting and evaluation of the different models. There are three main motives for choosing to study U.K. inflation using this set of variables. Firstly, the amount of available data must be considered large. Our full sample contains over 400 observations. Macroeconomists are fortunate if quarterly data for, say, 20–30 years is available. Secondly, visual inspection of the data indicates that the use of a linear model may be inappropriate. Over the full sample, we can identify long periods with high inflation and long periods with low inflation. The latter is particularly evident for the 1990s, or, more specifically, the "post-ERM" period. Finally, as evident in numerous papers and central bank reports, many people are interested in inflation forecasts. For them, this forecasting experiment may be interesting in its own right.

In the regime switching framework, the underlying idea is to allow for endogenous switches between different data generating processes at different points in time, where the switches are governed by an underlying discrete state Markov process. Markov-switching regression models were introduced in economics by Goldfeld and Quandt (1973), but time-series applications were not considered until the seminal papers by Hamilton (1988, 1989). Regime-switching models are inherently non-linear because the mean, as well as higher moments, are non-linear functions of the current state of the Markov process. The different states of the underlying Markov process have in economic applications, for example, represented periods of low and high exchange rate volatility (Klaassen, 2002), different level and volatility of real interest rates (Garcia & Perron, 1996), booms and slumps (Hamilton, 1989), and low and high volatility and correlation in stock markets (Ang & Bekaert, 2002). Particular applications to forecasting in the

RS-VAR framework include Krolzig (2004), who models output and employment growth and Blix (1999), who studies inflation in a trivariate RS-VAR system.

Artificial neural networks (ANNs) (Cheng & Titterington, 1994; Haykin, 1999; Rumelhart et al., 1986), on the other hand, consist of simple interacting processing units that are arranged in arbitrary layers with variable patterns of interconnectedness. Knowledge is represented as connection strengths (or weights) between connected units. Each processing unit spreads its activation to connected units after combining and processing its input using some linear or non-linear activation function. Learning occurs when general recursive rules are applied to adapt these weights in order to produce desired output responses.

ANNs are becoming increasingly popular in economics and are used in a large variety of modeling and forecasting problems. The main reason for this increased popularity is that these models have been shown to be able to approximate any non-linear function arbitrarily close (Cybenko, 1989; Hornik, 1991). Hence when applied to a time series which is characterized by truly non-linear dynamic relationships, the ANN will provide a global approximation to this unknown non-linear relationship. Previous studies unveil the applicability of ANNs to modeling and forecasting in economics. Applications include returns forecasting (Gençay, 1996; Gençay & Stengos, 1998; Haefke & Helmenstein, 1996a, b), option pricing (Qi & Maddala, 1995), exchange rates (Franses & van Griensven, 1998; Franses & van Homelen, 1998; Gençay, 1999; Giles et al., 2001; Kuan & Liu, 1995; Tenti, 1996), interest rates (Swanson & White, 1995) and inflation (Binner et al., 2002; Moshiri et al., 1999; Stock & Watson, 1999).[1] Due to the inherent ability of ANNs with recurrent connections (i.e. RNNs) to implicitly learn the non-linear temporal dynamics of sequential time series data without recourse to additional temporally-dependent external memory mechanisms, this work uses and evaluates RNN models.

The paper is organized as follows. Section two introduces the VAR model, the RS-VAR model and the RNN. Section three contains a brief description of the data together with a discussion of estimation (training) results. Section four presents the inflation forecast evaluation results. Section 5 concludes the paper.

2. NON-LINEAR MODELS

This section introduces the VAR model, the RS-VAR model and the RNN. The *raison d'être* for considering the latter models is an *a priori* belief that linear models will fair badly in our forecasting experiment. It is therefore logical to include a linear model as a benchmark model in our analysis. A natural choice is the VAR model, which is commonly used in macroeconomic forecasting experiments.

In the VAR model, the dynamics of $\mathbf{x}_t$, a k-dimensional vector of dependent variables at time t, is governed by the following pth order autoregressive process:[2]

$$\mathbf{x}_t = \alpha_0 + \mathbf{A}_1\mathbf{x}_{t-1} + \cdots + \mathbf{A}_p\mathbf{x}_{t-p} + \varepsilon_t, \tag{1}$$

where α_0 is a vector of intercepts, $\mathbf{A}_\ell$, $\ell = 1, \ldots, p$ are $k \times k$ coefficient matrices and ε_t is a white-noise distributed disturbance vector. Estimates of the coefficient matrices are obtained using ordinary least squares. The model is said to be stable if its reverse characteristic polynomial has no roots in and on the complex unit circle.

For the VAR model, a conditional $t + 1$ forecast of $\boldsymbol{x}$ is obtained from:

$$E_t(\mathbf{x}_{t+1}) = \alpha_0 + \mathbf{A}_1\mathbf{x}_t + \cdots + \mathbf{A}_p\mathbf{x}_{t-p+1}, \tag{2}$$

for each t using the information set $\mathbf{X}_t = \{\mathbf{x}_t, \mathbf{x}_{t-1}, \ldots\}$. The conditional $t + 1$ forecast in (2) is a special case of the conditional dynamic $t + \tau$ forecast where *forecasted* values of $\mathbf{x}$ should be used for certain lags (depending on the forecast horizon and the number of included lags).

2.1. The Regime Switching VAR Model

We follow the common practice in the economic regime switching literature and assume that a discrete time-homogenous s-state first order Markov process governs the endogenous switches between the regimes. This assumption implies that the probability of a switch between different regimes is described by a constant transition matrix, that there is a different VAR model in each regime and that only the most recent state of the Markov-process influences the transition probabilities. We also assume that $\mathbf{x}_t$ can be modeled as a discrete mixture of k-variate Gaussian distributions. This assumption implies that $\mathbf{x}_t$ is Normal distributed conditional on the prevailing regime and the information set $\mathbf{X}_{t-1}$ Finally, we restrict the variance-covariance matrix in each regime to be constant.

Taken together, if the prevailing regime at time t is $j \in \{1, \ldots, s\}$, then the VAR process is:

$$\mathbf{x}_t = \alpha_0^{(j)} + \mathbf{A}_1^{(j)}\mathbf{x}_{t-1} + \cdots + \mathbf{A}_{p(j)}^{(j)}\mathbf{x}_{t-p(j)} + \varepsilon_t^{(j)}, \tag{3}$$

where $p(j)$ is the order of the VAR model, i.e. the number of included lags in regime j. Further, the matrix of transition probabilities, $\mathbf{P} = \{p_{ij}\}$, where i, $j = 1,\ldots, s$ is determined by the equations:

$$p_{ij} = \Pr[S_t = j | S_{t-1} = i], \tag{4}$$

where the state variable S_t denotes the regime prevailing at time t.

This Markov switching model can be estimated via maximum likelihood as described in Hamilton (1994). A well-known problem of regime switching models is that the likelihood surface is multimodal. For this reason, a strategy of how to be able to report the global maximum should be advised. Our solution to this problem is to use simulated annealing to maximize the likelihood function. Simulated annealing is a derivative-free stochastic search algorithm that, in contrast to conventional gradient based algorithms, is able to escape from local maxima and hence is very well suited for estimation of regime switching models.[3]

As the regimes are unobservable, S_t can be regarded as missing data. However, the probability that a given observation belongs to a particular regime can be computed. From this information, we can construct an optimal forecast probability that the next observation belongs to a particular regime.

Let $\Pr_t\ (S_t = j)$ denote the updated (filtered) probability that $S_t = j$, given the information set $\mathbf{X}_t$. The forecast probabilities for time $t+1$, given the information available at time t, are denoted $\Pr_t\ (S_{t+1} = j)$ Following Hamilton (1994), let $\xi_{t|t}$ denote the vector of updated probabilities and $\xi_{t+1|t}$ the vector of forecast probabilities. The time t likelihood function value is obtained as a by-product from the following iterations to calculate the optimal forecast probabilities:

$$\xi_{t|t} = \frac{\xi_{t|t-1} \circ \varphi_t}{\mathbf{1}'(\xi_{t|t-1} \circ \varphi_t)}, \tag{5a}$$

$$\xi_{t+1|t} = \mathbf{P}'\xi_{t|t}, \tag{5b}$$

where φ_t is the vector of conditional densities, $\mathbf{1}$ is a unit column vector and $\circ$ denotes (vector) element-by-element multiplication. This filter is proposed in Hamilton (1989) and can be thought of as a non-linear Kalman filter, where the probabilities are the state variables. The filter outputs an optimal inference about the predicted and updated probabilities through one set of prediction equations and another set of updating equations. The time t likelihood function is given by the denominator in (5a). In words, the likelihood function is the weighted sum of the conditional densities, with weights given by the forecast probabilities.

2.1.1. Forecasting Method

For the RS-VAR model, the conditional $t+1$ forecast of $\mathbf{x}$ is the weighted average of the s different forecasted within regime means, with weights given by the probability of each regime to prevail in the next period, i.e.:

$$E_t(\mathbf{x}_{t+1}) = \sum_{j=1}^{s}\sum_{i=1}^{s} \Pr_t(S_t = j) p_{ij} \left(\alpha_0^{(j)} + \sum_{\ell=1}^{p(j)} \mathbf{A}_\ell^{(j)} \mathbf{x}_{t-\ell+1} \right), \tag{6}$$

for each t using the information set $\mathbf{X}_t = \{\mathbf{x}_t, \mathbf{x}_{t-1}, \ldots\}$. In an s-regime model there are S^τ possible outcomes for each dependent variable τ periods ahead and S^τ associated probabilities. Hence, the $t + \tau$ forecast of $\mathbf{x}$ is calculated as the probability weighted average by traversing the non-recombining tree generated by the underlying Markov-chain along all possible paths. Note that the probability trees should be multiplied by the s updated probabilities according to (6). This reflects the uncertainty of the prevailing regime today, or, econometrically, the fact that the Markov chain is unobserved even *ex post*.[4]

2.2. *Recurrent Neural Networks*

Recurrent neural networks based on gradient descent learning[5] are typically adaptations of the traditional feed-forward multi-layered perceptron (FF-MLP) trained with the back-propagation algorithm[6] (Rumelhart et al., 1986). This particular class of RNN extends the FF-MLP architecture to include recurrent connections that allow network activations to feedback (as internal input at subsequent time steps) to units within the same or preceding layer(s). Units that receive such feedback values are referred to as *context* or *state* units.

This internal memory enables the RNN to construct internal representations of temporal order and dependencies since the temporal structure embedded within the data will be encoded within the spatial structure of the RNN. This is due to the sequence of weights to each unit connected to the input layer being convolved with different sequences of input examples (Haykin, 1999). Assuming non-linear activation functions are used, the universal function approximation properties of FF-MLPs naturally extends to RNNs.

The pattern of recurrent connectivity strongly influences the computational power of RNNs where certain classes of RNNs have been shown to theoretically simulate Turing machines (Siegelmann & Sontag, 1991). More specifically, it has been shown that RNNs employing recurrent connections from the output layer back to the input layer (see Jordan, 1986) are analogous to infinite impulse response filters (IIRs) (Khan & Unal, 1995). Likewise, those RNNs that contain recurrent connections from the hidden layer back to the input layer, such as Elman's Simple Recurrent Networks (SRNs) (Elman, 1990) are similar in computational power to Hidden Markov Models (HMMs) (Lee, 1989). More generally, RNNs are considered dynamical systems that can represent at least auto-regressive with moving average (ARMA) estimators (Connor et al., 1994).

RNNs can be expressed generally as (modified from Chappelier et al., 2000):

$$\mathbf{r}_t = f(\mathbf{r}_{t-1}, \mathbf{x}_t, t, \Theta_t^f), \tag{7a}$$

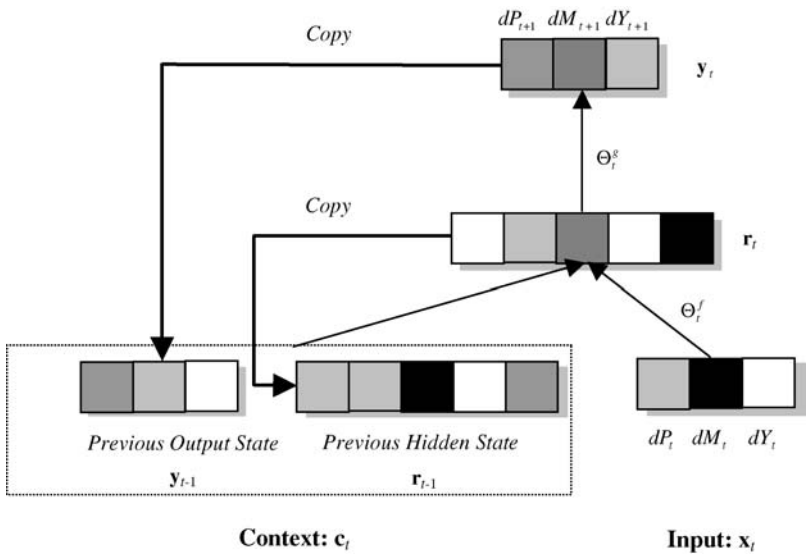


Fig. 1. The Jordan-Elman Hybrid Network Architecture.

$$\mathbf{y}_t = g(\mathbf{r}_t, t, \Theta_t^g), \tag{7b}$$

where **r** represents the context vector (recurrent variables); **y** refers to the activation vector on the output layer (regressands); **x** is the input vector (input variables or regressors); t refers to the current time step; Θ_t^f refers to the set of weights connecting the input layer to the hidden layer at time t; and Θ_t^g refers to the set of weights connecting the hidden layer to the output layer at time t; finally, functions f and g represent the activation vectors from the hidden and output layers respectively.

For the purposes of this paper, we use an RNN architecture that combines the relative strengths of the Jordan network (Jordan, 1986) with Elman's SRN (Elman, 1990) to form a hybrid RNN that feeds both the hidden unit activations and output unit activations back to the context units of the input layer. This is shown in Fig. 1. To reduce complexity and the number of parameters, our model does not utilize a recurrent input layer or self-loops.

We also use a common extension of the backpropagation algorithm for training RNNs called *backpropagation-through-time* (BPTT) (Rumelhart et al., 1986; Werbos, 1990; Williams & Peng, 1990), which has proven more powerful than standard backpropagation for training RNNs such as Jordan and Elman networks. BPTT works under the assumption that for every recurrent MLP network, a feedforward MLP with identical behavior can be obtained by "unfolding the

network over time," i.e. a recurrent network state at time t can be viewed as though it was obtained from the tth layer output of a corresponding FF-MLP network with T layers (Rumelhart et al., 1986), where T is the length of the sequence. When at the end of a sequence or h patterns have been propagated through the network, the external error is calculated and the errors and local gradients are backpropagated through the network from t until the initial time step. During this back-error propagation phase, an additional error term for recurrent connections can be introduced to emphasize unit-error contributions from subsequent time steps. Each individual weight change is then calculated as in standard backpropagation but applied and summed across all time steps resulting in one large weight change calculation for each weight. The weights are then adjusted as normal. In the case of *epoch-wise* BPTT (Williams & Peng, 1990) the context units are reset to some initial value at the beginning of each sequence. Due to these reset operations, we implement a discrete-time RNN as opposed to a continually running RNN.

The equation for the hybrid Jordan-Elman RNN used in this work can be expressed generally as (modified from Eqs (7a) and (7b) above):

$$\mathbf{r}_t = f(\mathbf{c}_t, \mathbf{x}_t, \Theta_t^f), \tag{8a}$$

$$\mathbf{y}_t = g(\mathbf{r}_t, \Theta_t^g), \tag{8b}$$

where $\mathbf{r}_t$ now represents the hidden unit activation vector at time t; $\mathbf{c}_t$ refers to the concatenation of the previous hidden state vector, $\mathbf{r}_{t-1}$, and the previous external output response vector, $\mathbf{y}_{t-1}$; $\mathbf{y}_t$ refers to the external output activation vector at time t; as illustrated in Fig. 1, x_t refers to the external vector of input variables at time t; Θ_t^f refers to the set of weights connecting the input layer to the hidden layer; and Θ_t^g refers to the set of weights connecting the hidden layer to the output layer; finally, functions, as before, f and g represent the activation vectors from the hidden and output layers, respectively.

2.2.1. Forecasting Method

When applying the selected RNN method to our forecasting task we follow the common practice of: (i) normalizing the input variables to accelerate the learning process by avoiding time consuming trajectories across the error surface caused by groups of successive observations sharing the same sign and thus direction; (ii) generating the training (or estimation) data from some in-sample using a sliding window of a pre-determined time-lag; (iii) using the volume of training data to set an upper-limit on the number of allowable free parameters (weights) and thus to constrain network size in a way that will reduce over-fitting and subsequently increase generalization ability; (iv) implementing an appropriate training regime with a learning rate schedule that encourages fast, stable and convergent learning;

(v) defining a suitable stopping criteria; and (vi) using the sliding window technique to generate forecasts of the required forecast horizon.

The fundamental purpose of the training phase is for the RNN to learn to forecast (on its output vector $\mathbf{y}$) each value of the dependent variables in $\mathbf{x}$ at time $t+1$ given a sequence of input vectors $\mathbf{X}_t = \{\mathbf{x}_t, \mathbf{x}_{t-1}, \ldots\}$.

We use the hyperbolic tangent function for calculating the activation level for all processing units (hidden and output units) of the network. The hyperbolic tangent function is simply the logistic function rescaled and biased with user defined constants. For training the RNNs, we employ a "*search and converge*" learning rate regime as defined by Darken and Moody (1991):

$$\eta = \frac{\eta_0}{n/(N/2) + c_1/\max(1, (c_1 - (\max(0, c_1(n - c_2 N))/(1 - c_2)N)))} \tag{9}$$

where η is the learning rate, η_0 is the initial learning rate which we set to 0.01, N is the total number of training epochs, n refers to the current training epoch, c_1 and c_2 are user defined constants. This provides an effective time varying learning rate constant that guarantees convergence of stochastic approximation algorithms and has proven effective for training RNNs to learn temporal problems (see Giles et al., 2001; Lawrence et al., 2000). Motivated by the impressive RNN results reported by Lawrence et al. (2000), we use constant values of 50 and 0.65 for c_1 and c_2 respectively. The momentum term constant is fixed at 0.9 due to the low learning rates. Note that since our model is a discrete-time RNN, the context units ($\mathbf{c}_t$) must be reset to some initial value after each $\mathbf{x}_{t+1}$ forecast during training (estimation) and testing (forecast evaluation).

For generating dynamic forecasts of each dependent variable τ periods ahead, we maintain two input streams: stream A containing all real observations found in the forecast evaluation (test) set and stream B consisting of the first $t - \tau$ real observations found in the training set, where t initially refers to the first observation to be forecast in the evaluation (test) sample. Stream B will grow dynamically over time as it accommodates the RNNs $t+1$ forecasts. We first initialize the RNNs context units and then using the standard sliding window technique we pass the first τ observations in stream B through the network, generating dynamic observations at each step and appending them to stream B. Clearly, the first $t + \tau$ dynamic forecast cannot be generated until the last real observation in stream B is in the first position of the sliding window (i.e. becomes $\mathbf{x}_t$). From thereon both streams are synchronously indexed. Dynamic $t + \tau$ forecasts are generated when the window of input observations becomes $\hat{\mathbf{X}}_t = \{\mathbf{x}_t, \mathbf{x}_{t+1} = \mathbf{y}_t, \ldots, \mathbf{x}_{t+\tau-1} = \mathbf{y}_{t+\tau-2}\}$, where $\mathbf{x}_t$ is always a real observation extracted from stream A. Note that each element of $\hat{\mathbf{X}}_t$ is presented sequentially to the RNN. We continue processing in this way until all observations in stream A have been represented in $\mathbf{x}_t$.

3. DATA AND ESTIMATION RESULTS

We obtain a monthly data set that covers June 1969–October 2003 from EcoWin, yielding a total of 413 observations. The three time-series we obtain are: (i) the Retail Price Index (P); (ii) M0 (M); and (iii) Industrial Production (Y).[7] Each of these series is logarithmically transformed and differenced, so that $\mathbf{x}_t' = (dP_t, dM_t, dY_t)$. For the RNN, all observations for each of the dependent input variables are

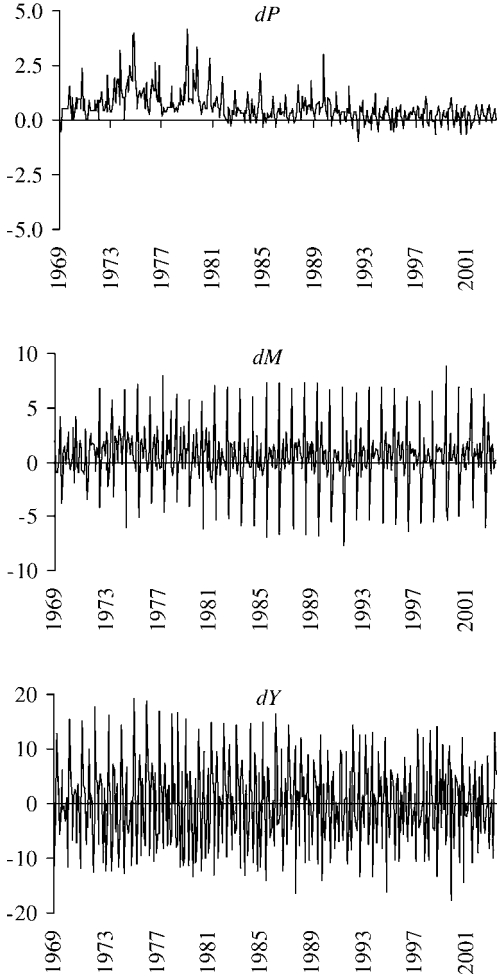


Fig. 2. The Log-differenced Time Series.

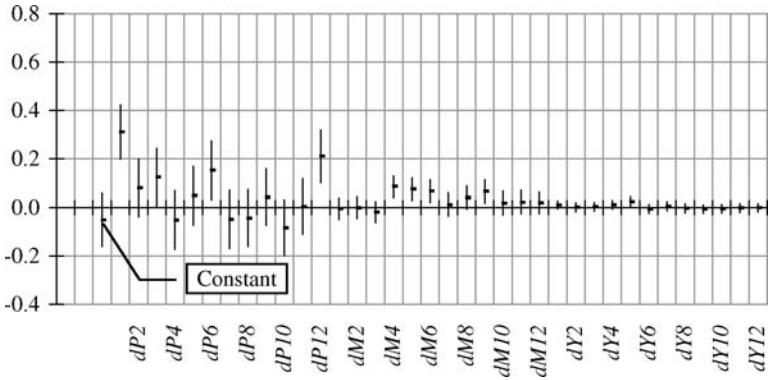


Fig. 3. Estimated Coefficients for Price Equation in VAR-Model with Two Standard Error Bands.

independently normalized such that each variable in **x** has a zero mean and a unit standard deviation. The data is shown in Fig. 2.

The data up to October 1998 is used for estimation (training). The last five years of the data (November 1998–October 2003; 60 observations) is used for forecast evaluation. To limit the scope of our study somewhat and to facilitate comparison of forecasts from the different models, we use a fixed lag-length of 12 for our two vector autoregressive models and for the RNNs.

Before turning to a formal comparison of the forecasting performance, we briefly present the estimation results for the different models. The estimated dynamic structure of the price equation in the VAR(12) model is visualized in Fig. 3. The parameters are depicted jointly with their two standard error bands.

As can be seen, the estimated coefficient for the constant term is insignificant. The first lag of inflation is (relative to remaining coefficients) large and highly significant. We also find various higher lags of inflation as well as changes in M0 and industrial production to be significant. Finally, it can be verified that the estimated model satisfies the theoretical stability restrictions.

3.1. RS-VAR

In the RS-VAR framework, we find that allowing for a different intercept in each regime, while leaving the dynamic structure unchanged across regimes, provides the best models for inflation forecasting purposes. When allowing for a different dynamic structure in different regimes, the forecasting performance deteriorates significantly, which may be interpreted as a classical case of in-sample over-fitting. This result is in accordance with Krolzig (2004), who argues that forecast errors of

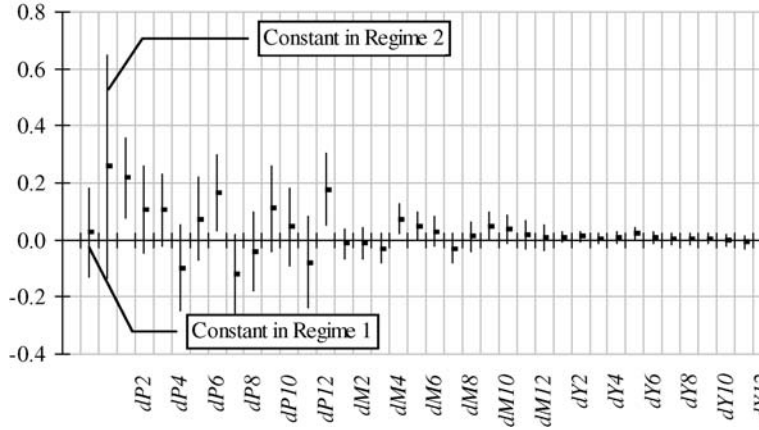


Fig. 4. Estimated Coefficients for Price Equation in RS-VAR-Model with Two Standard Error Bands.

economic time series from linear models are mainly due to shifts in the level or drift. Hence, we only report estimation (and forecasting) results for a two-regime RS-VAR(12), with a different intercept and the same dynamic structure in each regime.

The estimated dynamic structure of the price equation in the RS-VAR model can be found in Fig. 4. The point estimate of the intercept in the second regime is obviously much higher than in the first regime, although both are statistically insignificant. Turning to the lag structure, we find that the first, sixth and twelfth lag of inflation is significant as well as the fourth, fifth and ninth lag of changes in M0 and the fifth lag of industrial production. This pattern is very similar to the one found for the linear VAR model (Fig. 3).

The estimated probabilities for remaining in each regime are presented in Table 1. The expected duration for each regime is determined by $(1 - p_{jj})^{-1}$ where p_{jj} is the probability of remaining in regime j once there. For the preferred two-regime model, both regimes are relatively persistent, with expected durations of about 47 and 23 months, for regime one and regime two, respectively.

Table 1. Estimates of Transition Probabilities and Volatilities.

	P_{jj}	σ_{jdP}	σ_{jdM}	σ_{jdY}
Regime 1	0.9787 (0.0130)	0.2295 (0.0138)	0.6614 (0.0457)	2.2302 (0.1551)
Regime 2	0.9567 (0.0296)	0.6680 (0.0443)	1.3194 (0.1114)	3.7160 (0.3669)

Note: Standard errors in parentheses.

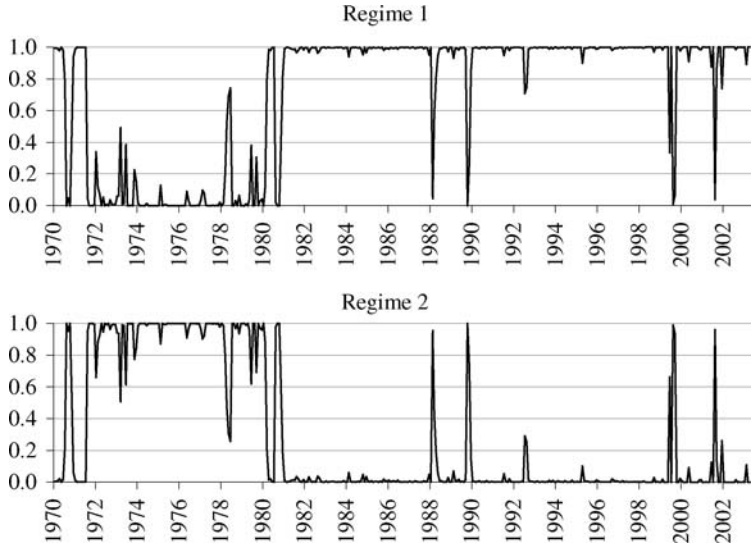


Fig. 5. Filtered Probabilities of Regimes in RS-VAR-Model.

Time-series plots of the updated probabilities can be found in Fig. 5. The probabilities attached to regime one and two vary (in principle) between zero and one. As is evident from Table 1, volatility is higher in the second regime for both inflation, money growth and growth in industrial production. This is consistent with the time-series plots of the updated probabilities, in which it can be seen that the second regime is dominating up until the beginning of the 1980s, while the first regime has dominated since then. In other words, the second regime seems to capture the period of high inflation and high inflation volatility during the first, say, 12 years of the sample, while the first regime captures the period of lower inflation and lower inflation volatility during the next 20 years. However, one must remember that we simultaneously model inflation, money growth and growth in industrial production, which means that the regimes must be interpreted as different regimes of the economy, rather than different regimes of inflation (only).

3.2. RNN

All RNN models contain three input units (to present $\mathbf{x}_t$ to the processing units) and three output units (to forecast $\mathbf{x}_{t+1}$). The number of hidden units is empirically established.

Using the sliding window technique to generate training sequences from the in-sample observations we obtain a training set consisting of the following total number of patterns:

$$z = \frac{q - m}{m}, \tag{10}$$

where z refers to the total number of input pairs (a pair refers to an input vector $\mathbf{x}$ and an output vector encoding the desired response, i.e. values of $\mathbf{x}$ at the next time period), q is the total number of observations in the time series and m is the window size. The resulting number of training sequences (collections of training pairs) is simply:

$$v = \frac{z}{m}. \tag{11}$$

For our purposes, $q = 352$ and $m = 12$, thus $z = 4080$ and $v = 340$.

The number of patterns generated (z) form a basis for our upper limit constraint on the number of allowable parameters (weights) and thus on the number of hidden units allowed. The architecture is therefore fixed in accordance with the data and with the complexity of the underlying problem. We expect over-fitting to occur as the number of free parameters approaches z, i.e. 4080 (e.g. 60 hidden units or above).

For all training (or estimation) experiments weight initializations were in the range of $[-0.1, 0.1]$. Our stopping condition is a function of the root mean squared error (RMSE) and number of training cycles performed. A training cycle is where the network has processed all training patterns and sequences once (a single pass of the whole data through the network) and is referred to as an epoch. An RNN is stopped training when either the number of epochs reaches 2000 or the rate of change of the RMSE is sufficiently small, whichever comes first. During each epoch, successive sequences are randomly selected to encourage a wider search of the weight space. The standard summed squared error (quadratic cost) function was used for all error calculations.

After performing a number of training experiments with various hidden unit configurations (below 60) we found that an RNN configuration with 50 hidden units, RNN(50), and 3003 free parameters (26% fewer than available training cases) provides an optimum fit for the in-sample data with respect to subsequent dP_{t+1} performance on the evaluation forecast data (generalization). After 2000 epochs, the RNN(50) obtained an RMSE of 5.0093 for the whole in-sample data set and an RMSE of 0.2826 for the dP observations.

We also assessed an RNN with more free parameters than there were training cases to confirm our intuitions that classic over-fitting would naturally occur. After 2000 epochs, an RNN with 100 hidden units, RNN(100), and 11,003 free

parameters (170% more than available training cases) obtained a tighter fit of the whole in-sample data set with an RMSE of 4.2246. As expected, it obtained a tighter fit (approximately 26% closer) for the in-sample dP observations with an RMSE of 0.2102 at the expense of yielding poorer generalization performance for dP_{t+1} forecasts.

4. FORECAST EVALUATION

Turning to a formal comparison of the forecasts, we use three common forecast evaluation criteria; mean errors (ME), root mean squared errors (RMSE), and mean absolute errors (MAE). ME can give an indication as to whether the forecast is biased. RMSE is the most frequently used measure, while MAE is known to be less sensitive to outliers. Let:

$$\text{ME} = \frac{1}{K} \sum_{t=1998:10+\tau}^{2003:10} (E_{t-\tau}[dP_t] - dP_t), \tag{12a}$$

$$\text{MAE} = \frac{1}{K} \sum_{t=1998:10+\tau}^{2003:10} |E_{t-\tau}[dP_t] - dP_t|, \tag{12b}$$

$$\text{RMSE} = \left[\frac{1}{K} \sum_{t=1998:10+\tau}^{2003:10} (E_{t-\tau}[dP_t] - dP_t)^2 \right]^{1/2}, \tag{12c}$$

where K is the total number of out-of-sample forecasts and τ is the forecast horizon. In our specific study $K = 60$ or 49, depending on whether τ equals one or twelve. We complement the raw measures in (12b–c) by presenting ratios of the MAE and RMSE for each model to that of the best performing model.

4.1. One-Month Ahead Forecasts

We present the $t + 1$ forecasts as a 2×2-panel in Fig. 6. Forecasts (black line) are compared to actual inflation (gray line) in each panel.

We immediately note that the VAR-model performs reasonably well in comparison with the non-linear models. It tends, however to overshoot large increases in monthly inflation on occasions. The forecasts from the RS-VAR model are quite similar to those obtained from the VAR-model. We note, however, that there are fewer tendencies to overshoot large changes in inflation compared to the

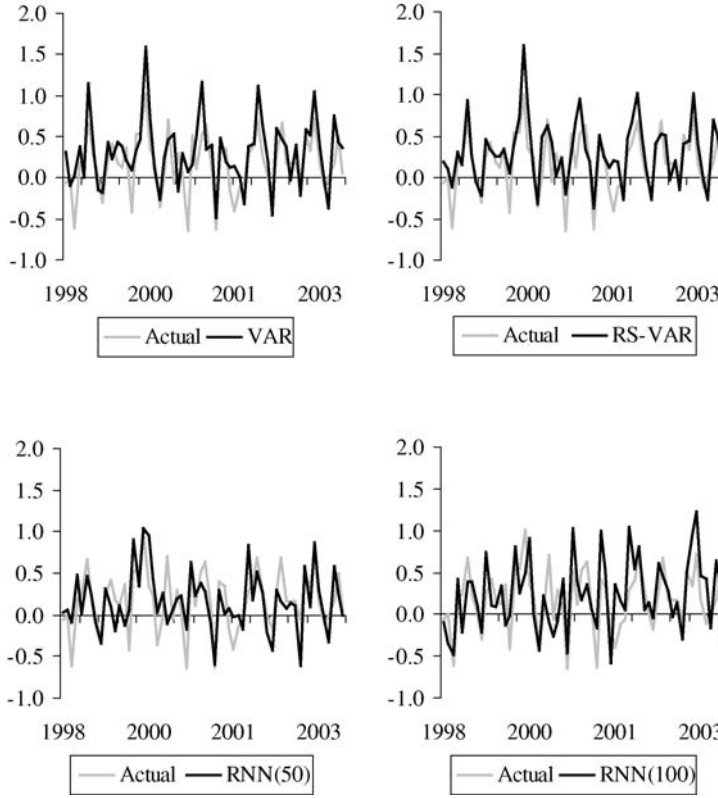


Fig. 6. Forecasted ($t + 1$) and Actual Values of Inflation.

standard VAR. Although the pattern of the forecasts obtained from the RNN(50) and RNN(100) models differs from the forecasts obtained by the two VAR-models, these models appear to give good forecasts of inflation as well. An interesting observation is that all models poorly capture the fall in the price level in January and July up to 2001.

As is evident from the calculated ME in Table 2, all models but RNN(50) are slightly biased upwards. For the two VAR-models and the RNN(100) model, the bias is approximately 0.1%, but substantially lower for the RNN(50) model. Turning next to RMSE and MAE, both criteria rank the different models under evaluation similarly. The RNN(50) model and the RS-VAR model are the best models. They perform approximately as well, with the RS-VAR model yielding slightly lower values for both MAE as well as RMSE. The RNN(100) model is

Table 2. Evaluation Criteria for $t + 1$ Forecasts.

	VAR	RS-VAR	RNN(50)	RNN (100)
ME	0.1285	0.1375	−0.0097	0.0616
MAE	0.2712	0.2245	0.2267	0.2987
RMSE	0.3186	0.2752	0.2889	0.3533
MAE-ratio	120.80%	100.00%	100.98%	133.03%
RMSE-ratio	115.77%	100.00%	104.97%	128.39%

the worst performing model, yielding higher values of both MAE and RMSE than are obtained from the linear VAR model. In fact, looking at ratios of the forecast-evaluation criterion, it performs up to 33% worse than the best performing RS-VAR model.

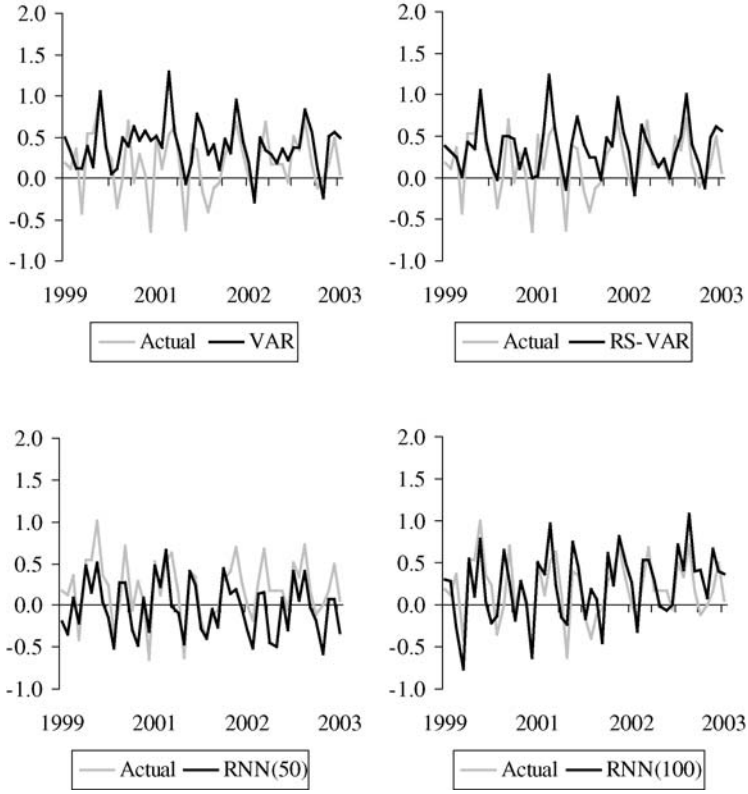


Fig. 7. Forecasted ($t + 12$) and Actual Values of Inflation.

Table 3. Evaluation Criteria for $t + 12$ Forecasts.

	VAR	RS-VAR	RNN(50)	RNN (100)
ME	0.1963	0.1739	−0.2160	0.0428
MAE	0.2984	0.2579	0.2771	0.2498
RMSE	0.3761	0.3177	0.3332	0.3018
MAE-ratio	119.44%	103.25%	110.91%	100.00%
RMSE-ratio	124.63%	105.28%	110.41%	100.00%

4.2. One-Year Ahead Forecasts

We present the $t + 12$ forecasts as a 2×2-panel in Fig. 7. As before, forecasts (black line) are compared to actual inflation (gray line) in each panel.

A noticeable difference between the two VAR models and the two RNNs is that forecasts from the VAR models appear to be biased upwards. This tendency seems to be more pronounced compared with the one-month ahead forecasts.

The fact that the VAR-models overshoot actual inflation for the one-year ahead forecasts is confirmed by the ME criterion presented in Table 3. The RNN(50) undershoots actual inflation. The RNN(100) still overshoots inflation, although slightly less than was the case on a one-month ahead basis. The RNN(100) model is now, judging from MAE and RMSE the best forecaster. The second best is the RS-VAR model followed by the RNN(50) model. The VAR model performs significantly worse than the other models.

Finally, we observed an interesting behavior when comparing RNN(50) and RNN(100) dP_{t+12} dynamic forecasts. Contrary to our intuitions, we found that the RNN(100) yielded superior performance with an RMSE of 0.3018, 9% better than that generated by the RNN(50) (i.e. an RMSE of 0.3332). This is a good example of the unpredictable nature of RNNs and further research is required to assess which factors within its complex internal dynamics is causing such perturbations in expected behavior. We are particularly interested in the approach taken by Lawrence et al. (2000) and Giles et al. (2001), for extracting discrete finite state automata from RNNs to help understand its behavioral patterns and intend to pursue this further.

5. CONCLUSIONS

In this study, we have compared the forecasting performance of two non-linear models, a RS-VAR model and a RNN model, with that of a benchmark linear VAR model. Our specific forecast experiment is U.K. inflation. We find that the

RNN model and RS-VAR model outperform the VAR model for both monthly and annual forecast horizons. The RS-VAR and the RNN perform approximately on par over both forecast horizons. For the RS-VAR model, we find that imposing the restriction that only the intercept is allowed to vary across regimes provides the best forecasts. For the RNN-model, the forecasting performance depends on the number of hidden units and thus free parameters included.

NOTES

1. For a detailed account of vector-based machine learning models applied to financial prediction, see Shadbolt and Taylor (2002) and for hybrid models, see Kovalerchuk and Vityaev (2000).
2. For thorough textbook treatments of VAR-models, see for example Enders (1995), Hamilton (1994) and Lütkepohl (1991).
3. Relevant references to simulated annealing include Corona et al. (1987), Goffe et al. (1994), Fishman (1996, pp. 384–406) and Robert and Casella (1999, pp. 194–202). The program used in this paper is a C++ implementation of the algorithm in Goffe et al. (1994).
4. From a computational perspective, there are therefore $s \cdot 2^{\tau}$ different probabilities τ periods ahead, each of which should be multiplied by the corresponding τ periods ahead forecast of $\mathbf{x}$.
5. A detailed treatment of gradient descent learning for RNNs is provided in Pearlmutter (1995) and for brevity is not repeated here.
6. From hereon simply referred to as FF-MLP.
7. Our sample is restricted backward in time by the availability of data on M0 in EcoWin.

ACKNOWLEDGMENTS

Thomas Elger and Birger Nilsson wish to thank the Jan Wallander and Tom Hedelius foundation (Award No. J03/19 Thomas Elger and Award No. J02/08 Birger Nilsson).

REFERENCES

Ang, A., & Bekaert, G. (2002). International asset allocation with regime shifts. *Review of Financial Studies*, *15*, 1137–1187.

Binner, J. M., Gazely, A. M., & Chen, S. H. (2002). Financial innovation and Divisia monetary indices in Taiwan: A neural network approach. *European Journal of Finance*, *8*, 238–247.

Blix, M. (1999). Forecasting Swedish inflation with a Markov switching VAR. Sveriges Riksbank Working Paper, No 76.

Chappelier, J.-C., Gori, M., & Grumbach, A. (2000). Time in connectionist models. In: R. Sun & L. Giles (Eds), *Sequence Learning: Paradigms, Algorithms, and Applications* (pp. 105–134). Springer Verlag.

Cheng, B., & Titterington, D. M. (1994). Neural networks: A review from a statistical perspective. *Statistical Science*, *9*, 2–54.

Connor, J., Martin, R., & Atlas, L. (1994). Recurrent neural networks and robust time series prediction. *IEEE Transactions on Neural Networks*, *5*, 240–254.

Corona, A., Marchesi, M., Martini, C., & Ridella, S. (1987). Minimizing multimodal functions of continuous variables with the 'simulated annealing' algorithm. *ACM Transactions of Mathematical Software*, *13*, 262–280.

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

Darken, C., & Moody, J. E. (1991). Note on learning rate schedules for stochastic optimization. In: R. P. Lippmann, J. E. Moody & D. S. Touretzky (Eds), *Advances in Neural Information Processing Systems* (Vol. 3, pp. 832–838).

Elman, J. (1990). Finding structure in time. *Cognitive Science*, *14*, 179–211.

Enders, W. (1995). *Applied econometric time series*. Wiley.

Fishman, G. (1996). *Monte Carlo concepts, algorithms and applications*. Springer Verlag.

Franses, P. H., & van Griensven, K. (1998). Forecasting exchange rates using neural networks for technical trading rules. *Studies in Nonlinear Dynamics and Econometrics*, *2*, 109–116.

Franses, P. H., & van Homelen, D. (1998). On forecasting exchange rates using neural networks. *Applied Financial Economics*, *8*, 589–596.

Garcia, R., & Perron, P. (1996). An analysis of the real interest rate under regime shifts. *Review of Economics and Statistics*, *78*, 111–125.

Gençay, R. (1996). Non-linear prediction for security returns with moving average rules. *Journal of Forecasting*, *15*, 165–174.

Gençay, R. (1999). Linear, non-linear and essential foreign exchange rate predictions with simple technical trading rules. *Journal of International Economics*, *47*, 91–107.

Gençay, R., & Stengos, T. (1998). Moving average rules, volume and the predictability of security returns with feedforward networks. *Journal of Forecasting*, *17*, 401–414.

Giles, C. L., Lawrence, S., & Tsoi, A. C. (2001). Noisy time series prediction using a recurrent neural network and grammatical inference. *Machine Learning*, *44*, 161–183.

Goffe, W., Ferrier, G., & Rogers, J. (1994). Global optimization of statistical functions with simulated annealing. *Journal of Econometrics*, *60*, 65–99.

Goldfeld, S., & Quandt, R. (1973). A Markov model for switching regressions. *Journal of Econometrics*, *1*, 3–16.

Haefke, C., & Helmenstein, C. (1996a). Forecasting Austrian IPOs: An application of linear and neural network error-correction models. *Journal of Forecasting*, *15*, 237–251.

Haefke, C., & Helmenstein, C. (1996b). Neural networks in the capital markets: An application to index forecasting. *Computational Economics*, *9*, 37–50.

Hamilton, J. (1988). Rational-expectations econometric analysis of changes in regime: An investigation of the term structure of interest rates. *Journal of Economic Dynamics and Control*, *12*, 385–423.

Hamilton, J. (1989). A new approach to the economic analysis of non-stationary time series and the business cycle. *Econometrica*, *57*, 357–384.

Hamilton, J. (1994). *Time series analysis*. Princeton University Press.

Haykin, S. (1999). *Neural networks: A comprehensive foundation* (2nd ed.). Upper Saddle River, NJ: Prentice-Hall.

Hornik, K. (1991). Approximation capabilities of multilayer feedforward networks. *Neural Networks*, *4*, 251–257.

Jordan, M. (1986). Attractor dynamics and parallelism in a connectionist sequential machine. *Proceedings of the 8th Annual Conference of the Cognitive Science Society*, 531–545.

Khan, E., & Unal, F. (1995). Recurrent fuzzy logic using neural networks. In: T. Furuhashi (Ed.), *Advances in Fuzzy Logic, Neural Networks, and Genetic Algorithms* (Lecture Notes in AI, pp. 48–55). Springer Verlag.

Klaassen, F. (2002). Improving GARCH volatility forecasts with regime-switching GARCH. *Empirical Economics*, *27*, 363–394.

Kovalerchuk, B., & Vityaev, E. (2000). *Data mining in finance: Advances in relational and hybrid methods*. Kluwer.

Krolzig, H.-M. (2004). Predicting Markov-switching vector autoregressive processes. Working Paper, Nuffield College, Oxford.

Kuan, C.-M., & Liu, T. (1995). Forecasting exchange rates using feedforward and recurrent neural networks. *Journal of Applied Econometrics*, *10*, 347–364.

Lawrence, S., Giles, C. L., & Fong, S. (2000). Natural language grammatical inference with recurrent neural networks. *IEEE Transactions on Knowledge and Data Engineering*, *12*, 126–140.

Lee, K. (1989). Hidden Markov models: Past, present and future. *Proceedings of Eurospeech*, *89*, 148–155.

Lütkepohl, H. (1991). *Introduction to multiple time series analysis*. Springer Verlag.

Moshiri, S., Cameron, N., & Scuse, D. (1999). Static, dynamic and hybrid ANN models in forecasting inflation. *Computational Economics*, *14*, 219–235.

Pearlmutter, B. A. (1995). Gradient calculations for dynamic recurrent neural networks: A survey. *IEEE Transactions on Neural Networks*, *6*, 1212–1228.

Qi, M., & Maddala, G. S. (1995). Option pricing using ANN: The case of S&P 500 index call options. *Neural Networks in Financial Engineering: Proceedings of the 3rd International Conference on Neural Networks in the Capital Markets*, 78–91.

Robert, P. C., & Casella, G. (1999). *Monte Carlo statistical methods*. Springer Verlag.

Rumelhart, D., Hinton, G., & Williams, R. (1986). Learning internal representations by error propagation. In: *Parallel Distributed Processing, Explorations in the Microstructure of Cognition* (Vol. 1, pp. 318–362). Princeton University Press.

Shadbolt, J., & Taylor, J. G. (2002). *Neural networks and the financial markets: Predicting, combining and portfolio optimisation* (Perspectives in neural computing). Springer Verlag U.K.

Siegelmann, H. T., & Sontag, E. D. (1991). Turing computability with neural nets. *Applied Mathematics Letters*, *4*, 77–80.

Stock, J. H., & Watson, M. W. (1999). A comparison of linear and non-linear univariate models for forecasting macroeconomic time series. In: R. Engle & R. White (Eds), *Cointegration, Causality and Forecasting: A Festschrift in Honor of Clive W. J. Granger* (pp. 1–44). Oxford: Oxford University Press.

Swanson, N. R., & White, H. (1995). A model selection approach to assessing the information in the term structure using linear models and artificial neural networks. *Journal of Business & Economic Statistics*, *13*, 265–275.

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

Werbos, P. J. (1990). Backpropagation through time: What it does and how to do it. *Proceedings of IEEE Special Issue on Neural Networks*, *78*, 1550–1560.

Williams, R. J., & Peng, J. (1990). An efficient gradient-based algorithm for on-line training of recurrent network trajectories. *Neural Computation*, *2*, 490–501.

USING NON-PARAMETRIC SEARCH ALGORITHMS TO FORECAST DAILY EXCESS STOCK RETURNS

Nathan Lael Joseph, David S. Brée
and Efstathios Kalyvas

ABSTRACT

Are the learning procedures of genetic algorithms (GAs) able to generate optimal architectures for artificial neural networks (ANNs) in high frequency data? In this experimental study, GAs are used to identify the best *architecture for ANNs. Additional learning is undertaken by the ANNs to forecast daily excess stock returns. No ANN architectures were able to outperform a random walk, despite the finding of non-linearity in the excess returns. This failure is attributed to the absence of suitable ANN structures and further implies that researchers need to be cautious when making inferences from ANN results that use high frequency data.*

1. INTRODUCTION

Over the years a large body of literature has developed on the ability of search algorithms from artificial intelligence (AI) to generate parameters that facilitate estimation, classification and prediction. Two such algorithms that have since been applied (separately and sequentially) in empirical research in finance and

Applications of Artificial Intelligence in Finance and Economics
Advances in Econometrics, Volume 19, 93–125

ISSN: 0731-9053/doi:10.1016/S0731-9053(04)19004-X

economics are genetic algorithms (GAs), and artificial neural networks (ANNs). For example, GAs and their extension in the form of genetic programming have been used to: (i) identify *optimal* trading rules aimed at assessing the economic benefit of trades based on financial asset price movements (see, e.g. Allen & Karjalainen, 1999; Neely et al., 1997); and (ii) to solve bankruptcy classification and prediction problems (see Varetto, 1998).[1] ANNs have also been used in related areas. For example, ANNs have been used to predict both exchange rates (see Qi & Wu, 2003) and mutual fund performance (see Indro et al., 1999), as well as to facilitate bankruptcy classification and prediction (see Altman et al., 1994). Estimations using GAs and ANNs are very appealing since they are essentially non-parametric econometric search techniques that do not require specific distributional assumptions regarding the innovations in the data – a condition which plagues traditional linear models in time series settings. Such non-parametric models are therefore useful when the dataset to be estimated contains several local optima, noise and non-linearities.

An ANN typically consists of nodes arranged in layers, with each node communicating its output to other nodes. There is always an input layer of nodes, into which the input data is stored, repeatedly, and an output layer of nodes, from which the predicted output is read off. Each node takes a weighted sum of its input, transforms the result with some function, which has one parameter, and outputs the result. The ANNs are generally used in a feed forward manner, with each node taking its inputs from the previous layer and sending its output to the nodes in the next layer. The weights and the function parameter in any node are estimated from the data, usually by comparing the output predicted by the model with the actual data. Weight estimation, called training, is continued until further refinement leads to over-fitting the training data, according to a separate validation data set.

The straightforward use of ANNs pre-supposes that their learning algorithm is able to find parameters that globally optimize the mapping from input to output variables. Indeed, theoretical work shows that any non-linear mapping can be well approximated by an ANN if it has a sufficient number of hidden nodes. Admittedly, this presumption of *perfect* approximation suggests that the parameters that need to be set by the modeler are known precisely, and that there is a sufficiently good match between the decision rules undertaken in the experiments and the learning undertaken by the algorithm. In practice, there is not much guidance on how to choose the appropriate number of hidden nodes; consequently standard applications of ANNs often suffer from over-fitting.

The most common application of ANN relies on a gradient search technique, such as back propagation (BP), for estimating the weights (parameters) linking the nodes, using a common weight adjustment factor. BP requires that the optimal architecture is chosen before network training, preferably in a scientific way. Also

the gradient decent method commonly used in BP is not a reliable way to train networks (see also, Curry & Morgan, 1997) as the method may get caught in local solutions that are not global. The solutions found depend on the initial weights that are drawn at random.[2] This method runs the risk that the ANNs may not accurately learn the parameters of the data if procedures are not in place to ensure global optimization. Thus, in practice, the output from a single ANN is not an adequate test of the suitability of the ANN to model the data. The most suitable ANN structure needs to be selected based on an assessment of its performance relative to that of other structures. In order to ensure that performance measures are reliably estimated it is desirable both to search for better estimation procedures to specify the ANN structure and to repeat the learning procedure several times as we have done in this experimental study or perhaps using a bootstrapping approach (see, e.g. LeBaron & Weigend, 1994).

To deal with the problem of gradient search and local convergence in BP, several alternative solutions have been proposed. Such techniques include: (i) direct optimization procedures such as a polytope algorithm; (ii) evolutionary programming approaches such as GAs; and (iii) global search procedures such as simulated annealing. These approaches appear to generate solutions that are superior to those of the straightforward use of the gradient technique in BP. GAs have been shown to have very *good* global convergence properties. Indeed, Sexton et al. (1999) report that the use of GAs achieves superior solutions compared with those of simulated annealing and both techniques perform better than the gradient technique of BP. As the GA seeks to provide a more parsimonious architecture that will increase the chances of the ANN finding a global optimum, the scope for the ANN to model complex problems is substantially reduced (see Sexton et al., 1998). The use of GAs however, entails some form of reinforcement learning for ANNs, which in turn reduces both the training and flexibility inherent in standard ANNs and their ability to model complex problems.

Specifically, GAs are a broad robust class of adaptive algorithms particularly adept to finding solutions to objective functions that exhibit many local optima, high dimensionality, noise and discontinuities (see, e.g. Bethke, 1981). They operate by testing a subset of solutions from a domain of possible solutions. In so doing, GAs have the ability to combine useful elements of distant information with new innovations in the data so as to generate the most representative elements of the data in their outputs. A GA seeks to evolve a population of good decision rules (or chromosomes) that represents different beliefs in response to past experiences. Each decision rule is evaluated via a fitness function, which measures the outcome or utility from the behavior dictated by the rule.

Arifovic and Gençay's (2001) work is one of the few experimental studies in financial research to employ a GA for generating the architecture for an ANN.[3] Using the ANN architecture selected by a GA, they compared the forecasting

performance of the ANN against that of two linear models for daily French franc exchange rates. The ANN generated an out-of-sample mean square prediction error (MSPE) which is 18% better than that of the linear models. This is a promising result, but suffers from an important drawback: their results only allow them to reject the linear models in favor of the ANN alternative since the forecasts were not compared with those of a naïve forecasting model, e.g. a random walk (RW).

To minimize the over-fitting problems of ANNs, researchers have also used bootstrapping procedures (see Franke & Neumann, 2000; LeBaron, 1997; LeBaron & Weigend, 1994). Bootstrapping is a method for estimating the generalization error based on resampling, and is useful for model selection and confidence interval estimation. These studies suggest that bootstrapping the network generates more parsimonious network structures although LeBaron (1997, p. 12) notes that "... no system trying to use training data for model selection can ... be completely free of overfitting problems." Despite this view, it seems reasonable to assess estimation techniques that are considered appropriate for minimizing the architecture selection bias in ANNs. This is a main objective of our study which we explore in much detail using a set of robust procedures.

This experimental study therefore investigates whether it is useful econometrically to employ GAs to prespecify the architecture for ANNs when seeking to forecast daily excess stock returns. Arifovic and Gençay's (2001) experimental work appears to lack much rigor both in the GA application and the validation and testing procedures for the network. Like most of that literature, their ANN forecast is not evaluated against a naïve scheme of forecasting no change. We seek to apply much more rigor to our experimental design by explicitly exploiting the evolution of the forecasts and their standard errors across GA generations. Our approach is in some respects similar in spirit to the use of bootstrapping to select suitable ANN structures. The architectures of the ANNs are also evaluated, both in- and out-of-sample, against a set of naïve RW models and the mean absolute error (MAE). The use of in-sample forecasts facilitates an assessment of the suitability of the chosen ANN configuration prior to implementation, while the use of RW models serves to evaluate the contribution of the ANNs to forecasting accuracy (see Collopy et al., 1994). We apply much rigor to our validation and testing procedures and seek to avoid some of the common weakness of existing ANN applications (see, e.g. Adya & Collopy, 1998). For example, many empirical studies do not appear to cross-validate their results so as to ensure that the chosen architecture does not result in over-fitting.[4]

Since stock index returns often serve as benchmarks against which the performance of both private clients and fund managers is judged (see Baker, 1998), it is economically useful to assess whether it is possible to forecast the returns of stock indices. However, a model that predicts stock index returns can appear to do

well by simply predicting the average positive drift in the index returns. Instead, we will see if the excess returns of two stock indices, i.e. the index returns over and above the risk free interest rate, can be predicted.[5] The empirical results will have important implications for the investment strategies that an investor might choose to follow. For example, a forecast at $t-1$ that next period's excess return will be positive would suggest that a fund manager will want to switch his/her investments from Treasury bills to the stock market to benefit from the stock price rise. Similarly, a forecast at $t-1$ that next period's excess return will be negative would imply that the fund manager will want to switch his/her investment to Treasury bills.[6] The use of *daily* excess returns imposes a very demanding task on the algorithms we employ, as high frequency data are likely to contain much more noise and local optima, and non-linearity, than (say) monthly data.[7] Since we do not account for transaction costs and risk aversion, we do not suggest that our findings necessarily (in)validate the efficient market hypothesis. Any finding relating to market (in)efficiency can only be interpreted in conjunction with an intertemporal equilibrium model of the economy.

The next section describes the data sets that are used. Much effort is put into identifying non-linearity in the series, as evidence of this strengthens the case for the use of non-parametric search algorithms. Section 3 introduces our benchmarks of forecasting performance. We describe the experimental design in Section 4. The main results are presented in Section 5 and we provide our conclusions in the final section.

2. THE DATA SETS AND DESCRIPTIVE STATISTICS

We seek to predict the realized excess return defined as

$$y_t = \ln\left(\frac{p_t}{p_{t-1}}\right) - \ln\left(1 + \frac{c_{t-1}}{360}\right) \tag{1}$$

but unknown at $t-1$ and each subsequent period. Here, p_{t-1} is the stock price at time $t-1$ and c_{t-1} is the risk-free annualized interest rate that will apply at the end of the investment period.

To conduct the experiments, two daily excess return series were generated using the: (i) Standard and Poor (S&P)–500 and the 13-week U.S. Treasury bill rates and (ii) the Financial Times Stock Exchange (FTSE)–100 indices and the 1-month U.K. Treasury bill rate. All the raw time series data were obtained from *DataStream International* online database. The data sets span the period from 4 January 1988 to 12 December 2000. The start date of 1988 was taken because we wanted to avoid the econometric problems associated with identifying the data points for structural

Table 1. Descriptive Statistics for Excess Returns.

Panel A

	N	Mean	S.D.	Skewness	Kurtosis	ADF Statistics			
						τ_μ	*m*	τ_τ	*m*
S&P excess return	2976	0.00043[b]	0.0090	−0.636[a]	7.044[a]	−16.373[a]	11	−16.407[a]	11
FTSE excess return	2974	0.00013	0.0076	−0.076	2.587[a]	−37.254[a]	1	−37.258[a]	1

Panel B: Autocorrelation Coefficients of Square of Excess Returns

	Lags							
	1	2	3	4	5	10	15	20
S&P excess return	0.184[a]	0.078[a]	0.040[a]	0.066[a]	0.135[a]	0.042[a]	0.040[a]	0.041[a]
FTSE excess return	0.147[a]	0.142[a]	0.115[a]	0.100[a]	0.091[a]	0.101[a]	0.084[a]	0.041[a]

Note: *m* is the lag structure for the ADF statistics. S.D. denotes the standard deviation. The samples span the period 4th January 1988 to 6th October 1999.

[a] Indicate statistical significance at less than the 1% level.

[b] Indicate statistical significance at less than the 5% level.

breaks around the period of the stock market crash of October 1987. The last 300 daily observations (10% of the data) were reserved for out-of-sample forecasting (testing) leaving 2977 (approx.) for the GA search and ANN training. The size of the test set was necessary to ensure that there were sufficient data points in the training set for the number of parameters in the ANN models, which could have been as high as 1590 but, in practice turned out to be around 100 or less.[8]

Before applying the learning algorithms, it is useful to understand the dynamics of the excess returns and their data generating process (DGP) as this will enable us to be confident regarding the outcome of our experiments. Panel A of Table 1 presents the descriptive statistics for the computed excess returns. Unless otherwise stated, the results are based on the observations up to 6th October 1999; that is, excluding the last 300 observations reserved for out-of-sample forecasting/testing. The table shows that the means of the daily excess returns y_t are positive, but only the mean for the S&P is significantly different from zero. Skewness and kurtosis are, however, significant and the DGP of the series appears to result in distributions with fat-tails that are leptokurtic. This feature suggests that the observations cluster during certain periods.

Panel B of the table also shows that the square of the excess returns are positively autocorrelated (p-value < 0.01). The standard errors of the autocorrelation coefficients were not adjusted for heteroscedasticity, but Hsieh's (1989) results suggest that adjusting them would provide stronger evidence of autocorrelation. To explore possible clustering of the data points, we performed the following simple experiment on the *raw* excess return series, y_t. Consider first the possibility that each observation for a given series can be either below or above the median. Attribute a "+" if an observation is above the median and a "−" otherwise. This procedure results is a new series which we call S. If the initial sequence contained an odd number of points we ignored the median point. So S will contain m "+"s and m "−"s and the length of S will be $2m$. We also define r_+ and r_- as the number of runs that contain "+" and "−" respectively and r to be equal to $r_+ + r_-$. If r is even in S we have $r_+ = r_-$. Otherwise, r is odd and either $r_+ = r_- + 1$ or $r_- = r_+ + 1$. We apply a runs test based on the intuitive notion that an unusually large or an unusually small value of r would suggest lack of randomness. More specifically, for a random series S, r is approximately normally distributed with mean of $E(r) = m + 1$ and a variance of $\mathrm{Var}(r) = m(m+1)/(2m-1) \sim (2m-1)/4$ when m is large.

To confirm this approximation to a normal distribution, we used the Matlab 5.2 facility to generate 5000 series of random numbers belonging in the interval [0,1] having the same length as each excess return series, y_t. For each series, each random number was replaced by a "+" if it was larger than the previous random number, and by a "−" otherwise. We calculated r for each series and compared it with the theoretical distribution. The results in Fig. 1 show that the distribution of r for the generated series does indeed closely follow the normal distribution. Using the standard normal distribution $N(0,1)$, we then calculate the probability $P(Y \leq \bar{r}_{\mathrm{obs}})$ for each of our actual data series, where: $\bar{r}_{\mathrm{obs}} = (r_{\mathrm{obs}} - E(r))/\mathrm{Var}(r)$, Y is a random variable of the standard normal distribution, and r_{obs} is the observed

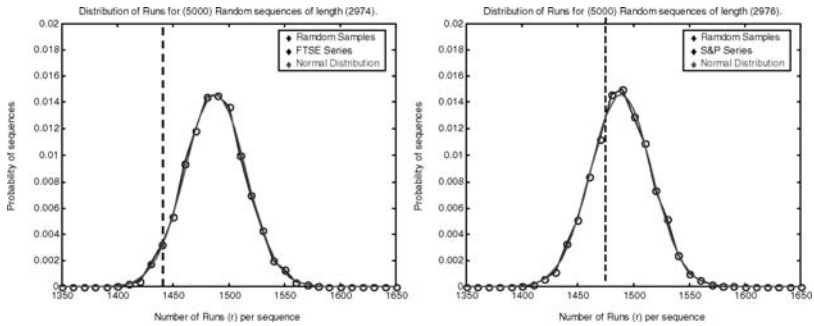


Fig. 1. Runs Tests for the Excess Return Series.

r for a series. We found that $P(Y \leq \bar{r}_{\text{obs}}) = 0.04$ for the FTSE, so we can be 96% confident that the observations in this series are not random. However, the value for the S&P excess return series is 0.45, so the null hypothesis of normality cannot be rejected.

The presence of autocorrelation in the univariate series (see Table 1, Panel B) also suggests that we should try to identify the ARMA model that best approximates each y_t series. Firstly, we computed the augmented Dickey-Fuller (ADF) statistics by varying the lag structure of the variable from 12 to 1. The reported statistics are the highest ones provided the residuals were white noise (p-value >0.05). The ADF statistics suggest the excess returns series are stationary (see Table 1, Panel A). So no further differencing is necessary.

We then estimated 25 ARMA(p, q) for each series such that $p, q = 0, 1, 2, 3, 4$, where p is the number of lagged variables and q the number of lagged moving averages. The chosen ARMA is the one with the highest Akaike information criterion (AIC). Panel A of Table 2 indicates that the series have a mix of AR and moving average representations. That is, ARMA(4,4) and ARMA(2,4) best describe the S&P and FTSE series, respectively. Thus the excess return series are low-order processes with a memory of up to four periods. Notice, though, that $\hat{\theta}_4$ is not significantly different from zero for the FTSE such that perhaps an ARMA(2,3) might have been more appropriate, although this is not supported by the AIC. If a relative short memory process encapsulates the DGP, it would appear that relatively simple algorithms rather then complicated ones might generate the *best* forecasts. As the autocorrelation coefficients of the square residuals of the ARMA models are significant (p-value < 0.01), not all the information contained in the data has been exploited.

As a final test of the iid condition, we fitted an EGARCH(1,1) model to capture the conditional distributional errors in each y_t series. Panel A of Table 3 shows that the estimated coefficients are non-zero but the fit of the model is not appealing on the basis of the t-distribution parameter. Similarly, the EGARCH(1,1) process is not too stable since $|\eta_1|$ is not less than one.

To test for non-linearity in the standardized residuals of the EGARCH(1,1) model, the BDS statistic (see Brock et al., 1991) was applied. The BDS statistic is considered to be one of the best tests for non-linearity (see Barnett et al., 1997). Panel B of the table shows that the standardized residuals of the EGARCH models are not iid across embedded dimensions. As noted by Brock et al. (1991), the size of the BDS statistic depends on the value of ℓ. However, the evidence against iid is still rejected for all measures of ℓ, although the magnitude of the statistic varies across embedded dimensions. de Lima (1998) reports similar results for the S&P 500 stock returns and indicated that the iid condition was overwhelming rejected in periods of high volatility.

Table 2. Estimates of ARMA Models.

Panel A

	ARMA Estimates								
	α	ϕ_1	ϕ_2	ϕ_3	ϕ_4	θ_1	θ_2	θ_3	θ_4
S&P excess returns	0.36E-3[b]	−0.2001[b]	0.0719	0.0429	0.2871[b]	0.1876	−0.0906	−0.0960	−0.3181[a]
	(0.14E-3)	(0.1019)	(0.1120)	(0.1023)	(0.1166)	(0.1022)	(0.1133)	(0.1026)	(0.1201)
	$\bar{R}^2 = 0.0034$, AIC = 9801.8								
FTSE excess returns	0.59E-4	1.3592[a]	−0.8028[a]			−1.2627[a]	0.6668[a]	0.0967[a]	−0.0191
	(0.67E-4)	(0.1039)	(0.0967)			(0.1057)	(0.0934)	(0.0300)	(0.0184)
	$\bar{R}^2 = 0.0062$, AIC = 10300.0								

Panel B: Autocorrelation Coefficients of Square Residuals of ARMA Models

	Lags							
	1	2	3	4	5	10	15	20
S&P excess return	0.196[a]	0.106[a]	0.053[a]	0.071[a]	0.139[a]	0.046[a]	0.041[a]	0.046[a]
FTSE excess return	0.149[a]	0.152[a]	0.121[a]	0.099[a]	0.093[a]	0.116[a]	0.088[a]	0.041[a]

Note: The samples span the period 4th January 1988 to 6th October 1999. The ARMA(p,q) model for series y_t can be stated as $y_t = \alpha + \phi_1 y_{t-1} + \cdots + \phi_p y_{t-p} + \varepsilon_t - \theta_1 \varepsilon_{t-1} - \cdots - \theta_q \varepsilon_{t-q}$. ϕ_p and θ_q are respectively the coefficients for the autoregressive and moving average operators. The statistical significance in Panel B is based on the Ljung-Box statistic.

[a] Indicate statistical significance at less than the 1% level.

[b] Indicate statistical significance at less than the 5% level.

Table 3. Conditional Estimates of EGARCH (1,1) Model and BDS Statistics.

Panel A

	Parameters of Conditional Heteroscedasticity for Exponential GARCH(1,1)				*t*-Distribution	$\bar{R}^2$	AIC
	ϖ_0	φ_1	φ_2	η_1			
S&P excess returns	−0.0647[b]	−0.0340[a]	0.0861[a]	0.9933[a]	5.5951[a]	0.0028	10142.2
	(0.0328)	(0.0110)	(0.0162)	(0.0034)	(0.5710)		
FTSE excess returns	−0.1106[a]	−0.0341[a]	−0.0911[a]	0.9889[a]	10.9872[a]	0.0079	10516.7
	(0.0398)	(0.0084)	(0.0165)	(0.0040)	(1.7739)		

Panel B: BDS Statistics for Standardized Residuals

	m								
	2	3	4	5	6	7	8	9	10
S&P excess returns									
$\ell = 0.25$	5.067	7.485	9.093	11.225	12.468	14.237	14.067	6.691	−5.194
$\ell = 0.50$	4.659	6.823	8.426	10.639	12.583	14.989	17.394	20.150	23.103
$\ell = 0.75$	4.818	6.957	8.560	10.613	12.648	14.979	17.292	20.283	23.785
$\ell = 1.00$	5.096	7.218	8.657	10.521	12.449	14.660	16.724	19.167	21.966
$\ell = 1.25$	5.282	7.404	8.850	10.550	12.161	13.966	15.562	17.331	19.237
$\ell = 1.50$	5.609	7.636	9.066	10.546	11.812	13.245	14.480	15.781	17.202
FTSE excess returns									
$\ell = 0.25$	4.297	6.763	8.574	9.886	9.114	15.533	31.529	81.491	−7.685
$\ell = 0.50$	4.287	7.260	8.933	10.556	12.489	15.395	17.814	18.663	17.669
$\ell = 0.75$	4.574	7.542	9.006	10.304	11.672	13.735	15.718	18.045	20.284
$\ell = 1.00$	4.965	7.926	9.460	10.698	12.101	14.083	15.898	17.758	19.811
$\ell = 1.25$	5.609	8.507	10.024	11.175	12.407	14.033	15.429	16.732	18.231
$\ell = 1.50$	6.445	9.335	10.823	11.823	12.841	14.170	15.260	16.230	17.414

Note: The EGARCH(1,1) conditional errors for an equation of $y_t = a_0 + b_0\, y_{t-i} + \mu_t$ at i lags can be written as $\log h_t^2 = \varpi_0 + \varphi_1(u_{t-1}/h_{t-1}) + \varphi_2(|u_{t-1}/h_{t-1}| - \mu_t) + \eta_1 \log h_{t-1}^2$ and $\mu_t|\Omega_{t-1} \sim N(0, h_t)$ where μ_t is the disturbance term, $h_t{}^2$ is the conditional variance of μ_t and Ω_{t-1} is the information set at *t–1*. $N(0, h_t)$ depicts a distrbution with a mean of zero and a variance of h_t which we assumed to be the *t*-density distribution with ν degrees of freedom. The EGARCH(1,1) model was estimated with a lag of 7 for the S&P series and a lag of 1 for the FTSE series. The BDS statistic is for ℓ-standard deviations and m-dimensions of 2–10. The statistic is distributed asymptotic normal so that the 99% critical value of 2.58 (two-tailed) rejects iid. The samples span the period 4th January 1988 to 6th October 1999.

[a] Indicate statistical significance at less than the 1% level.

[b] Indicate statistical significance at less than the 5% level.

In general, the excess return series appear to contain statistical properties that are appropriate for our learning algorithms. Indeed, LeBaron et al. (1999) applied a GA to artificial data that generated much of the statistical properties depicted here. The sources of the non-linearity in the excess returns might reflect the response of the stock market to both positive and negative news. It would seem *a priori* that our algorithms would be able to cope with the leptokurtic and conditional properties of the series. If, however, the DGP switches between tranquil periods of purely random motion and more turbulent periods as suggested for artificial data (see Chen et al., 2001), the statistical properties of the excess returns might well create problems of tractability for the algorithms.

3. BENCHMARKS FOR ASSESSING FORECASTING PERFORMANCE

To evaluate forecasting performance we use three different Theil-type measures as well as a MAE. The Theil-type measures are unit free and provide an easy way of assessing the ANN's forecasting performance against the forecasts from a naïve RW model. For example, a Theil statistic of less than one implies that the ANN generated forecasts that are superior to those generated by a RW. Following Theil (1966), we write the Theil statistic in a general form

$$\text{Theil} = \frac{\sqrt{\sum_{t=1}^{N}(y_t - \hat{y}_t)^2}}{\sqrt{\sum_{t=1}^{N}(y_t - y'_t)^2}} \tag{2}$$

where $\hat{y}_t$ is the forecast for time t, y_t is the actual observation, and y'_t is the expected excess stock return according to a RW. There are different ways for specifying the RW value y'_t.

Usually when using log ratios, as we do in Eq. (1), the RW prediction is zero, i.e. $y'_t = 0$, which occurs when the expected stock return equals the risk free rate. We can then write our first Theil statistic as

$$\text{TheilA} = \frac{\sqrt{\sum_{t=1}^{N}(y_t - \hat{y}_t)^2}}{\sqrt{\sum_{t=1}^{N} y_t^2}}. \tag{3}$$

Equation (3) is not the only way to specify a RW of excess returns for y_t. An alternative is a RW around *actual* prices, i.e. the prediction is for the price not to change from one day to the next. Thus $p'_t = p_{t-1}$ where p'_t is the prediction of the RW for day t. This alternative RW around today's price is not a strong competitor

against which to measure forecasting performance, because we know that there is a positive drift in the prices; we nevertheless use it in our forecast evaluation. Substituting p'_t for p_{t-1} in Eq. (1), we can write the prediction of the excess returns on day t according to a RW on *prices* as

$$y'_t = -\ln\left(\frac{c_{t-1}}{360} + 1\right). \tag{4}$$

Substituting Eq. (4) in Eq. (2) gives our second Theil-type statistic

$$\text{TheilB} = \frac{\sqrt{\sum_{t=1}^{N}(y_t - \hat{y}_t)^2}}{\sqrt{\sum_{t=1}^{N}(y_t - \ln(\frac{c_{t-1}}{360} + 1))^2}}. \tag{5}$$

A final Theil statistic we can specify is based on the idea that the *momentum* of the excess return series follows a RW, i.e. $y'_t = y_{t-1}$. Here, the RW assumes that the prediction of the excess return on day t is equal to the excess return on day $t-1$, i.e. bull and bear markets are likely to have long runs. We show this in TheilC which compares the forecast $\hat{y}_t$ with this alternative RW and is written as

$$\text{TheilC} = \frac{\sqrt{\sum_{t=1}^{N}(y_t - \hat{y}_t)^2}}{\sqrt{\sum_{t=1}^{N}(y_t - y_{t-1})^2}}. \tag{6}$$

Finally, we define the MAE as

$$\frac{\sum_{t=1}^{N} \left|y_t - \hat{y}_t\right|}{N}. \tag{7}$$

The MAE is known to be more reliable than either the mean square error or root mean square error and less sensitive to the presence of outliers (see, e.g. Meese & Rogoff, 1983).

4. EXPERIMENTAL DESIGN AND MODEL IMPLEMENTATION

Our experimental design, described in this section, has been particularly carefully chosen because we ultimately expected negative results in our research environment and we wanted to ensure that a very robust set of empirical results was ultimately generated.[9] To achieve this objective, the GA is given a wide area to search for ANN structures. The chosen ANN structures are trained up to 50 times using various criteria, since only repeated training gives an indication of the reliability of the results, given the non-deterministic nature of ANN training.

In particular, these actions guard against the temptation to report the best of the several runs that might have been carried out. Also, we compare the predictive model against three RW models, as well as the MAE, for two data sets.[10] Usually in ANN applications, the ANN forecasts are evaluated against the same metric which was used for training, but this is not robust. A good model should be good using any suitable measure. Hence we will test, repeatedly, each ANN not only against the metric used in its training, but also against the remaining three metrics. We believe that our cautious and rigorous methodological approach goes some way to allaying Adya and Collopy's (1998) concerns regarding previous ANN methodological applications. Given this rigorous methodological approach, our experimental results will remind researchers that they always need to use appropriate forecast benchmarks and extensively retrain their ANNs before they can have confidence in their results. We now describe our experimental results in more detail.

Several parameters need to be set in order to obtain suitable ANN architectures. Take an ANN structure *x*-*y*-*z*-1 where *x*, *y* and *z* are any positive integers. These parameters represent the size of the input, the first hidden and the second hidden layers, respectively. The integer one represents the single output node.[11] The transformation functions for all the hidden layer nodes was *tansigmoid*, and for the output node *nonlin*. The structure represents the size of space searched by the GA, up to the maximum established values. To allow a sufficiently wide and extensive search space, we chose to limit the inputs of the ANNs to at most 20 lags ($x \leq 20$) for each series. The goal was to allow as much flexibility as possible in the choice of lags for fitting the data thereby allowing the GA to search across a sufficiently broad range of lag structures. The number of hidden layers in each ANN was set to be at most two on the grounds that two hidden layers are sufficient to capture any function; the second hidden layer enables the first hidden layer to capture any pre-processing of the input data. The numbers of nodes in each hidden layer, *y* and *z*, were set to a maximum of 30 each. This allows quite a large area to search. A chosen structure that is smaller than that required by the data will prevent the algorithm from learning the required representation. Alternatively, a much larger structure than is required makes generalization difficult for the algorithms. So we experimented within that range. It turns out that in all the searches the best ANNs fell within, rather than at, these upper limits. The string used to represent the ANN structure in the GA was, therefore, three integers, each representing the number of nodes in the three layers: input, hidden1 and hidden2. That is, the alphabet from which the strings were made consisted of three symbols, each representing a number between 1 and 30.

Another set of parameters affects the GA search directly. They include not only the termination criterion (*maxGen*) and the number of chromosomes (rules) per generation (*m*), (where higher values of *m* and *maxGen* imply a more expensive

search), but also the three probability parameters that seek to capture the chances of selecting an architecture from one generation to the next. These parameters are: reproduction ($P_{\text{reproduction}}$), crossover ($P_{\text{crossover}}$) and mutation (P_{mutation}). Prior experimental work suggests a strong relationship between the desirable values of these parameters, although there is little agreement as to what the exact values should be. For example, Gathercole and Ross (1997) suggest that a small population for the training set is better than a large one, while Goldberg (1989) employed a population size $m = 50$, with $P_{\text{crossover}} = 0.90$ and $P_{\text{mutation}} = 0.01$.[12] Because we wanted to force the GA to make a sparser search in our space, we set: $m = 40$; *maxGen* $= 25$; $P_{\text{crossover}} = 0.60$; $P_{\text{mutation}} = 0.10$, and $P_{\text{reproduction}} = 0.30$. The cost of adopting this strategy is that it is more difficult for our GA to converge to a single network structure. We dealt with this consequence by selecting the three best network structures from the final GA generation.

To identify the most appropriate training algorithm, various ANN algorithms were run in a pilot study. The BP algorithm was found to be the fastest to converge, so needing fewest training epochs – a result similar to that of Demuth and Beale (1997). Therefore all the networks in this study were trained using the resilient BP algorithm with: (i) the initial weight adjustment factor, *delta_initial* $= 0.07$; (ii) the increase in the factor for each successive change in the same direction, *delta_increase* $= 1.20$; and (iii) the decrease for successive changes in opposite directions, *delta_decrease* $= 0.50$. In preliminary experiments, the GAs were also run several times to ensure that the reported results were robust to reasonable changes in the parameter choices. Any change to these parameters within reasonable ranges did not alter the results quantitatively. All parameter choices were guided by computational and robustness considerations and were fixed for all the reported experiments.

The experiments were repeated on all the series. The data points for each excess return series was divided into two sets for experimentation as in Section 2: a *Training set* (90% of the data) and a *Test set* (10%). Furthermore, to control the search and reduce the risk of over-fitting, the Training set was further subdivided into the following subsets: *Training1* (45% of the Training set), *Validation1* (also 45%) and *Validation2* (10%).

The experiments on the two data series were identical. They were carefully designed to ensure that if good ANN structures were available, they would be found and their weights reliably estimated. In view of the results, this approach turns out to be particularly useful. Each experiment consisted of three stages as shown in Fig. 2. The aim of the first stage was to select promising ANN structures. A GA was allowed to search the space of ANN structures, train each structure for 25 epochs on the Training1 subset and then evaluate each structure using the Validation1 subset. All four evaluation metrics were used in the process, i.e.

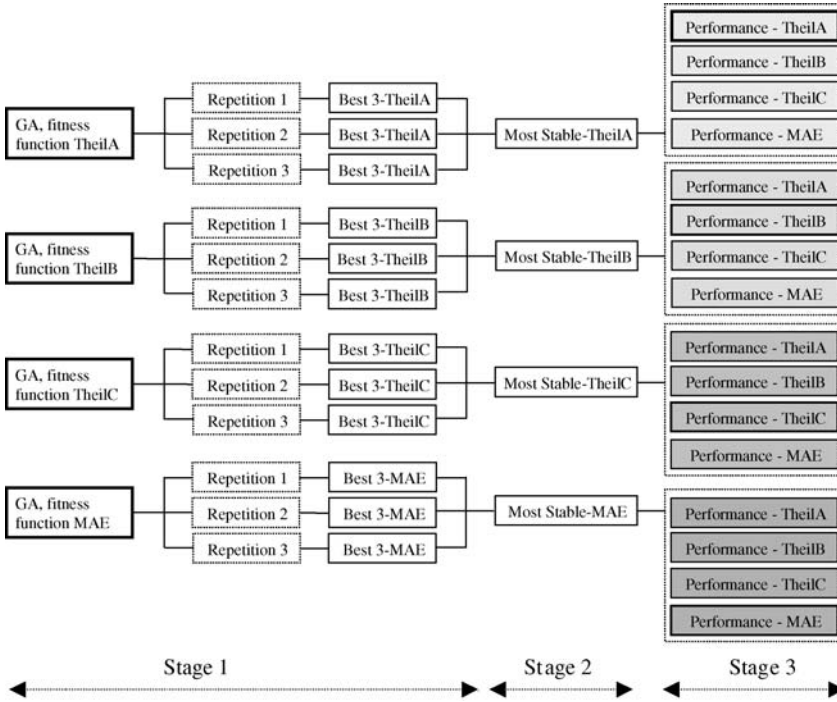


Fig. 2. Experimental Stages.

TheilA, TheilB, TheilC and MAE. For each of the four metrics the GA search was repeated three times. For each of these three repetitions, the best three networks were selected for further investigation, giving $3 \times 3 = 9$ network structures, and for each evaluation metric $4 \times 9 = 36$ network structures in total (see Fig. 2), all of which turned out to be slightly different, as expected.

The purpose of the second stage was to select between these 36 network structures, using the robustness of each structure when repeatedly evaluated. For each one of the 36 ANNs, the following procedure was repeated 50 times, each time starting with random initial weights. Thus, to determine the appropriate number of training epochs, each ANN was initially repeatedly trained on the Training1 subset and evaluated on Validation1 subset, using the same metric as had been used to select the ANN in the first stage, until performance on the Validation1 set deteriorated (see Fig. 2). The optimal number of training epochs was chosen at this training stage. Then the ANN was retrained on the data from the Training1 and Validation1 subsets together, using the indicated optimal number of epochs. Finally, the performance of the each trained ANN was tested on the unseen data

set in the Validation2 subset. For each performance statistic, the most stable of the 9 ANNs were identified in terms of the standard deviation of the performance metric over the 50 repetitions. The repetitions and test procedures at this stage were intended to ensure a robust set of results as the GA randomly combines operators and variables during estimation, and as it can occasionally get caught in local minimum.

The purpose of the third and final phase of the experiment was to repeatedly evaluate each of the four surviving structures against each of the four metrics, i.e. not only on the metric against which it was originally chosen, but also the other three metrics as well. Again the following procedure was applied 50 times. Each of the four networks was initially repeatedly trained on the first half of the Training set and evaluated on the remaining half, using the same metric as had been used to select it in the first place. Then, using the number of epochs indicated by this validation procedure, the network was retrained on the complete Training set. Finally each network was evaluated on the Test set against all four metrics. The performance for each network on each metric was measured again in terms of the standard deviation and mean of its performance over the 50 times that each network was trained, validated and tested. These statistics reflect the average characteristics of the population and can therefore be expected to reflect the overall patterns in the data sets.

5. MAIN EXPERIMENTAL RESULTS

We report the experimental results for both excess return series in parallel. Each of the three stages of the experiments is considered sequentially. The first stage seeks to identify reasonable architectures for the ANN models. In so doing, we consider the four metrics that are used to evaluate the ANNs. The second and third stages examine the stability of the best structures under repeated estimation of their parameters. To anticipate our findings the experimental results are not strong but serve a useful reminder to researchers intent on doing future work in this area.

5.1. Stage 1: TheilA Metric for the First Repetition

Recall that the purpose of the first stage of each experiment is to select promising ANN structures using the GA search. The structures are coded by the number of nodes in the input layer (max 20), two hidden layers (max 30 each) and the single node in the output layer.

The use of three repetitions of the GA search for the four metrics of both data sets, gave us a total set of 24 *different* outputs of results. To illustrate, we briefly focus on the structures chosen (but not presented) when using the TheilA metric in the first repetition.[13] For the S&P series, 10 ANN structures tended to be present in most cases over all 25 generations. Of course, a given structure may be present more than once in any generation although it may not necessarily be present in the final generation. All the most frequently occurring structures had 27 nodes in the first hidden layer. This is remarkably high suggesting that the number of dimensions needed to characterize the input data is also high. This seems to be the case even though the structures in the final generation had only one or two input nodes. To some extent, the same is true for the FTSE series. Here, the most frequently occurring node in the 25th generation had 16 first hidden layer nodes. For both sets of data the number of nodes in the second hidden layer was generally one or three, suggesting that the second hidden layer of nodes may not be necessary.

For the S&P series, the best structure at the final generation of the first repetition (not shown) was 2-27-3-1, where the structure 2-27-3-1 takes on our definition of x-y-z-1 as before (see Section 4). TheilA had a mean, μ of 0.997 and standard deviation, σ of 0.016. Similarly, the best structure for the FTSE series (not shown) was 1-7-3-1 (TheilA: $\mu = 1.005$, $\sigma = 0.007$). TheilA is not less than one in any case so the RW has outperformed the ANN forecasts. These preliminary findings suggest that the GA search is not finding structures any better than would be found by chance, despite earlier evidence of non-linearity in the data! We will now look at all the data for the first stage to see if this conclusion holds.

5.2. Stage 1: Comparison of Evaluation Metrics

Table 4 shows the mean and standard deviation of all four metrics for the structures selected at the first, the 12th and the final 25th generations of the GA search, for all three repetitions and for both series. The magnitude of the mean and standard deviation of the TheilA metric for the S&P for all structures are of concern. As expected, the mean has decreased but only in some cases and any decrease is not significant and not below one. Moreover, the standard deviation has not generally decreased. Indeed, the standard deviations of the 25th generation for the first and third repetitions are larger than those of the first generation in each case.

Four conclusions are clear from the overall results of Table 4: firstly, there is little variation between repetitions, which points to the difficulty that the GA has in choosing between structures but also adds credence to the robustness of the results. Secondly, there is an improvement in the mean value for each of the metrics

Table 4. Summary Statistics of Fitness Functions at Specific Generations.

Generation	Repetition 1			Repetition 2			Repetition 3		
	1	12	25	1	12	25	1	12	25
Panel A: S&P excess returns									
TheilA									
Average	1.0310	1.0122	1.0076	1.0463	1.0558	1.0009	1.0342	1.0073	1.0409
S.D.	0.0369	0.0558	0.0410	0.0902	0.2404	0.0164	0.0503	0.0245	0.2651
TheilB									
Average	0.7118	0.7045	0.6834	0.7172	0.6874	0.6941	0.7106	0.6895	0.6874
S.D.	0.0338	0.0839	0.0172	0.0402	0.0162	0.0251	0.0437	0.0283	0.0233
TheilC									
Average	1.1508	1.0302	1.0025	1.0324	1.0064	1.0042	1.0320	1.0006	1.0010
S.D.	0.4590	0.0394	0.0271	0.0566	0.0306	0.0221	0.0450	0.0176	0.0183
MAE									
Average	0.0105	0.0103	0.0101	0.0103	0.0102	0.0102	0.0104	0.0102	0.0103
S.D.	0.0004	0.0003	0.0001	0.0003	0.0003	0.0003	0.0005	0.0003	0.0006
Panel B: FTSE excess returns									
TheilA									
Average	1.0868	1.0129	1.0099	1.0411	1.0108	1.0116	1.0457	1.0120	1.0068
S.D.	0.2040	0.0272	0.0125	0.0445	0.0162	0.0222	0.0444	0.0183	0.0142
TheilB									
Average	0.7707	0.7337	0.7313	0.7713	0.7335	0.7346	0.7842	0.7329	0.7343
S.D.	0.0490	0.0091	0.0106	0.0444	0.0087	0.0121	0.1512	0.0095	0.0151
TheilC									
Average	1.1355	1.0110	1.0043	1.0384	1.0290	1.0114	1.0568	1.0081	1.0039
S.D.	0.4040	0.0168	0.0136	0.0382	0.0372	0.0156	0.0835	0.0088	0.0123
MAE									
Average	0.0090	0.0088	0.0087	0.0090	0.0087	0.0087	0.0092	0.0091	0.0088
S.D.	0.0004	0.0002	0.0001	0.0002	0.0002	0.0001	0.0012	0.0190	0.0002

Note: S.D. denotes the standard deviation. The mean and standard deviation were computed for the fitness functions over each generation. The samples span the period 4th January 1988 to 6th October 1999.

between the first and final generations (except for repetition three of TheilA on S&P) although more so from the first to the 12th generations. So the GA search is functioning as best it can, given the data. Thirdly, this improvement is not statistically significant. So the search space is indeed difficult for the GA – a feature that we will explore below. Fourthly, while TheilA and TheilC are each close to one, TheilB is considerably better – in the region of 0.70 – and significantly better than the RW for all repetitions (except the third repetition of the first generation

for the FTSE). The magnitude of the MAE appears to be small; however, there is no independent benchmark against which this metric can be compared.

The explanation for the result that ANNs outperform the RW according to the TheilB metric can be found in the definition of this metric (see Eq. (5)). The TheilB statistic would be one if investors predicted no price changes, in which case they would keep their funds in Treasury bills. So, investing in stocks during the 12-year training period covered by the two data sets would appear to pay off, not surprisingly.

5.3. Stage 1: In-Sample Dynamics

As indicated, the average size of TheilA and TheilC for the structures in the final generation is much higher than expected. One possible reason for this might be that the structures were generated randomly by the GA. To validate this view, we first examine the dynamics of the mean and standard deviation of the four metrics. These results are shown in Figs 3a and b. The plots show fluctuations in both the mean and standard deviations of the metrics for all repetitions. While the standard deviations are drawn symmetrically either side of the mean, the results themselves do not fluctuate symmetrically, as we will see below. In most cases, there is a slight and slow decrease in the mean value. The standard deviation, however, fluctuates rather than steadily decreases, indicating that we need to be wary that the last generation could contain some recently introduced inappropriate structures. The third repetition of the TheilA metric for the S&P data is, indeed, an instance in which this problem appears to arise. But otherwise, even though there are a few high standard deviations just before the last generation, in general, the searches have settled down.

A high standard deviation, even in the last generation, would not necessarily be damaging since our method requires us to select just the best three structures from each repetition. But it is also possible that a good structure was found in an earlier generation but was then discarded. The descriptive statistics associated with the figures indicate that the minimum metric value over all generations is better than the average of all the structures in the final generation. These minimum values occur in the final generation only once (see MAE of FTSE series; third repetition). This does not mean that the corresponding structures do not occur in the 25th generation, since the value of the metric depends on the training weights found and these vary for every instantiation of the structure, i.e. not only within one generation but also across generations. Moreover, these minimum values are not significantly different from the mean of the forecast measures for the final generation in each case (see Table 4).

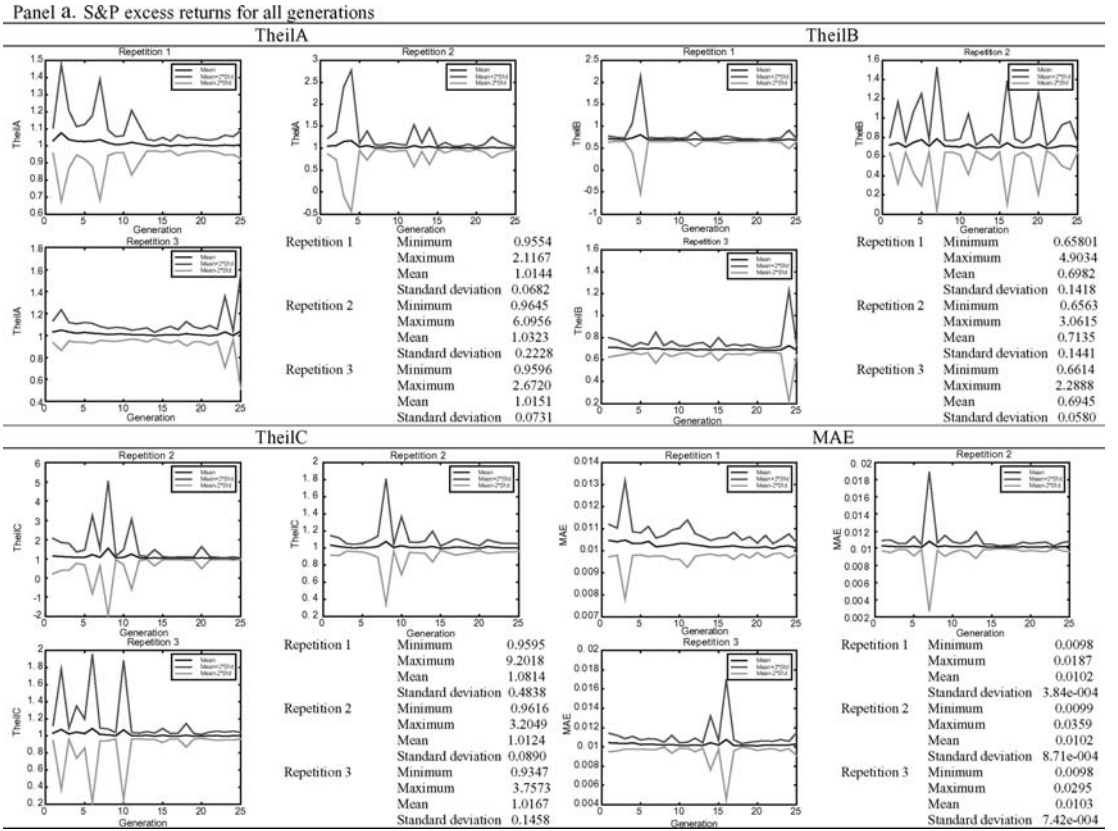
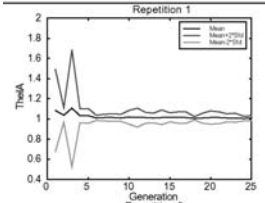
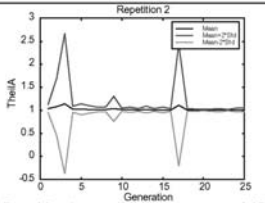


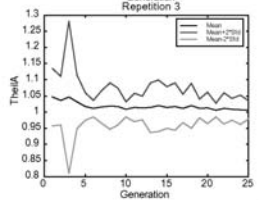
Fig. 3. Plots and Summary Statistics of Forecast Measures for 25 Generations of the (a) S&P Excess Returns and (b) FTSE Excess Returns. (Continued on next page)

Panel b. FTSE excess returns for all generations

TheilA

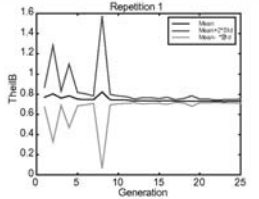


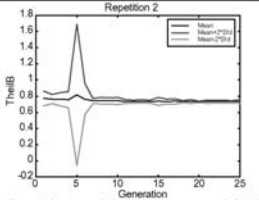


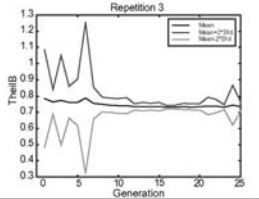


Repetition 1	Minimum	0.9767
	Maximum	2.5227
	Mean	1.0221
	Standard deviation	0.0782
Repetition 2	Minimum	0.9896
	Maximum	5.8549
	Mean	1.0266
	Standard deviation	0.2147
Repetition 3	Minimum	0.9840
	Maximum	1.7400
	Mean	1.0171
	Standard deviation	0.0375

TheilB

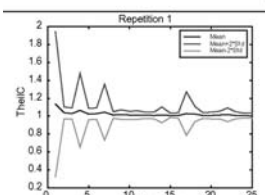


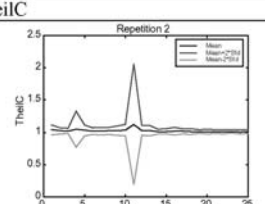


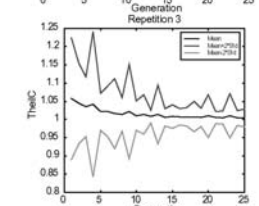


Repetition 1	Minimum	0.7168
	Maximum	3.1139
	Mean	0.7467
	Standard deviation	0.0981
Repetition 2	Minimum	0.7168
	Maximum	3.5140
	Mean	0.7441
	Standard deviation	0.0929
Repetition 3	Minimum	0.7142
	Maximum	2.2183
	Mean	0.7454
	Standard deviation	0.0694

TheilC

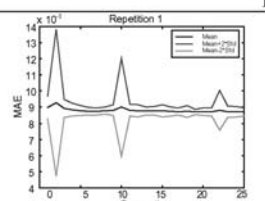


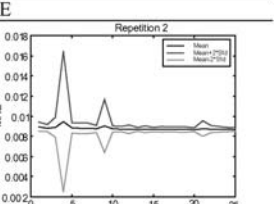


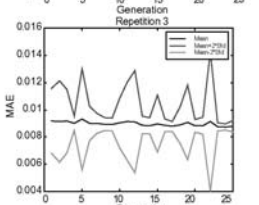


Repetition 1	Minimum	0.9847
	Maximum	3.2399
	Mean	1.0239
	Standard deviation	0.1038
Repetition 2	Minimum	0.9858
	Maximum	3.3702
	Mean	1.0252
	Standard deviation	0.10253
Repetition 3	Minimum	0.9830
	Maximum	1.6318
	Mean	1.0158
	Standard deviation	0.0403

MAE







Repetition 1	Minimum	0.0085
	Maximum	0.0232
	Mean	0.0088
	Standard deviation	5.91e-004
Repetition 2	Minimum	0.0085
	Maximum	0.0298
	Mean	0.0088
	Standard deviation	7.81e-004
Repetition 3	Minimum	0.0085
	Maximum	0.0250
	Mean	0.0090
	Standard deviation	0.001e-004

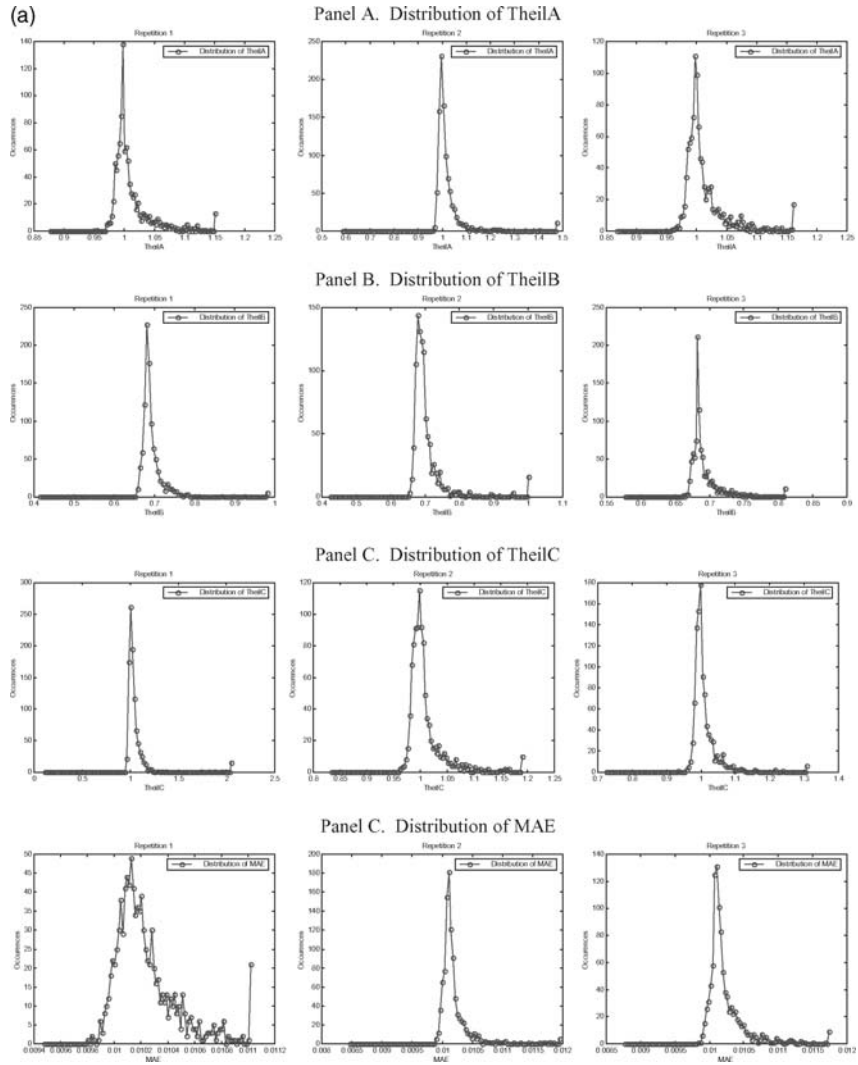


Fig. 4. Distribution of Forecast Measures for (a) S&P Excess Returns and (b) FTSE Excess Returns.

To get an idea of the space in which the GA was searching, we plotted histograms of the number of times each metric value was encountered in all 25 generations for each repetition separately. These are shown in Figs 4a and b for each series. All the histograms are very similar (except for one) in being highly skewed, with

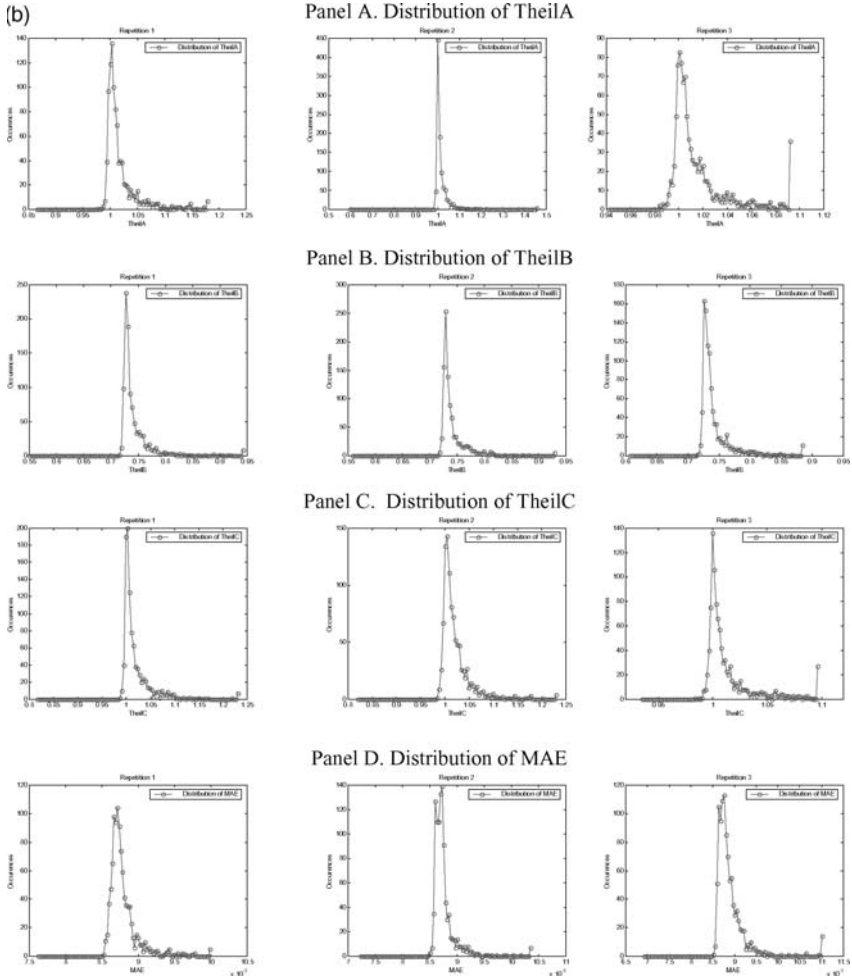


Fig. 4. (*Continued*)

a peakedness close to the lowest (best) metric value. The one exception is the first repetition of the MAE for the S&P series. Here, the plots show a broader distribution as well as a few structures with low values (<0.0098) which are not present in the 25th generation (see Table 5). Otherwise, the best structures are still present in the last generation. These results suggest that continuing the GA search beyond the 25th generation would not have improved either the mean of the forecast measures or the best structure.

Table 5. Performance of Fittest Networks Identified by GA.

Repetition	TheilA			TheilB			TheilC			MAE		
	Network	Mean	S.D.	Network	Mean	S.D.	Network	Mean	S.D.	Network	Mean	S.D.
Panel A: S&P excess returns												
1	2-27-3-1	0.9973	0.0163	5-8-1-1	0.6860	0.0104	7-13-7-1	1.0180	0.1209	6-26-4-1	0.0101	0.0002
1	2-27-1-1	0.9997	0.0134	5-8-4-1	0.6799	0.0177	7-17-7-1	1.0017	0.0281	6-9-4-1	0.0101	0.0001
1	**1-27-1-1**	**0.9996**	**0.0102**	5-14-4-1	0.6776	0.0104	7-26-7-1	0.9991	0.0231	6-22-4-1	0.0104	0.0002
2	3-8-3-1	0.9967	0.0163	2-24-5-1	0.6941	0.0207	6-4-7-1	0.9999	0.0258	1-22-2-1	0.0101	0.0001
2	3-28-3-1	1.0032	0.0306	5-6-5-1	0.6775	0.0106	3-4-7-1	1.0028	0.0204	1-7-2-1	0.0102	0.0003
2	3-8-6-1	1.0157	0.0563	5-24-5-1	0.6814	0.0176	6-4-9-1	1.0015	0.0175	1-22-1-1	0.0101	0.0001
3	6-7-2-1	0.9939	0.0158	1-8-1-1	0.6831	0.0071	6-3-3-1	0.9997	0.0221	3-19-3-1	0.0102	0.0003
3	6-7-4-1	0.9921	0.0151	**1-1-1-1**	**0.6848**	**0.0058**	**6-1-3-1**	**0.9972**	**0.0089**	**3-2-3-1**	**0.0101**	**0.0001**
3	6-10-2-1	0.9941	0.0133	1-6-1-1	0.6830	0.0103	6-8-3-1	0.9992	0.0227	3-2-9-1	0.0103	0.0008
Panel B: FTSE excess returns												
1	1-16-3-1	1.0082	0.0196	4-4-1-1	0.7281	0.0076	14-3-2-1	1.0065	0.0120	6-22-2-1	0.0087	0.0002
1	1-7-3-1	1.0049	0.0073	4-20-1-1	0.7313	0.0150	5-3-2-1	1.0022	0.0086	6-3-2-1	0.0087	0.0001
1	4-16-3-1	1.0149	0.0223	**7-4-1-1**	**0.7296**	**0.0028**	1-3-1-1	1.0018	0.0050	6-9-2-1	0.0087	0.0001
2	7-3-1-1	1.0023	0.0060	1-4-1-1	0.7317	0.0131	3-1-5-1	1.0072	0.0150	4-10-1-1	0.0087	0.0001
2	7-21-1-1	1.0076	0.0177	8-4-2-1	0.7346	0.0543	8-7-5-1	1.0160	0.0207	4-5-1-1	0.0087	0.0001
2	**4-3-1-1**	**1.0017**	**0.0040**	8-27-1-1	0.7287	0.0079	3-7-5-1	1.0134	0.0139	4-24-1-1	0.0087	0.0001
3	18-3-2-1	1.0105	0.0202	12-4-2-1	0.7311	0.0076	3-2-1-1	1.0014	0.0041	**9-2-1-1**	**0.0087**	**0.0001**
3	4-3-2-1	1.0049	0.0093	6-4-2-1	0.7307	0.0059	7-8-1-1	1.0036	0.0051	9-6-1-1	0.0087	0.0002
3	18-1-2-1	1.0049	0.0070	3-4-2-1	0.7341	0.0110	**1-2-1-1**	**1.0002**	**0.0027**	17-2-5-1	0.0087	0.0001

Note: S.D. denotes the standard deviation. The networks shown are for lowest fitness functions and together with their standard deviation. Each network was trained 50 times. The samples span the period 4th January 1988 to 6th October 1999.

5.4. Stage 2: In-Sample Validation

This section deals with the validation of the chosen structures. The three structures that had the most representatives in the final generation of stage 1 were selected for each of the three repetitions on all four metrics. If there was a tie, the tie was broken by selecting the structure that was visited the most frequently by the GA over all generations. If this did not break the tie, the simplest structure was chosen. The rationale for this selection process is that the structures that occur frequently and survive are indeed the best structures. The structures chosen are characterized by having at most five and generally one or three nodes in the second hidden layer, indicating that this second hidden layer may not be needed (see Table 5). The number of nodes in the first hidden layer varies considerably. In a few cases, the number of nodes is up to 27 (close to the maximum), thereby indicating the high dimensionality of the space. The number of lags, given by the number of nodes in the input layer, varies as well, generally being below 10 although one structure uses 18, close to the maximum of 20.

Each of these structures was trained, validated, retrained and then tested on the Validation2 subset, 50 times. The mean and standard deviation of the parameters of each of the $3 \times 3 = 9$ ANN structures are also shown in Table 5. The structures with the lowest standard deviation are shown in bold face. The mean values for both TheilA and TheilC are still close to one (as before) for all structures while TheilB shows improved performance. For all three Theil metrics, the structure with lowest standard deviation has a single input node for both the S&P and FTSE series. This is particularly obvious for TheilB of the S&P, where the structure is 1-1-1-1. For all three metrics, the best structure is not best by much. The large variation in the structures together with TheilA and TheilC values of around one indicates that ANNs are not capable of predicting the series sufficiently well to give an excess return over the Treasury bill rate. To confirm this conclusion, we continue with stage 3 to see whether or not the chosen structures cannot, after all, be improved by a better choice of parameter weights.

5.5. Stage 3: Out-of-Sample Forecasts

In this stage of the experiment, the best structure from the set of nine structures was further tested against all four metrics using the unseen Test sets (300 data points each). This generated the out-of-sample forecasts. Each of the 4×2 Test sets $= 8$ structures was trained anew. This new round of training for each structure consisted of training on the first half of the appropriate Training set, validating on the other half to choose the number of learning epochs, and finally retraining

Table 6. Out-of Sample Forecasting Performance of Fittest Networks for All Fitness Functions.

Network	TheilA		TheilB		TheilC		MAE	
	Mean	S.D.	Mean	S.D.	Mean	S.D.	Mean	S.D.
Panel A: S&P excess returns								
1-27-1-1	0.9991	0.0059	0.7045	0.0042	0.9990	0.0059	0.0104	0.0001
1-1-1-1	1.0032	0.0092	0.7074	0.0065	1.0031	0.0092	0.0104	0.0001
6-1-3-1	1.0087	0.0289	0.7093	0.0203	1.0086	0.0289	0.0105	0.0003
3-2-3-1	1.0013	0.0199	0.7063	0.0140	1.0011	0.0199	0.0104	0.0002
Panel B: FTSE excess returns								
4-3-1-1	1.0008	0.0049	0.7247	0.0036	1.0008	0.0049	0.0086	0.0001
7-4-1-1	1.0017	0.0122	0.7208	0.0088	1.0019	0.0122	0.0086	0.0001
1-2-1-1	1.0031	0.0170	0.7248	0.0123	1.0031	0.0170	0.0086	0.0002
9-2-1-1	1.0013	0.0052	0.7205	0.0038	1.0013	0.0052	0.0086	0.0001

Note: S.D. denotes the standard deviation. The forecasts are for the last 300 observations.

on the entire Training set. The newly trained ANN was then tested against all four metrics. This procedure was repeated 50 times. The resulting statistics for the out-of-sample forecasts are shown in Table 6. The results show that the mean and standard deviation of the metrics are very similar to those obtained at stage 2.

Training using a Theil metric and testing against another Theil metric seems to make little difference to the results. Training on a Theil metric and testing using the MAE metric confirms earlier results. Training on the MAE metric and testing on a Theil does not give appreciably worse results. In general, the performance of the networks has not altered in any meaningful way.

To further explore the above results, we identified an AR(1) linear model based on the AIC. This model generated out-of-sample forecast metrics that are very close to those of the ANNs. Indeed, TheilB was equal to 0.7001 and 0.7226 for the S&P and FTSE series respectively while both TheilA and TheilC were close to one.[14] So the performance of the AR(1) model is not appreciably better than those of the ANNs and those results further confirm that the forecasts are not good for any estimation method.

6. CONCLUSION, EXTENSIONS AND IMPLICATIONS

This empirical study was concerned with the use of GAs in finding the *optimal* architectures for ANN input. We sought to apply rigorous procedures in our

research design so that we can generalize the results more forcefully. Non-linearity was observed in the excess returns of both series, implying that prediction was theoretically possible. The structures that were found by the GA search were repeatedly trained, validated and tested against four different forecast metrics. The structure with the least variance against each metric was again repeatedly trained and tested against a Test data set. TheilB generated a value of around 0.71 with σ of around 0.01 (for the Test set), which is better than the RW forecast and appears to suggest that an ANN had been found that was a good predictor. This is however not the case as TheilB is a RW based on constant prices, a RW which is too easy to beat. It also shows that predicting prices rather than excess returns would have been too easy. Both mean values for TheilA and TheilC were close to or equal to one while the MAE was small but its relative importance is difficult to assess. The overall results show that it is unlikely that one day excess returns can be predicted, using ANNs with simply the excess return time series as inputs.

Is it possible that there were ANN structures that would have performed better, but which were not found by the GA searches? The total search space for ANN structures was set at 20×30^2, i.e. 18,000. With a population size of 40 and 25 epochs, each GA search evaluated at most 1000 of these structures, but most probably a few hundred. However, each GA search was repeated three times, so our guess is that around 1000 ANN structures were evaluated for each metric, i.e. 1 in 18. For the GA search to have missed a much better ANN structure the surface of the excess returns would have to be very uneven, spiking at one or more critical values of the ANN structure parameters. In view of the remarkably smooth nature of the space actually explored by the GA, this seems unlikely.

It could be argued that the size of the Test set (about 14 months worth of daily data) was too small for testing purposes. However, the in-sample forecasts which are from a much longer series are no better. Certainly if positive results had been obtained, it would have been reasonable to validate the result by testing on a longer series. However, in view of the negative result, the only criticism could be that this particular 14 months, mostly in the year 2000, was unusually difficult to predict in light of the 10 previous year's worth of data. There is no reason to believe that this is so; the major decline in the markets did not begin until 2001.

The results of the above experiments lead us to the following conclusions:

(i) The TheilB metric, based on constant prices, should not be used as the RW when evaluating prediction models; it is too easy to beat using any model, including the other RW models. Either of the other RW models should be used instead: price rises drifting upwards to just compensate for the risk free rate of return (TheilA) or tomorrow's price change being the same as yesterday's (TheilC).

(ii) The finding that the GAs converge slowly leaves us with the view that network structures as good as those identified could have been obtained by a very simple random search;
(iii) As the networks that were generated were unable to outperform the RW, except for the TheilB metric, there are useful implications regarding the suitability of ANNs for finding patterns in daily excess returns and to forecast such series.
(iv) Unfortunately, no firm conclusion can be drawn regarding the suitability of GAs for finding suitable ANN structures, because there weren't any such structures.[15]

The finding that the simplest network structures performed (marginally) better is consistent with related work. Indeed, it is well known that simple mis-specified linear models tend to outperform complex well specified models in out-of-sample forecasts. It can be argued that the observed non-linearity in the series does not mean that the series are predictable. This is because the non-linearity may not exist over a sufficiently long period for it to be captured by the ANNs. This echoes some of the explanations provided by Kilian and Taylor (2003) for the failure of the non-linear models to outperform the RW. So the algorithms may not have been able to capture and track the local patterns in a meaningful way, given also the dynamics of the series.

We believe that the inability of the networks to beat the RW is associated with the difficulty that networks have in recognizing temporally local patterns in the data. Perhaps, a methodology that eliminates more noise in the data might prove useful.[16] However, it should be recognized that treating the data for noise would remove features that are inherent to the data and would alter the conditions under which the algorithms learn. If ANN structures are to be found that exploit the lack of iid in daily excess returns, then it may well require different structures for different types of period, e.g. for stable vs. volatile periods in the time series. Indeed, we do not put forward the alternative argument that the failure to beat the RW is because the stock market is efficient since we have acknowledged the specific conditions under which market (in)efficiency will apply.

A secondary lesson of this study is that it is important both to repeatedly estimate the weights of a single ANN structure, using training and validation, in order to find the most stable structure, and also to repeatedly retrain the weights of the best structure(s) for out-of-sample testing. This is the approach we have emphasized in our experiments. Although the use of an ANN structure with weights estimate just once is not uncommon in related experimental work, this approach should be discouraged. The main lesson from this study is the importance of fully exploiting the evolution of the data and comparing forecasting performance using

no change forecast benchmarks. The use of linear models to compare the relative forecasting performance of ANNs, as is often done in empirical work, should also be discouraged.

NOTES

1. GAs are computer intensive search techniques – pioneered by Holland (1975) – based on Darwin's theory of natural selection. Under a GA search the encodings are of fixed length character strings. Koza (1992) extended Holland's work to include explicit hierarchical variable length character strings. Given the nature of our research problem we focus on GAs rather than Genetic Programming in this study.

2. Recently, Conti et al. (2000) put forward a learning algorithm within an ANN framework that minimizes the statistical averages of the error rather than their nominal values. This in turn, takes weight tolerances into account, thereby guaranteeing low sensitivity to the initial random weights.

3. Their GA search converged on a single ANN structure after just 19 generations. The structure contained 5 lagged inputs and 15 hidden nodes in a single layer. Using rules based on information criteria, their linear models had 5 lagged inputs and the equivalent of one hidden unit. The use of GAs to identify the appropriate architecture for ANNs is not typically found in empirical work in finance and economics. Indeed, Schaffer et al. (1992) provide a review of experimental work in several (other) research areas while Sexton and Sikander (2001) provide experimental results based on the Henon mapping problem. Both ANNs and GAs generally appear to work well in those and other settings (see also, Arifovic, 1996; Indro et al., 1999; Noe & Pi, 2000).

4. Following Adya and Collopy's (1998) review, most studies compare the forecasting performance of the ANNs against some linear model in terms of a MSE cost function rather than a no change forecast such as a RW. This is a major failing of prior ANN applications in finance and economics although an exception is Haefke and Helmenstein's (1996) work which is concerned with initial public offerings. If an ANN outperforms a linear model this does not mean that the forecasts are useful if indeed better forecasts could have been obtained from a naïve RW.

5. The economic importance of the use of excess return forecasts can be illustrated as follows. Consider an investor who holds a risky stock portfolio but can also switch to a risk-free investment (say) Treasury bills on receiving a signal that it is economically beneficial to do so. The choice of the investment strategy will depend on the investor's forecast of the amount by which the return from the stock portfolio is expected to exceed the risk-free rate, given also transaction costs and the investor's level of risk aversion. Excess returns are also an important measure in asset pricing models.

6. Several studies (see, e.g. Pesaran & Timmermann, 1994) show that the extent to which investors can benefit from switching between risky and the riskless assets following a signal from the excess return forecasts depends on the frequency of the trading rule that is applied.

7. Empirical work shows that a substantial proportion of monthly, quarterly and annual variation in excess stock returns is predictable when estimated in conjunction with macroeconomic explanatory variables (see, e.g. Balvers et al., 1990). Except for Pesaran and Timmermann's (1994) study, the predictions are typically not truly ex ante in such

experimental work as they are based on observations over the entire sample period. Most macroeconomic variables do not exist at high frequency.

8. As there are 300 data points for the out-of-sample forecasts (about 14 months), this can be considered to be long given the number of data points. It is well-known that the performance of forecast models declines the longer the forecast horizon. This is because forecast models are unable to anticipate future disequilibrium or structural breaks within the sample parameters of their own estimation period. So a larger test set is likely to generate worse forecasts than those reported here. Since we will show that both the in- and out-of-sample forecast are not good, we do not believe that the use of a larger test or estimation set would necessarily improve on the forecast results that are presented.

9. Our preview that the forecast results *might* be negative was based on a small experimentation using a basic GA-ANN structure. We were initially surprised by this preliminary finding since most prior applications of ANNs and indeed GA-ANNs (see, e.g. Arifovic & Gençay, 2001) suggest that good forecasts can be found, albeit such applications used different data sets and forecast benchmarks. In view of our initial findings and the results of existing empirical work, we felt that it was essential that a very rigorous methodological approach was adopted to confirm the reliability of the experimental results.

10. We have noted (see Note 4) that one of the failings of many ANN applications in finance and economics is that the RW forecast is not used in most forecast evaluations and many of the forecast results are likely to be affected by over-fitting due to the failure to apply appropriate validation criteria.

11. Throughout we will use this format to report the results of the ANN structures, but in integer terms. Normally, a ANN with no second hidden layer would be represented as *x*-*y*-0-1, i.e. setting *z* to 0. For technical reasons to do with the automatic training of the ANNs and the GA search, the actual minimum value of *z* had to be set to 1 rather than 0. A structure of the form *x*-*y*-1-1 can be treated as having no second layer of hidden nodes as the output of the single node in the second hidden layer is the only value sent to the output node. A structure of the form *x*-1-1-1 is equivalent to a tansigmoid function applied to the output of a linear model.

12. Man et al. (1999) also suggest that for a large population size of $m = 100$, $P_{\text{crossover}} = 0.60$ and $P_{\text{mutation}} = 0.001$ work well, while for small population size ($m = 30$), $P_{\text{crossover}} = 0.90$ and $P_{\text{mutation}} = 0.01$ are acceptable values.

13. Understandably, a large body of numerical results was generated from the experiments. All experimental results not fully presented can be obtained from the authors.

14. Haefke and Helmenstein (1996) also identified an ANN structure that generated an out-of-sample Theil statistic (similar to TheilC) of 0.896 for the Austrian Traded Index stock (raw) return. Their AR(1) linear model generated a Theil statistic that exceeded one. Notice that in both our study and Haefke and Helmenstein's (1996) study, the ANN and linear models are both being assessed against the forecast performance of a RW.

15. This has since been confirmed. A complete search for all possible ANN structures for predicting the Dow and FTSE indices over similar time periods failed to find any structures that could outperform the RW. We are indebted to Robert Woolfson, University of Manchester, for this empirical result.

16. Kaboudan (2001) shows that the performance of Genetic Programming is adversely affected by the structural complexity of the DGP as well as the level of noise contained in the data. Arguably, our use of GAs rather than Genetic Programming would not appear to alter our results if indeed the excess returns also contained noise. However, to see if the

level of noise in the data adversely impacted on the choice of the architecture and in turn the performance of the ANNs, we reduced the level of noise in each series as follows. Each excess return was recoded as zero if its value was between the mean excess return of the series and ± 1 standard deviation. Values greater than the mean plus one standard deviation were coded "+1" and those less than the mean less one standard deviation were coded "−1." The GA was then used to generate the architecture for the ANNs and the ANNs were used to generate the forecasts. The metrics generated values that were very similar to those reported here, thereby giving us more confidence in our earlier results.

REFERENCES

Adya, M., & Collopy, F. (1998). How effective are neural networks at forecasting and prediction? A review and evaluation. *Journal of Forecasting*, *17*, 481–495.

Allen, F., & Karjalainen, R. (1999). Using genetic algorithms to find technical trading rules. *Journal of Financial Economics*, *51*, 245–271.

Altman, E., Marco, G., & Vareto, F. (1994). Corporate distress diagnostic: Comparison using linear discriminant analysis and neural networks. *Journal of Banking and Finance*, *18*, 505–529.

Arifovic, J. (1996). The behaviour of the exchange rate in the genetic algorithm and experimental economies. *Journal of Political Economy*, *104*, 510–541.

Arifovic, J., & Gençay, R. (2001). Using genetic algorithms to select architecture of a feedforward artificial neural network. *Physica A*, *289*, 574–594.

Baker, M. (1998). Fund managers' attitudes to risk and time horizons: The effect of performance benchmarks. *The European Journal of Finance*, *4*, 257–278.

Balvers, R., Cosimano, T., & McDonald, B. (1990). Predicting stock returns in an efficient market. *Journal of Finance*, *55*, 1028–1109.

Barnett, W., Gallant, A., Hinich, M., Jungeilges, J., Kaplan, D., & Jensen, M. (1997). A single-blind controlled competition among tests for non-linearity and chaos. *Journal of Econometrics*, *82*, 157–192 Reprinted in: W. A. Barnett & J. M. Binner (Eds) (2004). *Functional Structure and Approximation in Econometrics* (Chap. 26). Amsterdam: Elsevier/North-Holland.

Bethke, A. (1981). *Genetic algorithms as function optimisers*. Ph.D. dissertation, University of Michigan, Ann Arbor.

Brock, W., Hsieh, D., & LeBaron, B. (1991). *Nonlinear dynamics, chaos, and instability: Statistical theory and economic evidence*. Massachusetts: MIT Press.

Chen, S.-H., Lux, T., & Marchesi, M. (2001). Testing for non-linear structure in an artificial financial market. *Journal of Economic Behavior & Organisation*, *46*, 327–342.

Collopy, F., Adya, M., & Armstrong, J. (1994). Principles for examining predictive validity: The case of information systems spending forecasts. *Information Systems Research*, *5*, 170–179.

Conti, M., Orcioni, S., & Turchetti, C. (2000). Training neural networks to be insensitive to weight random variations. *Neural Networks*, *13*, 125–132.

Curry, B., & Morgan, P. (1997). Neural networks: A need for caution. *Omega*, *25*, 123–133.

de Lima, P. (1998). Non-linearities and nonstationarities in stock returns. *Journal of Business & Statistics*, *16*, 227–236.

Demuth, H., & Beale, M. (1997). *Neural network toolbox: For use with Matlab* (4th ed., 3rd version). US: MathWorks Inc. (Online: http://www.mathworks.com/access/helpdesk/help/toolbox/nnet/nnet.shtml.)

Franke, J., & Neumann, M. (2000). Bootstrapping neutral networks. *Neural Computation*, *12*, 1929–1949.

Gathercole, C., & Ross, P. (1997). Small populations over many generations can beat large populations over few generations in genetic programming. In: J. Koza, J. Deb, M. Dorigo, D. Fogel, H. Iba & R. Riolo (Eds), *Genetic Programming: Proceedings of the Second Annual Conference* (pp. 111–118). San Francisco: Morgan Kaufmann.

Goldberg, D. (1989). *Genetic algorithms in search, optimisation and machine learning*. Reading, MA: Addison-Wesley.

Haefke, C., & Helmenstein, C. (1996). Neural networks in the capital markets: An application to index forecasting. *Computational Economics*, *8*, 37–50.

Holland, J. (1975). *Adaptation in neural and artificial systems*. Ann Arbor, MI: University of Michigan Press.

Hsieh, D. (1989). Testing for non-linear dependence in daily foreign exchange rates. *Journal of Business*, *62*, 339–368.

Indro, D., Jiang, C., Patuwo, B., & Zhang, G. (1999). Predicting mutual fund performance using artificial neural networks. *Omega*, *27*, 373–380.

Kaboudan, M. (2001). Genetically evolved models and normality of the fitted residuals. *Journal of Economic Dynamics & Control*, *25*, 1719–1749.

Kilian, L., & Taylor, M. (2003). Why is it so difficult to beat the random walk forecast of exchange rates? *Journal of International Economics*, *60*, 85–107.

Koza, J. (1992). *Genetic programming: On the programming of computers by means of natural selection*. Cambridge, MA: MIT Press.

LeBaron, B. (1997). *An evolutionary bootstrapping approach to neural network pruning and generalization* (Online: http://people.brandeis.edu/∼blebaron/wps/evolnn.pdf.)

LeBaron, B., Arthur, W., & Palmer, R. (1999). Time series properties of an artificial stock market. *Journal of Economic Dynamics & Control*, *23*, 1487–1516.

LeBaron, B., & Weigend, A. (1994). *Evaluating neural network predictors by bootstrapping* (Online: http://cs.colorado.edu/homes/andreas/public_html/home.html).

Man, F. K., Tang, S. K., & Kwong, S. (1999). *Genetic algorithms: Concepts and designs*. Heidelberg: Springer-Verlag.

Meese, R., & Rogoff, K. (1983). Empirical exchange rate models of the seventies: Do they fit out of sample? *Journal of International Economics*, *14*, 3–24.

Neely, C., Weller, P., & Dittmar, R. (1997). Is technical analysis in the foreign exchange market profitable? A genetic programming approach. *Journal of Financial and Quantitative Analysis*, *32*, 405–426.

Noe, T., & Pi, L. (2000). Learning dynamics, genetic algorithms, and corporate takeovers. *Journal of Economic Dynamics & Control*, *24*, 189–217.

Pesaran, M., & Timmermann, A. (1994). Forecasting stock returns: An examination of stock market timing in the presence of transaction costs. *Journal of Forecasting*, *13*, 335–367.

Qi, M., & Wu, Y. (2003). Non-linear prediction of exchange rates with monetary fundamentals. *Journal of Empirical Finance*, *10*, 623–640.

Schaffer, J., Whitley, D., & Eshelman, L. (1992). Combinations of genetic algorithms and neural networks: A survey of the state of the art. COGANN-92 Combinations of Genetic Algorithms and Neural Networks (pp. 1–37). Los Alamitos, CA: IEEE Computer Society Press.

Sexton, R., Dorsey, R., & Johnson, J. (1998). Toward globalisation optimisation of neural networks: A comparison of the genetic algorithm and backpropagation. *Decision Support Systems*, *22*, 171–185.

Sexton, R., Dorsey, R., & Johnson, J. (1999). Optimisation of neural networks: A comparative analysis of the genetic algorithm and simulated annealing. *European Journal of Operational Research*, *114*, 589–601.

Sexton, R., & Sikander, N. (2001). Data mining using a genetic algorithm – trained neural network. *International Journal of Intelligent Systems in Accounting, Finance & Management*, *10*, 201–210.

Theil, H. (1966). *Applied economic forecasting*. Amsterdam: North-Holland.

Varetto, F. (1998). Genetic algorithms applications in the analysis of insolvency risk. *Journal of Banking & Finance*, *22*, 1421–1439.

CO-EVOLVING NEURAL NETWORKS WITH EVOLUTIONARY STRATEGIES: A NEW APPLICATION TO DIVISIA MONEY

Jane M. Binner, Graham Kendall and Alicia Gazely

ABSTRACT

This work applies state-of-the-art artificial intelligence forecasting methods to provide new evidence of the comparative performance of statistically weighted Divisia indices vis-à-vis their simple sum counterparts in a simple inflation forecasting experiment. We develop a new approach that uses co-evolution (using neural networks and evolutionary strategies) as a predictive tool. This approach is simple to implement yet produces results that outperform stand-alone neural network predictions. Results suggest that superior tracking of inflation is possible for models that employ a Divisia M2 measure of money that has been adjusted to incorporate a learning mechanism to allow individuals to gradually alter their perceptions of the increased productivity of money. Divisia measures of money outperform their simple sum counterparts as macroeconomic indicators.

Applications of Artificial Intelligence in Finance and Economics
Advances in Econometrics, Volume 19, 127–143

ISSN: 0731-9053/doi:10.1016/S0731-9053(04)19005-1

1. INTRODUCTION

Macroeconomic policy-makers aim to promote sustained economic growth and rising living standards. It is now widely accepted that a low rate of inflation must be maintained if these objectives are to be achieved. A standard result of most textbook macroeconomic models which include money and prices is that changes in the money supply lead, eventually, to proportional changes in the price level. Friedman and Schwartz (1982) present a simple analysis of the correlation between U.S money and prices over a span of more than a hundred years while Hallman et al. (1991) provide evidence of a long-run link between M2 and the price level using the P-star model based on the long run quantity theory of money. It appears that the long run causal chain is just as Friedman said it should be, inflation is a monetary phenomenon.

In the United States, the favored monetary aggregate among monetarists, particularly Milton Friedman, during the early to mid 1970s was simple sum M2. Barnett (1997) paints a very clear picture of the monetarist stance in the States in the early 1980s in his description of the "broken road." Forecasts showed that the rise in growth of M2 from under 10% to over 30% between late 1982 and early 1983 was bound to result in renewed stagflation, i.e. recession accompanied by high interest rates and rising inflation. Friedman's very visible forecast failure, according to Barnett (1997), delivered a very "serious blow to 'monetarism' and to advocates of stable simple sum money demand equations."

Central banks around the world became convinced of the importance of money as a policy control variable and confident in the use of monetary aggregates as intermediate monetary targets just at a time when everything started to go embarrassingly wrong. During the mid to late 1970s evidence of the deterioration in the formerly stable demand for money function was beginning to emerge, making the monetarist reign a short one. It was becoming apparent throughout the developed economies in the mid 1980s that increased competition within the banking sector and the computer revolution in the financial world was beginning to have substantial effects on the relative user-costs of bank liabilities and the ever-increasing array of substitutes for them. It is now well established that monetary targeting failed in the major macroeconomies because the chosen target aggregates did not remain stably related to other key macroeconomic variables such as nominal income. Some countries such as the USA and Germany, moved from narrow to broader money targeting in the mid 1980s, before officially abandoning targeting altogether in the late 1980s (e.g. USA and Canada). The Bundesbank kept its monetary goal until the formation of the European Central Bank (ECB) although Svensson (2000) claims that the Bundesbank has been an inflation targeter in deeds and a monetary targeter in words only. The consensus of opinion at the end of the

1980s was that it was not possible to re-establish the former apparently stable demand for money functions, even though broad monetary aggregates had been redefined and extended to include higher interest bearing building society deposits (see Hall et al., 1989).

Recent research into the construction of monetary aggregates (see Barnett, 1997, for the USA) attributes the breakdown in demand for money functions during the 1980s to the use of conventional official simple sum aggregates. Simply summing the constituent component assets to form the aggregate creates flawed index numbers because aggregating any set of commodities with equal weights implies that each good is a perfect substitute for every other good in the group. The simple sum aggregation method will lead to the mismeasurement of the monetary services provided, particularly during periods of significant financial development when interest rate yields on the various components of broad money are changing over time. The use of equal weights for the user costs of the constituent component assets is wholly inappropriate during periods of high financial innovation since the introduction of new instruments and the technological progress that occurred in making transactions has almost certainly have diverse effects on the productivity and liquidity of monetary assets.

Many attempts have been made to improve the measurement of money. The most well known attempt is that of Friedman and Schwartz (1970). They suggested applying some form of weighting of the components in the aggregate depending on their relative "moneyness." The pioneering work of Barnett (1978, 1980) has provided a consistent method to perform this weighting. Economic aggregation theory provides methods for choosing which assets to include in a monetary aggregate and how to construct aggregator functions, whilst index number theory provides parameter- and estimation free methods to perform the aggregation. One index that has particularly good properties for the purpose of constructing monetary quantity indices is the Divisia index, derived from the class of superlative index numbers discussed by Diewert (1976). Much attention has also been devoted to the viability of alternative weighting schemes, such as weighting by bid-ask spreads, turnover rates, price variations and denomination size. The preference for a Divisia monetary index was founded on neoclassical microeconomic theory, approximation theory and revealed preference theory. When applying these theories to the construction of monetary aggregates it becomes apparent that the components included should be weighted depending on the monetary services they provide. It can be shown that traditional simple sum aggregation is only justified when all asset components are perfect substitutes (Barnett, 1984).

Indeed, many economists concede that, in principle, reported simple sum aggregates are flawed and based on untenable assumptions. See, for example, Belongia (1996). In this study using U.S. data, Belongia re-estimated empirical

models by replacing simple sum money with Divisia and thereby significantly altered the conclusions that should have been reached by several influential studies. Similarly, Barnett (1980) showed that an apparent decline in velocity was removed when Divisia measures replaced simple sum. Central banks continue to publish simple sum measures of the money stock and draw policy inferences from their behavior even though it has been demonstrated conclusively that such data violate basic principles of economic and index number theory. Barnett et al. (1992) provide a survey of the relevant literature, whilst Drake et al. (1997) review the construction of Divisia indices and associated problems.

The hypothesis developed over a series of studies (summarized in Gazely & Binner, 2000) is that measures of money constructed using the Divisia index number formulation are superior indicators of monetary conditions when compared to their simple sum counterparts. This hypothesis is reinforced by a growing body of evidence from empirical studies around the world which demonstrate that broad Divisia weighted index number measures may be able to overcome the drawbacks of the simple sum, provided the underlying economic weak separability and linear homogeneity assumptions are satisfied. Whenever new assets are included in Divisia monetary aggregates, the issue of separability must be addressed. Economic theory provides several methods, both parametric and non-parametric, for choosing which assets are admissible for inclusion. The non-parametric (nonpar) approach to demand analysis developed by Varian (1982, 1983) is particularly interesting since there is no need to be specific on the functional form of the utility function. Studies by Swofford and Whitney (1988, 1994) on U.S. data and Belongia (2000) on using data from the U.S., Germany and Japan have applied the nonpar procedure. Varian's non-parametric approach has, however, been heavily criticized in the literature. It has been shown by Barnett and Choi (1989) (using Monte Carlo simulations), that the test results are biased towards rejection. A stochastic extension to the nonpar-procedure has been suggested and is the subject of ongoing research (see, for example, Binner et al., 2002a). In the current work it is assumed that the Divisia monetary aggregates under investigation satisfy the separability assumption.

This paper aims to provide further support for the use of Divisia indices by policy makers and academic economists. The potential of a new generation of Divisia monetary aggregates is explored and adjusted to take account of the recent developments in the financial sector in Taiwan over the period 1970 to date. Ultimately, such evidence could reinstate monetary targeting as an acceptable method of macroeconomic control, including price regulation.

The inflation forecasting potential of the standard Divisia and simple sum indices are compared with that of two new Divisia indices designed to capture the true

user costs of the component assets during times of high financial innovation. Hence the first new Divisia index, inspired initially by Hendry and Ericsson (1990) and used subsequently by Ford et al. (1992), uses a learning adjustment of the retail sight-deposit interest rate to reflect the adaption of agents to the introduction of interest-bearing sight deposits in 1984. The second modified Divisia series assumes a period of gradual and continuous learning by agents as they adapt to the changes in the financial system throughout the period.

The novelty of this paper lies in the use of co-evolution, using neural networks and evolutionary strategies, to examine Taiwan's recent experience of inflation. Our preference for evolutionary strategies is due to their suitability for optimising real valued variables in an economic and financial domain (Dorsey & Mayer, 1995). Other techniques, such as genetic algorithms and simulated annealing do not offer such a natural representation and, in addition, there are more parameters to these approaches which must be selected and tuned. This is a unique tool in this context and its use in this research is highly exploratory although results presented here give us confidence to believe that significant advances in macroeconomic forecasting and policymaking are possible using advanced Artificial Intelligence (AI) methods such as this. We build on the linear ES model reported in Kendall et al. (2001) and compare our results to those already produced for Taiwan using the AI technique of neural networks, Binner et al. (2002b) as a means of evaluating the explanatory power of both Divisia and simple sum measures of broad money as indicators of inflation.

The paper proceeds as follows; it begins by motivating the study with a review of recent financial innovations in Taiwan before describing the data in Section 3. Section 4 introduces the co-evolution model while results of a simple inflation forecasting experiment designed to evaluate the empirical performance of the innovation adjusted Divisia indices compared with the traditional Divisia and simple sum counterparts are presented in Section 5. Section 6 concludes and offers suggestions for further development of this research.

2. FINANCIAL INNOVATION AND THE DIVISIA MONETARY AGGREGATE IN TAIWAN

At the beginning of the 1980s, drastic economic, social and political changes took place in Taiwan creating a long-term macroeconomic imbalance. Rising oil prices caused consumer prices to rise by 16.3% in 1981, followed by a period of near zero inflation in the mid 1980s. From the 1990s onwards, inflation has been fluctuating around the 5% mark and hence the control of inflation has not been the mainstay

of recent economic policy in Taiwan, in contrast to the experience of the western world. Rather, policy in Taiwan has focused more on achieving balanced economic and social development.

The revolution in the financial and monetary sectors of Taiwan over the last two decades has resulted in the implementation of major financial liberalization policies. In July 1987, trade-related foreign exchange controls were abolished and capital-flow related foreign exchange controls were relaxed. The entry of new securities firms was permitted in January 1988, increasing their number from 60 to 150 within the first year. The banking system in Taiwan was heavily regulated by the Central Bank and the Ministry of Finance until September 1989, which saw the introduction of the revised Banking Law. As a result, bank interest rates on deposits and loans were completely liberalized and new private commercial banks became established. Deregulations of financial price variables and market entry resulted in the rapid expansion of local financial markets and interest rates, exchange rates and stock prices becoming increasingly sensitive to market forces. The introduction of new instruments and the increased sophistication associated with transactions technology have almost certainly had a considerable influence on the liquidity and monetary services provided by the component assets of money.

As outlined by Shih (2000), in her pioneering work on the performance of Divisia money aggregates in Taiwan, such financial liberalization had significant impact upon the stability of the monetary aggregates. The narrowly defined aggregate, M1B, was vulnerable to deposit shifts and was thus replaced by the broadly defined monetary aggregate, M2, as the intermediate target variable for monetary policy in Taiwan in 1990. Concerns were expressed as to whether the technique of simply summing the balances of component assets with equal weights can adequately capture the increased productivity of the monetary assets.

This question was taken one step further by Ford et al. (1992, p. 87) who asked "do the Divisia aggregates adequately capture the effects of all these financial innovations?". This question is revisited and the econometric performance of a new generation of Divisia indices that have been reformulated to take account of recent financial innovations in Taiwan is explored. Thus two innovation adjusted Divisia series are analyzed, using data provided by Ford, that have been modified to allow for a learning process by individuals as they adapt to changes in the productivity of monetary assets and adjust their holdings.

- One adjusted series, Innovation1 Divisia, assumes that individuals who had been adjusting well to cosmetic changes in interest rates were slow to react to the increased productivity of money, initially underestimating the effect of financial innovation. In keeping with Ford (1997, p. 21) the approach proposed in Baba et al. (1985, 1990) is adopted, which imposes a learning adjustment

process on the user cost of interest-bearing sight deposits in the construction of monetary indices. Specifically, the learning function used is an ogive-shaped weight function, w_t, representing the agents' learning about the assets and is applied to the interest rates. Thus,

$$w_t = (1 + \exp[\alpha - \beta(t - t^*_{+1})])^{-1} \quad \text{for } t \geq t^* \text{ and } 0 \text{ otherwise.} \qquad (1)$$

Here, t^* denotes the time at which the innovation occurred. α represents the existing level of knowledge about the innovation and β measures the speed to respond to it.

- The second series, namely Innovation2 Divisia, assumes a period of gradual and continuous learning throughout the whole period as individuals adjust to the increased productivity of money. The approach adopted in Ford (1997, p. 4) is used, whereby an approximate estimate of the degree of productivity improvements is obtained by using an index number of bank branches of all kinds (although medium sized business banks are excluded).

In the case of Taiwan financial innovation accelerated around the end of 1989 and the changes are assumed to occur gradually and continuously throughout the years and be assimilated by individuals as they occurred, Innovation2; or a period of learning occurs before individuals adapt to the change of regime, Innovation1.

3. DATA AND MODEL SPECIFICATION

The level of monetary aggregation selected for this study was M2, as this is the measure currently monitored by the monetary authorities in Taiwan. Four different M2 measures were used independently to predict future movements in the inflation rate. Monetary data thus consisted of three Divisia series provided by Ford (1997), one conventional Divisia, (DIVM2), Innovation1 (INN1) and Innovation2 (INN2), together with a simple sum series (M2), constructed from component assets obtained from the Aremos-Financial Services database in Taiwan. The Divisia M2 (DIVM2) aggregate is constructed by weighting each individual component by its own interest rate whilst Innovation1 (INN1) and Innovation2 (INN2) seek to improve upon the weighting system by capturing the true monetary services flow provided by each component asset more accurately. Thus INN1 is a development of DIVM2 and it should be noted that it does not diverge from the conventional Divisia measure until the late 1980s. The second modified Divisia series, INN2 assumes a period of gradual and continuous learning by agents as they adapt to the increased productivity of money throughout the period and corrects, at least partially, for the distortion arising from technological progress. Individuals are thus

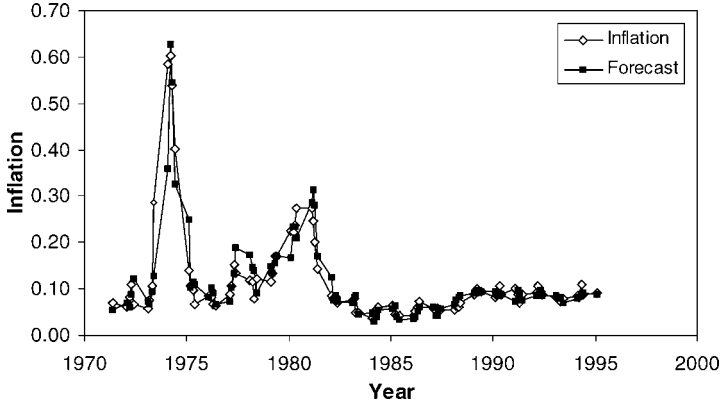


Fig. 1. Inflation and Predicated Inflation Using Innovation1.

assumed to adjust their holdings of financial assets until the diffusion of financial liberalization is complete.

Inflation was constructed for each quarter as year-on-year growth rates of prices. Quarterly data over the sample period 1970Q1–1995Q3 was used as illustrated in Fig. 1. Our preferred price series, the Consumer Price Index (CPI), was obtained from DataStream. The four monetary series were subjected to a smoothing process by taking three quarter averages to reduce noise. Finally, to avoid the swamping of mean percent error (MPE) by huge values during a period of very low inflation from 1983 to 1986, the entire series was translated upwards by 5%. Measures of absolute error (RMS and MAD) are unaffected by this translation and results are presented on this basis. Of the total quarterly data points available, after loss of data points due to the smoothing process and the time lag implicit in the model of up to four quarters, 96 quarters remained, of which the first 85 were used for training and the last 7 were used as a validation set. The first 4 items were only used as a basis for the first prediction.

The aim of the co-evolutionary model is to evolve a neural network that represents the predictive function. In previous work, Kendall et al. (2001), an evolutionary strategy used a linear function. One of the criticisms of the previous work is the use of a linear model. In this work, due to the activation function used within the neural network, it has the ability to evolve a non-linear function.

4. CO-EVOLUTION MODEL

Evolutionary strategies (ES) are closely related to genetic algorithms, with (possibly) some of the same inherent difficulties (Horn & Goldberg, 1994;

Szipro, 2002). Originally evolutionary strategies used only mutation, only used a population of one (i.e. a single individual) and were used to optimise real valued variables. More recently, ES's have used a population size greater than one, they have used crossover and have also been applied to discrete variables (see Back et al., 1991; Herdy, 1991). However, they are extensively used as an optimisation tool for real variables, using mutation, rather than crossover.

An individual in an ES is represented as a pair of real vectors, $v = (x, \sigma)$. The first vector, x, represents a point in the search space and consists of a number of real valued variables. The second vector, σ, represents a vector of standard deviations.

Mutation is performed by replacing x by

$$x_t + 1 = x_t + N(0, \sigma) \tag{2}$$

where $N(0, \sigma)$ is a random Gaussian number with a mean of zero and a standard deviation of σ. This mimics the evolutionary process that small changes occur more often than larger ones.

In evolutionary computation there are two variations with regard to how the new generation is formed. The first, termed $(\mu + \lambda)$, uses μ parents and creates λ offspring. Therefore, after mutation, there will be $\mu + \lambda$ members in the population. All these solutions compete for survival, with the μ best selected as parents for the next generation. An alternative scheme, termed (μ, λ), works by the μ parents producing λ offspring (where $\lambda > \mu$). Only the λ compete for survival. Thus, the parents are completely replaced at each new generation. In this work, we use a $1 + 1$ strategy.

Good introductions to evolutionary strategies can be found in Fogel (1998, 2000) and Michalewicz and Fogel (1996, 2000).

The artificial neural networks we utilise are feed forward networks that have their weights adapted using evolutionary strategies, rather than a supervised learning technique such as back propagation. The networks comprise an input layer, with the neurons within that layer using an identify function. A hidden layer, with a varying number of neurons, is used, with the number of nodes being a matter of experimentation (see Table 1 and Section 4). The network has a single output neuron, which is the predicted inflation rate. The hidden and output neurons use a non-linear activation function, which, in these experiments is either sigmoid (Eq. (3)) or tanh (Eq. (4)). In (3) and (4), x is the input value to the given neuron and a is the slope of the sigmoid function.

$$f(x) = \frac{1}{1 + e^{-ax}} \tag{3}$$

$$f(x) = \frac{e^{x} - e^{-x}}{e + e^{-x}} \tag{4}$$

Table 1. Co-Evolutionary Results Average Errors Over 5 Trials.

	M2	DIVM2	INN1	INN2
Within sample				
RMS	0.050759	0.038414	0.039012	0.044366
MAD	0.024788	0.023520	0.023335	0.024782
MPE	20%	22%	21%	21%
Out-of-sample				
RMS	0.035561	0.014817	0.020179	0.151284
MAD	0.031064	0.012219	0.017148	0.128701
MPE	34%	14%	19%	145%

In the experiments a population of 20 neural networks are created. The networks have a fixed structure and the population is homogeneous from the point of view of the structure. The weights are assigned small random values.

Each neural network is presented with the entire set of training data (86 samples) and the output for each sample is compared against the actual inflation rate for that quarter. The fitness for a particular neural network is then given by

$$\sum_{i=1}^{86}(t_i - o_i)^2 \tag{5}$$

where $t =$ the training value presented to the network and $o =$ the actual output from the network.The aim is to minimize Eq. (5) and the final network (after the entire evolutionary process) that produces the minimum value is the one that is used on the validation set (10 samples).

Once all 20 neural networks have had the training data presented, they are sorted based on their fitness (Eq. (5)) and the best 10 are selected to survive to the next generation. These are copied (in order to maintain a population size of 20) and the copied networks have their weights mutated in order to try and improve their predictive ability. This is done using Eq. (2). In addition, the value of σ is also mutated using Eq. (6).

$$\sigma = \sigma^* \exp(\tau^* N(0, \sigma)) \tag{6}$$

where $\tau = (2(N_W)^{0.5})^{-0.5}$ and $N_W =$ The number of weights in the neural network.

This process continues for 1,000,000 iterations. One million iterations was chosen as a good trade off between time taken (about one hour on a standard desktop PC (1.5GHZ)) and to give the possibility of a suitably good neural network evolving. There is a danger that we could "overfit" the data (Lawrence et al., 1997). That is, the validation set can produce a larger error than the training data. We did

carry out some experiments, in order to find out if this was the case, and we do not believe that this is happening.

In this work we evolve a population of neural networks which are evaluated by considering how well the network can predict the test cases in the data (in sample). Once evolution has completed the best individual (i.e. evolved neural network) is tested to see how well it can predict the data that it has not seen before (out of sample). The input supplied to the neural network is the four previous quarters from the money supply currently being tested (i.e. M2, DIVM2, INN1, INN2) and an autoregressive term in the form of the previous months inflation figure. The network has one output, a prediction of the next quarters inflation rate.

A population of 20 networks was randomly created and, after evaluating them, the top 10 are retained and evolved using an evolutionary strategy. In addition to evolving the weights in the network, the sigma value (that is, the standard deviation value used in the mutation operator) is also evolved. Sigma is initially set to 0.05.

Various experiments were conducted. The number of hidden neurons was varied between 3 and 5 and sigmoid and tanh activation functions were used in the hidden layer (an identify function was used for the input and output layer). Each test consisted of 1,000,000 iterations so as to be comparable with the results reported in Kendall et al. (2001).

In summary, the various parameters we used are as follows:

- Measures: {M2, DIVM2, INN1, INN2);
- Population Size of Networks: 20;
- Iterations: 1,000,000;
- Networks Retained and Mutated: 10;
- Input Neurons: 5;
- Hidden Neurons: {3,4,5};
- Activation fn (hidden): {sigmoid, tanh};
- Activation fn (input/output): identity;
- All results averaged over 6 runs.

Therefore, we conducted 120 (|Measures| × |Hidden Neurons| × |Activations fn (hidden)| × Averaged over six runs) runs, each of 1,000,000 iterations.

5. TESTING AND RESULTS

The four money measures (M2, DIVM2, INN1 and INN2) were tested independently and these results compared against previous results obtained on the same data using a neural network approach, but with some slight variations

Table 2. Co-Evolutionary Results for Best-Fit Model.

	M2	DIVM2	INN1	INN2
Within Sample				
RMS	0.050696	0.040938	0.041234	0.048419
MAD	0.024900	0.024000	0.023000	0.026300
MPE	20%	22%	20%	23%
Out-of-Sample				
RMS	0.017289	0.013175	0.011312	0.080092
MAD	0.013800	0.012000	0.008300	0.078500
MPE	15%	14%	9%	89%

Note: All these results used the tanh activation function with five hidden neurons. However, the other models gave similar results.

(see Binner et al., 2002b, for a full description of the neural network procedure employed). The results reported here are the arithmetic means calculated over six individual trials of the co-evolutionary approach and are divided between within sample (the training set) and out-of-sample (the validation set). Within these two categories, three standard forecasting evaluation measures were used to compare the predicted inflation rate with the actual inflation rate, namely, Root Mean Squared Error (RMS), Mean Absolute Difference (MAD) and Mean Percent Error (MPE). The in-sample and out-of-sample results produced by the co-evolutionary approach averaged over six trials are shown in Table 1. These six trials represent varying the number of hidden neurons (3) and the hidden layer activation function (2). These two parameters are used when testing all the Divisia measures (M2, DIVM2, INN1, INN2). The best fitting model is shown in Table 2. This trial represents 3 hidden neurons and using the tanh activation function. Previous results, using neural networks are shown in Table 3. Results for trial 6 only are presented

Table 3. Comparison with Neural Network Results.

	M2	DIVM2	INN1	INN2
Within Sample				
RMS	0.032106	0.022578	0.018764	0.026806
MAD	0.024700	0.017500	0.013900	0.018200
MPE	30%	22%	16%	21%
Out-of-Sample				
RMS	0.014801	0.016043	0.010715	0.011575
MAD	0.013800	0.015000	0.00800	0.009000
MPE	16%	17%	9%	10%

Table 4. Forecasts from a Random Walk.

Within Sample	Inflation
RMS	0.0428365
MAD	2.12%
MPE	63.24%
Out-of-Sample	
RMS	0.013972
MAD	1.02%
MPE	26.79%

for reasons of brevity, although the pattern of findings is consistent across all six trials performed. Table 4 presents results obtained from a simple random walk model for comparative purposes.

A comparison of Tables 1 and 3 reveals that co-evolution clearly competes favorably with the neural network, on average, in terms of forecasting capabilities across all forecasting evaluation methods both in- and out-of sample. When the results of the best-fitting co-evolutionary model are considered, however, using trial 6 presented here in Table 2, the co-evolutionary method produces forecasts equal to or superior to the neural network in eight out of 12 out-of-sample cases analysed. This result is representative of all six co-evolutionary trials performed. The best inflation forecast is achieved using the INN1 monetary aggregate, where the co-evolutionary approach RMS error is 14% lower than that achieved for Divisia M2 and 34% lower using forecasts from the simple sum M2 model. Figures 1 and 2

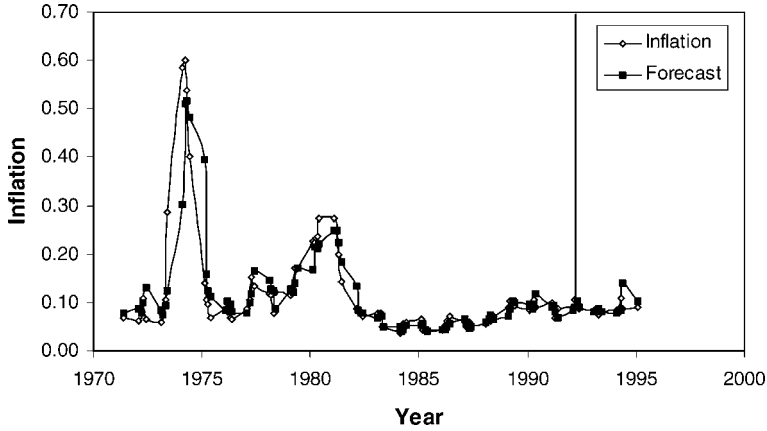


Fig. 2. Inflation and Predicated Inflation Using Simple Sum M2.

illustrate the best fitting (INN1) and worst fitting (M2) forecasts for the co-evolutionary technique. On average, evidence presented in Table 1 clearly indicates that both INN1 and standard Divisia M2 outperform the simple sum M2 counterpart in all cases out-of-sample. INN2 is undoubtedly the worst performing aggregate, producing out-of-sample RMS errors some 7.5 times greater than INN1 on average.

Comparison of Tables 3 and 4 reveal that the best fitting neural network RMS errors (INN1) are 44% lower than the equivalent random walk forecast within sample and 23% lower than the random walk forecast out of sample. In the case of the co-evolutionary model, RMS errors for the best fitting model (INN1) are 4% lower than the random walk forecast within sample and 19% lower than the random walk forecast out of sample.

6. CONCLUDING REMARKS

This research provides a significant improvement upon Kendall et al. (2001) in terms of comparative predictive performance of co-evolution, which has been found to compete very favourably with neural networks and has the potential to beat neural networks in terms of superior predictive performance when co-evolution is used to evolve neural networks. Artificial Intelligence techniques in general and co-evolution in particular are highly effective tools for predicting future movements in inflation; there is tremendous scope for further research into the development of these methods as new macroeconomic forecasting models.

The evidence presented here provides overwhelming support for the view that Divisia indices are superior to their simple sum counterparts as macroeconomic indicators. It may be concluded that a money stock mismeasurement problem exists and that the technique of simply summing assets in the formation of monetary aggregates is inherently flawed. The role of monetary aggregates in the major economies today has largely been relegated to one of a leading indicator of economic activity, along with a range of other macroeconomic variables. However, further empirical work on Divisia money and, in particular, close monitoring of Divisia constructs that have been adjusted to accommodate financial innovation, may serve to restore confidence in former well established money-inflation links. Ultimately, it is hoped that money may be re-established as an effective macroeconomic policy tool in its own right. This application of evolutionary strategies to explore the money – inflation link is highly experimental in nature and the overriding feature of this research is very much one of simplicity. It is virtually certain in this context that more accurate inflation forecasting models could be achieved with the inclusion of additional explanatory variables, particularly those

currently used by monetary authorities around the world as leading indicator components of inflation.

ACKNOWLEDGMENT

The authors gratefully acknowledge the help of Prof Jim Ford at the University of Birmingham for providing the Innovation Adjusted Divisia data.

REFERENCES

Baba, Y., Hendry, D. F., & Starr, R. M. (1985). A stable U.S. money demand function, 1960–1984, Paper presented at the Fifth World Congress of the Econometric Society. Cambridge, MA, USA, August.

Baba, Y., Hendry, D. F., & Starr, R. M. (1990). *The demand for M1 in the USA, 1960–1988, mimeo*. San Diego, USA: Department of Economics, University of California.

Back, T., Hoffmeister, F., & Schwefel, H.-P. (1991). A survey of evolution strategies. In: R. Belew & L. Booker (Eds), *Proceedings of the Fourth Conference on Genetic Algorithms* (pp. 2–9). San Mateo, CA: Morgan Kaufmann.

Barnett, W. A. (1978). The user cost of money. *Economic Letters*, *1*, 145–149. Reprinted in: W. A. Barnett & A. Serletis (Eds), *The Theory of Monetary Aggregation* (Chap. 1, 2000, pp. 6–10). Amsterdam: North Holland.

Barnett, W. A. (1980). Economic monetary aggregates; An application of index number and aggregation theory. *Journal of Econometrics*, *14*(1), 11–48. Reprinted in: W. A. Barnett & A. Serletis (Eds), *The Theory of Monetary Aggregation* (Chap. 2, 2000, pp. 11–48). Amsterdam: North Holland.

Barnett, W. A. (1984). Recent monetary policy and the divisia monetary aggregates. *The American Statistician*, *38*, 165–172. Reprinted in: W. A. Barnett & A. Serletis (Eds), *The Theory of Monetary Aggregation* (Chap. 23, 2000, pp. 563–576). Amsterdam: North Holland.

Barnett, W. A. (1997). Which road leads to stable money demand? *Economic Journal*, *107*, 1171–1185. Reprinted in: W. A. Barnett & A. Serletis (Eds), *The Theory of Monetary Aggregation* (Chap. 24, 2000, pp. 577–592). Amsterdam: North Holland.

Barnett, W. A., & Choi, S. (1989). A Monte Carlo study of blockwise weak separability. *Journal of Business and Economic Statistics*, *7*, 363–377.

Barnett, W. A., Fisher, D., & Serletis, A. (1992). Consumer theory and the demand for money. *Journal of Economic Literature*, *30*, 2086–2119. Reprinted in: W. A. Barnett & A. Serletis (Eds), *The Theory of Monetary Aggregation* (Chap. 18, 2000, pp. 389–430). Amsterdam: North Holland.

Belongia, M. T. (1996). Measurement matters: Recent results from monetary economics re-examined. *Journal of Political Economy*, *104*(5), 1065–1083.

Belongia, M. T. (2000). Consequences of money stock mismeasurement: Evidence from three countries. In: M. T. Belongia & J. M. Binner (Eds), *Divisia Monetary Aggregates: Theory and Practice* (Chap. 13, pp. 292–312). Basingstoke, UK: Palgrave.

Binner, J. M., Elger, T., & dePeretti, P. (2002a). Is UK risky money weakly separable? A stochastic approach, Department of Economics, Lund University, Working Paper. S-woPEc No. 13.

Binner, J. M., Gazely, A. M., & Chen, S. H. (2002b). Financial innovation in Taiwan: An application of neural networks to the broad money aggregates. *European Journal of Finance*, *8*(2), 238–247.

Diewert, W. E. (1976). Exact and superlative index numbers. *Journal of Econometrics*, *4*(2), 115–145.

Dorsey, R. E., & Mayer, W. J. (1995). Genetic algorithms for estimation problems with multiple optima, nondifferentiability and other irregular features. *Journal of Business and Economics Statistics*, *13*, 53–66.

Drake, L., Mullineux, A. W., & Agung, J. (1997). One divisia money for Europe. *Applied Economics*, *29*(6), 775–786.

Fogel, D. B. (1998). *Evolutionary computation the fossil record*. IEEE Press.

Fogel, D. B. (2000). *Evolutionary computation: Toward a new philosophy of machine intelligence* (2nd ed.). IEEE Press.

Ford, J. L. (1997). Output, the price level, broad money, and Divisia aggregates with and without innovation: Taiwan, 1967(1)–1995(4). Discussion paper 97–17. Department of Economics, the University of Birmingham, UK.

Ford, J. L., Peng, W. S., & Mullineux, A. W. (1992). Financial innovation and Divisia monetary aggregates. *Oxford Bulletin of Economics and Statistics*, 87–102.

Friedman, M., & Schwartz, A. J. (1970). *Monetary statistics of the United States*. New York, USA: Columbia University Press.

Friedman, M., & Schwartz, A. J. (1982). *Monetary trends in the United States and in the United Kingdom*. Chicago, USA: University of Chicago Press.

Gazely, A. M., & Binner, J. M. (2000). The application of neural networks to the Divisia index debate: Evidence from three countries. *Applied Economics*, *32*, 1607–1615.

Hall, S. G., Hendry, S. G. B., & Wilcox, J. B. (1989). The long run determination of the U.K. monetary aggregates. Bank of England Discussion Paper, No. 41, August.

Hallman, J. J., Porter, R. D., & Small, D. H. (1991). Is the price level tied to the M2 monetary aggregate in the long run? *American Economic Review*, *81*(4).

Hendry, D. F., & Ericsson, N. R. (1990). Modelling the demand for narrow money in the U.K. and the United States, Board of Governors of the Federal Reserve System, International Finance Discussion Papers, No. 383.

Herdy, M. (1991). Application of the evolution strategy to discrete optimization problems. Proceedings of the First International Conference on Parallel Problem Solving from Nature (PPSN). In: H.-P. Schwefel & R. Manner (Eds), *Lecture Notes in Computer Science* (Vol. 496, pp. 188–192). Springer-Verlag.

Horn, J., & Goldberg, D. (1994). Genetic algorithm difficulty and the modality of fitness landscapes. In: *Proceedings of Foundations of Genetic Algorithms (FOGA) 3*. Workshop held July 30 to August 2. Colarado, USA.

Kendall, G., Binner, J. M., & Gazely, A. M. (2001). Evolutionary strategies vs. neural networks: An inflation forecasting experiment. In: H. R. Arabnia (Ed.), *Proceedings of the International Conference on Artificial Intelligence* (*IC'AI*) (pp. 609–615, ISBN 1-892512-79-3). Las Vegas Nevada USA: CSREA Press.

Lawrence, S., Giles, C. L., & Tsoi, A. C. (1997). Lessons in neural network training: Overfitting may be harder than expected. In: *Proceedings of the Fourteenth National Conference on Artificial Intelligence* (*AAAI-97*) (pp. 540–545). Menlo Park, CA: AAAI Press.

Michalewicz, Z., & Fogel, D. B. (1996). *Genetic algorithms + data structures = evolution programs* (3rd revision and extended ed.). Berlin: Springer-Verlag.

Michalewicz, Z., & Fogel, D. B. (2000). *How to solve it*. ISBN 3-540-66061-5. Springer-Verlag.

Shih, Y. C. (2000). Divisia monetary aggregates for Taiwan. In: M. T. Belongia & J. M. Binner (Eds), *Divisia Monetary Aggregates; Theory and Practice* (Chap. 10, pp. 227–248). Basingstoke, UK: Palgrave.

Svensson, L. (2000). The first year of the Eurosystem: Inflation targeting or not? NBER Working Paper, W7598.

Swofford, J. L., & Whitney, G. A. (1988). A comparison of non-parametric tests of weak separability for annual and quarterly data on consumption, leisure and money. *Journal of Business Economics and Statistics*, *6*, 241–246.

Swofford, J. L., & Whitney, G. A. (1994). A revealed preference test for weakly separable utility maximization with incomplete adjustment. *Journal of Econometrics*, *60*(1–2), 235–249.

Szipro, G. G. (2002). Tinkering with genetic algorithms: Forecasting and data mining in finance and economics. In: S.-H. Chen (Ed.), *Evolutionary Computation in Economics and Finance*. Heidelberg: Physica Verlag.

Varian, H. R. (1982). The non-parametric approach to demand analysis. *Econometrica*, *50*, 945–974.

Varian, H. R. (1983). Non-parametric tests of consumer behavior. *Review of Economic Studies*, *50*, 99–110.

FORECASTING THE EMU INFLATION RATE: LINEAR ECONOMETRIC VS. NON-LINEAR COMPUTATIONAL MODELS USING GENETIC NEURAL FUZZY SYSTEMS

Stefan Kooths, Timo Mitze and Eric Ringhut

ABSTRACT

This paper compares the predictive power of linear econometric and non-linear computational models for forecasting the inflation rate in the European Monetary Union (EMU). Various models of both types are developed using different monetary and real activity indicators. They are compared according to a battery of parametric and non-parametric test statistics to measure their performance in one- and four-step ahead forecasts of quarterly data. Using genetic-neural fuzzy systems we find the computational approach superior to some degree and show how to combine both techniques successfully.

1. INTRODUCTION

Inflation forecasts are highly significant for economic and political agents. Central banks are a user of forecasts as they need insight into probable future inflation rates

Applications of Artificial Intelligence in Finance and Economics
Advances in Econometrics, Volume 19, 145–173

ISSN: 0731-9053/doi:10.1016/S0731-9053(04)19006-3

when setting their policy instruments which affect economic behaviour with a time-lag. Especially those central banks which run an inflation targeting regime require accurate inflation forecasts. A sound foundation for monetary policy decisions requires insight into the processes responsible for inflation. Ideally, an empirical model should be available that produces reliable conditional inflation forecasts, thus indicating how to adjust policy instruments in order to keep inflation in conformity with the definition of price stability. Erroneous inflation forecasts lead to poor policy decisions that may impact negatively on real output and employment. Equivalent to central banks, private agents also have a clear need for accurate inflation forecasts, because they require them for such purposes as wage negotiations (labour unions and employers) or for designing credit contracts (financial institutions).

However, forecasting inflation reliably remains a difficult task and vast resources are invested in estimation activities that yield an acceptably accurate and rational forecast output. In empirical economics, a broad variety of models has been applied, ranging from univariate to multiple regression models (time series methods such as univariate ARIMA or multivariate VAR models, as well as theory-driven econometric specifications ranging from reduced form representations to structural multi-equation systems). Moreover, artificial intelligent systems have been developed and used within complex environments and opaque systems. Recently, seminal comparative work has been conducted on the predictive power of many commonly used econometric procedures and Artificial Neural Networks (e.g. Rech, 2002; Swansson & White, 1997; Zhang et al., 1998).

In addition to the technique-driven search for optimal forecasting methods, the lack of consensus as to which economic theory – if any – best derives indicator variables for predicting movements in prices, is also highly relevant when testing various forecasting approaches. Typically, such indicators are based on macroeconomic concepts such as the Pstar model (e.g. Gerlach & Svensson, 2003; Nicoletti Altimari, 2001), mark-up pricing (e.g. Banerjee et al., 2001; De Brouwer & Ericsson, 1998) or classical final demand specifications underlying the Phillips curve (e.g. Stock & Watson, 1999). However, findings as to the best predictor of price movements are specific to country data and sample choice, resulting in rather mixed empirical evidence. For U.S. data Stock and Watson (1999) found real activity variables derived from the final demand Phillips curve best suited to explain and forecast inflation for the last decades, while similar forecasting systems yield superior results for monetary aggregates predicting inflation for the EMU area (e.g. Nicoletti Altimari, 2001).

Moreover, when evaluating inflation indicators based on long-run regressions, fairly eclectic models are generally used in the literature, that allocate different levels of significance to different indicators in subsequent periods and therefore

treat inflation as a general phenomenon deriving from different sources (e.g. Fic, 2003; Hendry, 2001; Kim, 2001). The advantage of eclectic models is that they make use of a number of variables that need not stem from one a-priori chosen theoretical concept alone. Thus, the potential for empirical investigation is widened, while the risk of possible model misspecifications is reduced.

The scope of this paper extends to three dimensions. The first dimension is methodical: we seek to determine whether the predictive power of dynamic single-equation, linear econometric models outperform models based on a novel computational approach using genetic-neural fuzzy rule-bases when forecasting the EMU inflation rate. The latter approach is a non-linear, fully interpretable procedure that is implemented with the software GENEFER (GEnetic NEural Fuzzy explorER, www.genefer.de). The software automatically constructs, simplifies and tunes a fuzzy rule-base FRB, on the basis of a training database, by means of Genetic Algorithms and Neural Networks. Since linear econometric models are by far the most commonly applied inflation predictors, they can be seen as established competitor for the novel computational approach.

The second dimension accounts for the variety of (competing) inflation theories for constructing econometric and computational models. This facilitates a reconsideration of the ongoing dispute among economists as to which theory best explains inflation, when deriving leading indicators of price level changes. The forecasts are evaluated according to a broad battery of test statistics, which can be seen as a third dimension of our analysis. We distinguish between parametric and non-parametric evaluation criteria in order to evaluate, which indicator-model-combination best predicts the general direction and pattern changes or the absolute level of inflation. The advantage of combining multiple dimensions in this competition set-up is that – since we evaluate the applied forecasting techniques on a broader and more balanced modelling basis with respect to a variety of test statistics – the results are more robust with respect to each dimension. In this way, we can provide sound recommendations with respect to: (1) the superior forecasting technique; and (2) the best indicator identification procedure.

The remainder of the paper is organised as follows: We describe the empirical data set in Section 2. In Section 3 we derive a set of inflation indicators linked to the outline of the econometric forecasting technique. The presentation of the computational approach follows in Section 4. The forecasting set-up and evaluation criteria based on parametric and non-parametric test statistics are discussed in Section 5. In Section 6 we present and analyse out-of-sample forecast results for various models of both estimation techniques. Finally, Section 7 summarises the key findings and provides suggestions for further research.

2. DATA SET

The time series used in the forecasting experiments are mainly from an ECB study by Fagan et al. (2001).[1] The database contains time series on a quarterly basis from the first quarter of 1980 until the fourth quarter of 2000 (in total 80 observations). All series are aggregated data for an area-wide model (AWM) of the EMU based on the EU11. The inflation rate is defined as the quarter-to-quarter change of an artificially constructed harmonized consumer price index, using a fixed set of weights for each country. Beside the AWM database the time series of oil price changes (Δpoil) is from the World Market Monitor (WMM) database, based on the spot market price for oil. The ECU/U.S.$ nominal exchange rate is given by Eurostat (obtained via Datastream with Mnemonic: USESXECU. The exchange rate is originally the Dollar to ECU rate and was converted for the purpose of this paper). The change in energy prices (Δpenergy) as well as the Bundesbank's inflation objective stem from a study of Svensson and Gerlach (2003).[2] The latter is used to derive a time series for an implicit central bank inflation objective for the EMU area, yielding inflation expectations as one of the models' driving forces. All time series were transformed into natural logarithms.

Since the main purpose of this paper is a technical comparison between different estimation techniques, we do not pay explicit attention to the historical circumstances of the chosen sample. For example, problems in the European exchange rate mechanism in the beginning of the 1990s, indicated by a break in long-run velocity are expected to worsen the fit of monetary indicators. Further research should consider these issues more carefully. Also, one should bear in mind that the database is an artificial construction and may be biased due to the calculation method. However, until longer time series for the EMU are available, the use of constructed data can be seen as the most promising approach for modelling and forecasting purposes.

We divided the database into an in-sample or training subset from 2/82[3] to 4/96 (59 observations) and an out-of-sample subset from 1/97 until 4/2000. All models are exclusively estimated with respect to the former, whereas the latter is used for evaluation purposes only.

3. ECONOMETRIC FORECASTING TECHNIQUE

In terms of econometrics GENEFER is most similar to a single-equation approach. Therefore, we drop VARs and structural multi-equation systems when contrasting the predictive power of standard econometric tools with the computational approach.

To derive a set of single equation inflation models, we apply (multi)cointegration analysis based on a two-stage modelling strategy that is closely related to the well-known Engle and Granger (1987) methodology: At the first stage, particular cointegration relationships are estimated separately to obtain leading indicators capturing deviations from a stable long-run relationship between the aggregate price level, real activity and monetary variables. At the second stage, the calculated error-correction terms are inserted into the final inflation equation as a vector of explanatory variables.

Defining equilibrium as a stable long-run cointegration relation, a measure of disequilibrium in a particular market can be derived as a deviation of the price level from its normal path. For modelling purposes, we use a dynamic single-equation autoregressive distributed-lag approach, as supposed by Pesaran and Shin (1995), of the following general ARDL($q_0, q_1, \ldots, q_n$) form[4]

$$y_t = \alpha_0 + \sum_{i=1}^{q_0} \alpha_{1,i} y_{t-i} + \sum_{i=0}^{q_1} \alpha_{2,i} x_{1,t-i} + \cdots + \sum_{i=0}^{q_n} \alpha_{n,i} x_{n,t-i} + \varepsilon_t, \qquad (1)$$

where y is the dependent variable, which is regressed on q_0 lags of its own and on q_1 to q_n lags of n explanatory variables x. α_0 to α_n are coefficients, with α_0 being a constant, ε_t is an (iid) disturbance term and t is a time index. From (1) the long-run coefficients for all exogenous variables can be calculated based on the assumption that in a long-run equilibrium, there is no tendency for change such that $y_t = y_{t-1} = y_{t-2} = \cdots = y^{lr}$. Using these long-run characteristics, we can calculate the long-run coefficients as

$$y^{lr} = \frac{\alpha_0}{1 - \sum_{i=1}^{q_0} \alpha_{1,i}} + \frac{\sum_{i=0}^{q_1} \alpha_{2,i}}{1 - \sum_{i=1}^{q_0} \alpha_{1,i}} x_1^{lr} + \cdots + \frac{\sum_{i=0}^{q_n} \alpha_{n,i}}{1 - \sum_{i=1}^{q_0} \alpha_{1,i}} x_n^{lr} \qquad (2)$$

Using OLS, we start estimating a general model including a maximum lag length of 5 for all variables. We therefore use an algorithm that estimates a total of $(q+1)^{n+1}$ models (with $q = 5$ for the maximum number of lags and n as the number of variables) and choose the best specification according to the Schwartz-Bayesian (SBC) information criterion (e.g. Pesaran & Shin, 1995).

From (2), we can calculate error-correction terms (ect) as the difference of actual and long-run values of ect $= (y - y^{lr})$. They capture the deviation from a long-run cointegration path and are used as leading indicators for future inflation. At the second stage, we estimate various models based on the derived ec-terms using the general-to-specific modelling approach. That is, we start from a rather general model and subsequently exclude statistically insignificant variables to obtain a more parsimonious form. The general set-up for the inflation model can be

written as:

$$\pi_t = \sum_{i=1}^{4} \alpha_i D_{t,i} + \sum_{n=1}^{N} \sum_{n=1}^{T} \phi_{n,i} \mathrm{ect}_{n,t-i} + \sum_{m=1}^{M} \sum_{j=0}^{T} \eta_{m,j} z_{m,t-j} + \zeta_t. \qquad (3)$$

Equation (3) specifies the inflation rate (π_t), which is defined as $\pi_t \equiv p_t - p_{t-1}$ (with p = price index), in terms of the variables on the right with $z_1, \ldots, z_M$ = other exogenous (short-run) variables, D_i = deterministic seasonal, α, ϕ, η = coefficients, ζ = residual. Inserting more than one error correction term into the equation constitutes the main difference with regard to the commonly used error-correction model.

When deriving the error-correction terms we concentrate on different models based on the augmented Phillips curve theory assigning different weights to demand pressure variables in production factors, final goods and monetary aggregates as:

$$\mathrm{markup} = (\mathrm{mup}_t - \mathrm{mup}_t^*) \quad \text{with: } \mathrm{mup}^* = \mathrm{f(ulc, pim)} \qquad (4)$$

$$\mathrm{ygap} = (y_t - y_t^*) \quad \text{with : } y^* = \mathrm{f(trend)} \qquad (5)$$

$$\mathrm{mgap} = (\mathrm{mr}_t - \mathrm{mr}_t^*) \quad \text{with : } \mathrm{mr}^* = \mathrm{f}(y^*, i^*). \qquad (6)$$

Equation (4) relates inflationary pressure to deviations of marked-up prices over marginal costs (mup) from their equilibrium level (mup*), recasting the markup model of inflation.[5] In the markup model, mup* is a function of unit labour costs (ulc) and import prices (pim). Turning to indictor models of excess final demand rather than factor demand pressure, we obtain the final demand Phillips curve in (5), relating inflation to deviations of actual output (y) from its equilibrium path (y^*), which is expressed in the output gap (ygap). In (5), we model y^* as a function of a time trend.[6]

Moreover, we can derive a monetarist version of the modified Phillips curve, starting from a quadratic price adjustment cost model (e.g. Roberts, 1995; Rotemberg, 1982; Tödter, 2002) which can be seen as an extension of the standard markup pricing approach. The model results in the real money gap (mgap) as the driving force for inflation. This model relates deviations of real money balances ($\mathrm{mr} = m - p$, with m = nominal money supply) from equilibrium value mr* as a function of equilibrium output (y^*) and interest rates (i^*), thereby recasting the quantity theory of money (e.g. Gerlach & Svensson, 2003). Finally, as suggested by Nicoletti Altimari (2001) we can further use the monetary overhang (monov) as a closely related monetarist indicator to (6). The monov is calculated as:

$$\mathrm{monov} = (\mathrm{mr}_t - \mathrm{mr}_t^{lr}), \quad \mathrm{mr}^{lr} = f(y, i), \qquad (7)$$

Long-run real money balances (mr^{lr}) are estimated as a function of output (y) and nominal interest rates (i).[7]

In order to obtain the ec-terms of (4), (6) and (7), we apply the ARDL model while (5) is estimated as a trend regression. The algorithm estimating an ARDL based on the SBC criterion, yields the following lag-structure: Equilibrium values of marked up prices (mup*) are estimated as an ARDL(4,0,1) model, the mgap* yields an ARDL(5,0) form,[8] the monov results in an ARDL(5,2,0) model with the following, significant long-run coefficients (t-values in brackets):

$$\underset{}{\text{mup}^*} = \underset{(11,05)}{4,2738} + \underset{(22,53)}{0,8812}\,\text{ulc} + \underset{(2,06)}{0,1673}\,\text{pim} \tag{8}$$

$$y^* = \underset{(1187,4)}{6.67} + \underset{(44,42)}{0.006}t \tag{9}$$

$$(m-p)^* = \underset{(-27,44)}{-6,8174} + \underset{(41,83)}{1,4899}y^* \tag{10}$$

$$(m-p)^{lr} = \underset{(-14,01)}{-5,6125} + \underset{(26,78)}{1,3495}y - \underset{(-2,96)}{0,0849}r. \tag{11}$$

We further calculate rather simple disequilibrium terms for external inflationary pressure caused by excess demand in the international goods and financial markets using:

$$e_t^{\text{real}} = 0,362 - (e_t + p_t^f) - p_t \tag{12}$$

$$\varphi_t = \Delta e_t + (r - r^f)_{t-1}, \tag{13}$$

with e^{real} = real exchange rate as deviations from purchasing power parity (PPP, with the sample mean set to zero), φ = deviations from uncovered interest parity (UIP), e = nominal Euro-Dollar exchange rate, Δe = exchange rate change, p^f = foreign price index.

Before we estimate a set of models, we must specify the additional exogenous variables z_M, which initially include an expectational component accounting for the influence of expectation formation on inflation:

$$\pi_t^e = (1-\alpha)\pi_t^{\text{obj}} + \alpha(\pi_{t-1}^{\text{obj}} - \pi_{t-1}), \tag{14}$$

where π_t^e = expected inflation rate, π_t^{obj} = implicit ECB inflation objective, the superscript e denotes expected values. In analogy to the ec-terms, (14) can be seen as a partial adjustment mechanism: Inflation expectations are formed according to the central bank's inflation objective (π_t^{obj}) and a correction factor ($\pi_{t-1}^{\text{obj}} - \pi_{t-1}$) that accounts for lagged deviations of actual inflation from the inflation objective. For details see Gerlach and Svensson (2003). Furthermore, the exogenous variables

Table 1. Econometric Models and In-Sample Fit.

Model	Dependent Variable: $\pi(t)$ List of Explanatory Variable	R^2 (SEE)[a]	Breusch-Godfrey LM Test[b]	ARCH Test[b]	White Test [b]	Jarque-Bera Test[b]	RESET Test[b]
ygap_trend	D, $\pi^e(t)$, ygap_trend$(t-1)$, $\Delta e(t-1)$, Δpoil(t)	0.8864 (−0.0075)	1.795 (0.177)	0.575 (0.451)	1.22 (0.294)	1.72 (0.424)	1.079 (0.347)
ygap_hp	D, $\pi^e(t)$, ygap_hp$(t-1)$, $\Delta e(t-1)$, Δpoil(t)	0.8674 (−0.0081)	2.99 (0.059)	0.006 (0.937)	1.6 (0.113)	1.743 (0.418)	0.657 (0.522)
ygap_cd	D, $\pi^e(t)$, ygap_cd$(t-1)$, $\Delta e(t-1)$, Δpoil(t)	0.8674 (−0.0081)	0.766 (0.470)	0.365 (0.548)	1.921 (0.047)	1.912 (0.384)	0.676 (0.512)
monov	D, $\pi^e(t)$, monov$(t-1)$, $\Delta e(t-1)$, Δpoil(t)	0.8654 (−0.0083)	0.083 (0.919)	0.061 (0.804)	2.163 (0.028)	0.676 (0.713)	0.152 (0.859)
markup	D, $\pi^e(t)$, markup$(t-1)$,$\Delta e(t-1)$, Δpoil(t)	0.8993 (−0.0067)	0.819 (0.446)	1.462 (0.231)	1.408 (0.186)	0.262 (0.877)	0.453 (0.638)
mgap	D, $\pi^e(t)$, mgap$(t-1)$, $\Delta e(t-1)$, Δpoil(t)	0.8433 (−0.0078)	0.05 (0.951)	1.498 (0.226)	1.932 (0.046)	1.483 (0.476)	0.013 (0.986)
eclectic	D, ygap_trend$(t-1)$, mgap$(t-1)$, $\pi^e(t)$, markup$(t-1)$, monov$(t-1)$, $\varphi(t-1)$, Δpenergy$(t-2)$, $\Delta e(t-1)$, Δpoil(t), Δpoil$(t-2)$	0.9425 (−0.0053)	0.829 (0.442)	0.126 (0.723)	2.326 (0.011)	0.189 (0.909)	0.765 (0.386)

[a] Standard Error of Regression.

[b] Prob-values are given in brackets.

(z_m) contain the current and lagged changes in the energy price index (Δpenergy) and the oil price (Δpoil) to account for short-run shocks, as well as lagged changes in the Euro-Dollar exchange rate (Δe). Table 1 presents the in-sample results for the single ec-term models for (8) to (11), as well as an eclectic specification. All derived models perform well in-sample, as indicated by the R^2 in the first column of Table 1. The abbreviations are defined as follows: hp = Hodrick-Prescott (HP)-filter, cd = Cobb-Douglas production function, trend = trend regression.

In order to test for the in-sample properties of the models, we report some standard diagnostic test statistics.[9] We are especially interested in whether or not linear specifications represent the proper form of the model. This question is motivated by our modelling strategy in the next section that uses intelligent system analysis to allow for non-linearities. Thus, if we detect non-linearities in the input/output relationship, we expect superior results when switching to intelligent system models. The explicit testing for models based on monetary aggregates and real activity variables for non-linearities is suggested by Claus (2000) and Nicoletti Altimari (2001).

All models pass the standard tests for normality of the residuals (Jarque-Bera test) and model misspecifications (Ramsey RESET test). Furthermore, we apply more extended tests for neglected non-linearities (McLeod-Li test, BDS test). The McLeod-Li test uses the standard Ljung-Box Portmanteau test with the null hypothesis of no serial correlation to the squared residuals from the derived linear models (e.g. Lee et al., 1993). The BDS-test makes use of the concept of correlation integral and is applied to the estimated residuals, testing the null hypothesis of

Table 2. Results of the McLeod-Li Test.

LB(k)	LB(1)	LB(6)	LB(12)	LB(24)	LB(36)
eclectic	0.00	9.46	16.03	24.20	38.61
	(0.994)	(0.149)	(0.190)	(0.450)	(0.352)
mgap	0.06	6.78	17.84	39.8**	55.53**
	(0.811)	(0.341)	(0.121)	(0.022)	(0.020)
mov	1.17	4.12	4.60	14.81	20.53
	(0.280)	(0.661)	(0.970)	(0.926)	(0.982)
output gap	0.12	3.84	16.09	33.26***	43.87
	(0.727)	(0.698)	(0.187)	(0.099)	(0.200)
mark-up	1.55	5.06	14.82	21.92	35.83
	(0.214)	(0.537)	(0.252)	(0.584)	(0.477)

Note: Prob-values are given in brackets. LB(k) is the Ljung-Box statistic with a lag length of k. The test statistic is asymptotically χ^2-distributed with k degrees of freedom.

** Significance on the 5% significance level.

*** Significance on the 10% significance level.

Table 3. Results of the BDS Test.

M = 2	$\varepsilon = 0.5\sigma$	Critical Value	$\varepsilon = 1\sigma$	Critical Value
eclectic	3.445**	3.36	−2.781**	−2.20
mgap AR(1)-filt.	−3.545**	−3.24	−2.286**	−2.20
monov AR(1)-filt.	−0.398	Non sign.	−1.227	Non sign.
output gap	−7.069*	−4.87	3.903*	3.40
mark-up	−0.775	Non sign.	−0.237	Non sign.

Note: The BDS statistics is a function of two arguments: m as the dimension of the correlation integral and ε as a metric bound. We choose m = 2 as suggested for small sample size and ϵ both as 0.5 and 1 times of the standard deviation (σ) of the underlying series. Having a small sample size (T) with $(T - m + 1)/m \leq 200$ we use the critical values from Brock et al. (1991). The residuals of the mgap and monov are filtered according to an AR(1) model, since we detected autocorrelation in the residuals.

*Significance on the 1% level.

**Significance on the 5% level.

an identical and independent distribution (iid).[10] As argued in Lee et al. (1993), the BDS test is an appropriate test for detecting general stochastic non-linearities. The advantage of the test is that it is able to detect additive and multiplicative non-linearities, whereas the McLeod-Li test is sensitive against multiplicative non-linearity.

Turning to the results in Tables 2 and 3, the McLeod-Li test does not reject the null hypothesis of linear specifications for most of the models and appropriate lag lengths. Nevertheless, for the mgap model the null hypothesis of linearity in the residuals can be rejected at the 5% significance level for Ljung-Box statistics with a lag length of 24 and 36 respectively. Also, for the output gap specification, we obtain weak support for non-linearities in the models. In order to confirm or reject this presumption, we can refer to the BDS test that seems more powerful than the McLeod-Li test for small samples of 50 observations.[11] The BDS test results indicate non-linear models for the output gap and price gap specification. The former result is consistent with previous literature (e.g. Claus, 2000) and both results are consistent with the McLeod-Li test. Furthermore, the result for the eclectic model provides some support for a non-linear functional form.[12]

4. A COMPUTATIONAL MODELLING APPROACH

As stated in the previous section, the starting point for econometric modelling is typically a theoretical model that is assumed to explain how an economy or

particular market works. The next step is to estimate the model's parameters using an empirical data set. In a highly complex economy, it may become necessary to estimate many different models, based on different explanatory variables. As Berlemann and Nelson (2002) argue, the final problem is to identify the econometric model that facilitates a given forecasting success at least effort.

The genetic-neural fuzzy rule-base approach is simultaneously somewhat similar and somewhat different to the econometric approach. It also relies on a given database in order to set the models parameters, but no theoretical model is needed that explains how inputs and output are linked with each other. GENEFER is able freely to generate and tune a fuzzy rule-base without any prior knowledge as to which inputs to select, how many rules to formulate and how to link the fuzzy sets within one rule. Theoretical models help to pre-select the explanatory variables that are given to the software as potential inputs, but they are not needed to identify the functional form of the relationship between the output and input variables.

In order to work with GENEFER, all time series must be stored in an Excel worksheet (output data in the first column and input candidates in the following ones). Depending on the user's pre-knowledge, the relevant inputs can either be selected manually or be determined automatically, based on the fuzzy curve/fuzzy surface (FC/FS) algorithm (e.g. Lin et al., 1996). The selected inputs are then indexed by superscript from 1 to n.

The next step consists in fuzzifying all variables. After specifying the degree of granularity f (3, 5 or 7)[13] and the type of the fuzzy sets (triangular or gaussian), their widths (wl,wr) and centres (c) can be set manually or determined by setting a common overlap degree or applying a clustering algorithm. After completing this second step, GENEFER provides a fuzzification base FB which is the basis for the following rule-base generating step.

$$\begin{aligned} \mathrm{FB} = \{\{(\mathrm{wl}_1^0, \mathrm{c}_1^0, \mathrm{wr}_1^0), \ldots, (\mathrm{wl}_\mathrm{f}^0, \mathrm{c}_\mathrm{f}^0, \mathrm{wr}_\mathrm{f}^0)\}, \{(\mathrm{wl}_1^1, \mathrm{c}_1^1, \mathrm{wr}_1^1), \ldots, \\ \times (\mathrm{wl}_\mathrm{f}^1, \mathrm{c}_\mathrm{f}^1, \mathrm{wr}_\mathrm{f}^1)\}, \ldots, \{(\mathrm{wl}_1^n, \mathrm{c}_1^n, \mathrm{wr}_1^n), \ldots, (\mathrm{wl}_\mathrm{f}^n, \mathrm{c}_\mathrm{f}^n, \mathrm{wr}_\mathrm{f}^n)\}\}. \end{aligned} \quad (15)$$

The software provides three different methods of setting up a rule-base RB. The user may either set the number of rules and link the fuzzy sets within each rule manually, or apply an evolutionary or neural generating algorithm. Since we work only with the evolutionary algorithm here, we ignore the other two.[14] The evolutionary algorithm selects each training pattern (output and input data for one observation within the training data set) and generates a fuzzy rule by combining the fuzzy sets of each variable that yield the highest degree of membership for the crisp values. The generated rule is called a candidate rule and is copied into

a candidate rule set. After the entire training data set has been processed by this algorithm, all candidate rules are evaluated according to a multi criterion fitness function that accounts for the completeness and consistency of a fuzzy rule-base FRB. The best candidate rule is copied in the generated rule set and all observations within the training database that are covered to a given degree, are removed. Following Gordón and Herrera (1997) this procedure is repeated as long as there are still observations in the training data set that are not covered. The result of this third step is a consistent fuzzy rule-base that completely covers all observations within the training data set. Each rule can be identified by its termcode which is a string of fuzzy set indexes beginning with the output. Example: Given a fuzzification with five fuzzy sets for the output, input2 and input3 variables and a fuzzification of input1 with three fuzzy sets, a termcode of "1252" can be interpreted as follows:

IF Input 1 is *medium* AND Input 2 is *very large*
AND Input 3 is *low* THEN output is *very low*.

The words in italics are fuzzy terms that represent the linguistic labels of one of the fuzzy sets for each variable in the FB. They facilitate reading the system in words and render the fuzzy rule-base transparent and interpretable. Because there might be redundant rules in the generated rule-base due to the parameter settings of the evolutionary algorithm, the rule base can be simplified. The simplification algorithm is based on a binary encoded GA that seeks to reduce the number of rules with the aim of lowering the MSE of the training database, while simultaneously maintaining completeness and consistency (e.g. Gordón & Herrera, 1997).

The forecast performance of a fuzzy rule-base is determined mainly by the cooperation of FB and RB. Since both are generated in subsequent steps, the fuzzy rule-base can be tuned to further minimize MSE. For that purpose, GENEFER transforms the fuzzy system into an equivalent neural-fuzzy-system and applies the (modified) error backpropagation algorithm which guarantees interpretability of the FB. Alternatively, the user may also choose a genetic procedure (e.g. Gordón & Herrera, 1997; Kooths & Ringhut, 2003).

The final result is a set of fuzzy rules of equal length (identical number of inputs in each rule) that were constructed automatically, by acquiring the knowledge within the training database. The user can now present new input observations to the FRB and let GENEFER do the forecasting. Since new observations mean new experience, the forecast errors indicate the need for learning procedures that update the knowledge base (= FRB). Therefore GENEFER provides learning algorithms that allow modifying the RB as well as the FB during the out-of-sample forecast computation (see Section 6). Additionally, there is a COM-interface available that

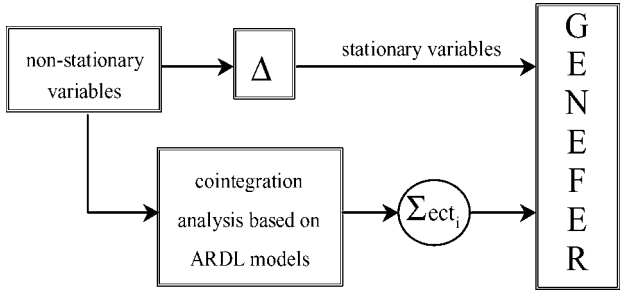


Fig. 1. GENEFER Estimation Set-Up.

links GENEFER to Excel and allows for pre-processing input data or evaluating GENEFER's forecasts. The software itself also provides analysis tools for exact tracking of each calculation and for analysing the forecasts and their evolution during the learning process.

In accordance with the econometric modelling approach, we develop various GENEFER models that are comparable with respect to the major input variables as well as some additional models. The so-called 2-stage (or cooperative) GENEFER models contain the disequilibrium terms derived from cointegration analysis using the ARDL approach (first stage) as inputs for the inflation forecast (second stage). The 2-stage GENEFER models therefore represent a combination of econometric and computational approaches. The other (1-stage) GENEFER models use the (lagged) first differences, indicated by the symbol Δ, of the explanatory variables, in order to ensure an equal order of integration. The models are labelled according to the initial databases before input identification. For instance, the model g_mgap_Δ used the mgap database as in the econometric approach. The explanatory variables may still differ, due to the FC/FS-algorithm. Figure 1 outlines the two input sources of the GENEFER modelling strategy.

None of the models includes dummies, because the fuzzy approach is so far not appropriate for binary variables. Table 4 reports the estimated computational models. We also computed the R^2 and performed standard in-sample diagnostic residual tests for these kinds of models (Table 2 reports a standard F-test and p-values in brackets). As the test results show, the models' in-sample performance is comparable to the econometric specifications. All models pass further conducted stability tests (CUSUM, CUSUMQ).

However, as argued in economic literature, in-sample results say rather little about out of sample performance: Therefore, we turn to the forecast set-up and evaluation as the second test for GENEFER models.

Table 4. GENEFER Models and In-Sample Fit.

Model	Dependent Variable: $\pi(t)$ List of Explanatory Variables	#Fuzzy Sets (π/Input) Type	R^2 (SEE)[a]	Breusch-Godfrey (LM test)[b]	ARCH Test[b]	White Test[b]
g_ygap_trend	$\pi^e(t)$, ygaptrend$(t-1)$, Δpenergy(t), $\Delta e(t-1)$	(5/5) gaussian	0.859 (−0.00642)	0.009 (0.990)	0.936 (0.337)	0.153 (0.858)
g_ygap_hp	$\pi^e(t)$, ygaphp$(t-1)$, Δpenergy(t), $\Delta e(t-1)$	(5/3) gaussian	0.859 (−0.00607)	4.694 (0.013)	0.029 (0.864)	0.356 (0.701)
g_ygap_cd	$\pi^e(t)$, ygapcd$(t-1)$, Δpenergy(t), $\Delta e(t-1)$	(5/3) gaussian	0.854 (−0.00634)	2.198 (0.120)	3.53 (0.065)	0.389 (0.678)
g_monov	$\pi^e(t)$, monov$(t-1)$, Δpenergy$(t-1)$, $\Delta e(t-1)$	(5/3) triangular	0.773 (−0.00799)	2.388 (0.101)	0.072 (0.788)	0.838 (0.437)
g_markup	$\pi^e(t)$, markup$(t-1)$, $\Delta e(t-1)$, Dpenergy(t), Dpenergy$(t-1)$	(7/3) triangular	0.788 (−0.00728)	3.459 (0.038)	0.001 (0.983)	1.504 (0.231)
g_mgap	$\pi^e(t)$, mgap$(t-1)$, $\Delta e(t-1)$, Δpenergy$(t-1)$	(5/3) agaussian	0.854 (−0.00612)	2.951 (0.153)	0.001 (0.976)	1.539 (0.223)
g_eclectic	$\pi^e(t)$, ygaptrend$(t-1)$, monov$(t-1)$, mark-up$(t-1)$, $e(t-1)$, Δenergy(t), $j(t-1)$	(7/3) triangular	0.891 (−0.00587)	3.421 (0.0233)	0.91 (0.408)	0.565 (0.571)
g_markup_D.	$\pi^e(t)$, Δulc$(t-1)$, $\Delta e(t-1)$, Δpenergy(t), Dpenergy$(t-1)$,	(5/5) triangular	0.873 (−0.00556)	1.425 (0.248)	0.285 (0.595)	3.014 (0.057)
g_mgap_D	$\pi^e(t)$, $\Delta m(t-4)$, Dm$(t-5)$, $r(t-1)$, $\Delta y^*(t-1)$, $\Delta e(t-1)$, Δpenergy(t)	(5/3) gaussian	0.88 (−0.00596)	1.097 (0.340)	0.055 (0.815)	0.571 (0.567)
g_monov_D	$\pi^e(t)$, $\Delta m(t-5)$, $\Delta m(t-4)$, $\Delta y(t-1)$, $r(t-1)$, $\Delta e(t-1)$, Δpenergy(t)	(7/3) triangular	0.782 (−0.0084)	2.951 (0.060)	2.915 (0.098)	1.279 (0.286)
g_ecletic_D	$\Delta m(t-1)$, $\Delta m(t-3)$, $\Delta m(t-5)$, $r(t-1)$, Δpenergy(t), $\Delta e(t-1)$, $\pi^e(t)$, $\Delta y^*(t-1)$, Δulc$(t-1)$, Δpim$(t-1)$	(5/3) gaussian	0.896 (−0.00565)	0.858 (0.429)	1.792 (0.176)	4.152 (0.021)

[a] Standard Error of Regression.
[b] Prob-values are given in brackets.

5. FORECAST SET-UP AND EVALUATION CRITERIA

In this section, we compare the results and forecasting power of the econometric and computational methods, using a simulated out-of-sample task over h-quarter ahead forecasts. For this reason, we calculate a set of rolling one-step (one quarter) and four-step ahead (one year) forecasts for each model in a pseudo-dynamic forecast setting (ex-post forecasts).[15] For instance, for the four-step ahead procedure, we first compute four forecasts for 1/97–4/97 using the actual values for the explanatory variables and the period-to-period forecasted values for the inflation rate. The models are then re-estimated for an updated sample until 1/97 to calculate the next four out-of-sample forecasts from 2/97 until 1/98. This sequential updating process is continued until the sample is exhausted. This recursive strategy generates 16 one-step ahead and 13 four-step ahead forecasts for the entire forecast interval until 4/00.

In order to compare the predictive power of the various models, we calculate a set of commonly used parametric accuracy measures: the mean squared errors (MSE), the root mean squared errors (RMSE) and the mean absolute percentage error (MAPE). We further calculate the Theil's U statistic with an AR(1) as a naive benchmark [RMSE(model)/RMSE(AR(1))]. If that test statistic is less than one, the model at least outperforms the benchmark. In analogy with Theil's U, we compute the relative form of the mean absolute error (MAE) as [MAE(model)/MAE(AR(1))] whose interpretation is in line with Theil's U. We also compute the ΔTheil's U, in order to gauge the model's ability to predict turning points.

Taking into account that measures of the squared sum of residuals such as RMSE and MAE may derive misleading conclusions and suboptimal results, we also calculate non-parametric statistics. As a 2×2 contingency table the confusion matrix identifies the ability of each model to correctly predict the direction of change in the level of the variable being forecasted, regardless of how closely the forecast matches the true level. From the confusion matrix, we can derive the confusion rate (CR) which is the number of falsely predicted changes in the inflation rate, divided by the number of observed changes. The closer the CR statistic is to zero, the better the forecasts of the direction of change in inflation, regardless of the level of accuracy.

Moreover we can use the 2×2 contingency table in the form of the confusion matrix to perform a χ^2-test of independence among the forecasts and the actual inflation rate.[16] The null hypothesis of the test states that the forecasted and actual changes in the inflation rate are independent of each other. As Stekler (1991) notes, if a model is assumed to predict changes in the inflation rate correctly, it should at least be able to pass the test and reject the nullhypothesis. We are aware of the

small sample size and also report the Yates corrected form, as well as the exact Fisher test statistic, based on a hypergeometric distribution.[17]

6. RESULTS

A total of 19 models (7 econometric, 11 computational, 1 benchmark) are estimated and their out-of-sample predictive power is compared. All forecast time series are evaluated according to the test statistics mentioned above. In order to provide a quick and illustrative insight into our main results, Fig. 2 contains a graph in the upper left, which shows the three dimensions of our forecasting contest. The other graphs show the results of 15 of 19 models (we omitted the eclectic models, since they cannot be classified into real activity or monetary models) according to different evaluation criteria. Additionally, Table 5 reports the test statistics for the one-step ahead forecasts. The first row displays the AR(1) benchmark model, the following upper half shows the econometric models and the lower part contains the GENEFER models.

The following findings are striking:

- For most of the models, Theil's U, ΔTheil's U and relative MAE are almost always below the critical value of one, indicating a superior predictive power in comparison to the benchmark AR(1).
- The linear econometric specification of the real money gap model (mgap) is most accurate according to the majority of parametric measures presented in the first part of Table 5 (see bold values). Only with respect to MAPE and ΔTheil's U the model is outperformed by the linear eclectic and computational g_mgap model respectively.
- According to the Theil's U statistic, the computational g_mgap and g-monov_Δ models also perform well, indicating that monetary aggregates are good indicators of future inflationary pressure, using both econometric and computational models.
- According to the confusion rate CR, the smallest values are found for the GENEFER models with the g_mgap_Δ model having the highest predictive power with respect to the direction of change in inflation rates.
- Only GENEFER models pass the (Yates corrected) χ^2 test based on the confusion matrix, rejecting the null hypothesis of independence between the predicted and actual directions of change. Again, the g_mgap_Δ model forecasts best. It is significant at a 95% (90%) confidence level for the (Yates corrected) χ^2 test.

A closer look at Fig. 2 reveals that the results for monetary indicators are to some extent superior to real activity variables with respect to the displayed

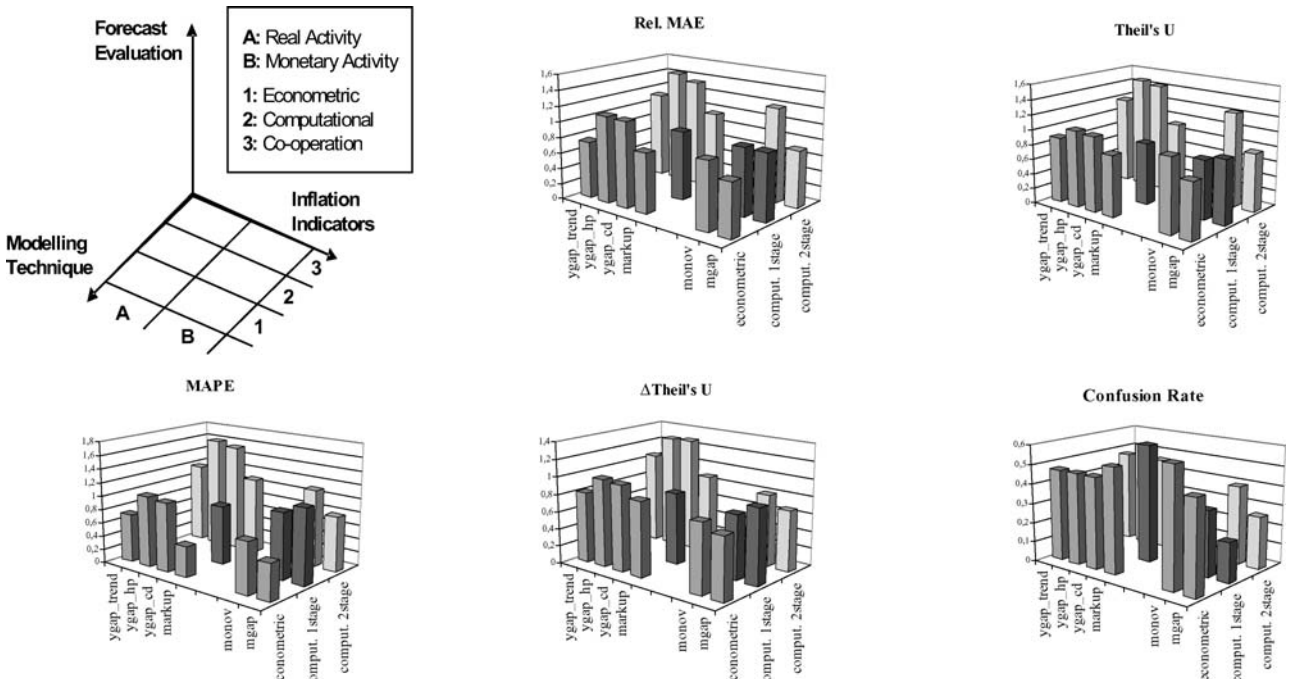


Fig. 2. Multidimensional Evaluation Scheme One-Step Ahead Forecasts.

Table 5. Predictive Power of Various Inflation Forecasting Models for One-Step Ahead Forecasts.

Model	MSE	RMSE	MAPE	Rel. MAE	Theil's U	ΔTheil's U	CR	χ^2 (1)	χ^2 Yates
AR(1)	1.011	1.005	0.963	1.000	1.000	1.078	0.667	0.434	0.632
ygap_trend	0.781	0.884	0.710	0.730	0.879	0.807	0.467	0.045	0.101
ygap_hp	1.064	1.032	1.040	1.105	1.026	1.002	0.467	0.045	0.101
ygap_cd	1.013	1.007	1.009	1.098	1.001	0.981	0.467	0.045	0.101
monov	0.961	0.980	0.745	0.857	0.975	0.784	0.600	0.077	0.101
markup	0.669	0.818	0.450	0.763	0.814	0.855	0.533	0.579	0.058
mgap	**0.543**	**0.737**	0.531	**0.676**	**0.733**	0.700	0.467	0.045	0.632
eclectic	0.891	0.944	**0.386**	0.725	0.939	0.925	0.467	0.077	0.101
g_ygaptrend	1.379	1.174	1.168	1.103	1.168	1.044	0.467	0.077	0.058
g_ygaphp	2.241	1.497	1.619	1.441	1.489	1.284	0.533	0.045	0.101
g_ygapcd	2.124	1.457	1.561	1.363	1.45	1.294	0.467	0.077	0.058
g_monov	1.589	1.261	1.132	1.208	1.254	1.038	0.400	0.714	0.101
g_markup	0.914	0.956	1.117	0.888	0.951	0.903	0.267	3.233***	1.637
g_mgap	0.608	0.780	0.806	0.732	0.776	**0.696**	0.267	3.233***	1.637
g_eclectic	0.747	0.864	0.688	0.805	0.860	0.845	0.667	1.727	0.632
g_eclectic_Δ	0.597	0.773	0.926	0.811	0.768	0.798	0.267	3.233***	1.637
g_markup_Δ	0.690	0.831	0.864	0.934	0.826	0.824	0.600	0.714	0.101
g_mgap_Δ	0.738	0.859	1.097	0.857	0.854	0.860	**0.200**	**5.402****	**3.225*****
g_monov_Δ	0.624	0.790	0.973	0.863	0.786	0.732	0.333	1.607	0.546

**Significance on the 5% significance level.

***Significance on the 10% significance level.

evaluation criteria. This accounts especially for the output gap models that perform poorly on average. The (real activity) markup model yields similar results to the monetary overhang model, but it is persistently outperformed by the real money gap model. The results from different modelling techniques are mixed so that no general trend can be determined: However, econometric models clearly outperform computational ones on the basis of the MAPE, while computational models are superior with respect to the confusion rate and to a minor extent for the Theil's U (disregarding ygap models). For the other evaluation criteria, Fig. 2 shows somewhat heterogeneous results.

Comparing 1- and 2-stage GENEFER models, Fig. 2 indicates slightly better predictions of the 1-stage models, especially with respect to Theil's U and rel. MAE. However, this is more valid for the markup and monov indicator, while the real money gap model using an ec-term, clearly outperforms its rivals.

In summary, the results are promising even for the difficult one-step ahead forecasts, where structural relations between input and output variables are generally weaker than for longer time horizons. So far, neither the linear, nor the GENEFER models can persistently outperform the other estimation technique. The results favour the real money gap models (in linear and non-linear specification), which appear to be among the most reliable indicators to forecast inflation. It is interesting to note that although eclectic models are best equipped to trace back the multiple inflation determinants in-sample (see the goodness-of-fit values), they are generally outperformed by more parsimonious models in terms of actually forecasting inflation out-of-sample.

As the general comparison of GENEFER and linear econometric specifications shows, the linear models generally have smaller values for the MAE and MAPE statistics, while GENEFER models perform slightly better on the basis of the RMSE (except for the GENEFER output gap models). This can be seen as an indication that the linear models forecast quite well on average, but are biased towards (large) outliers, which are given more weight when calculating statistics based on a quadratic loss function, such as the RMSE. In comparison, computational models predict inflation more stable and get turning points right. The latter is obvious with respect to the results of the non-parametric tests. GENEFER models clearly perform better in predicting turning points.

In order to come to a broader judgement, we also compute four-step ahead forecasts. Table 6 reports the same statistics for these predictions, and Fig. 3 highlights the results graphically.

- Compared to the naïve benchmark, the relative performance of the models increases. This indicates, that the longer the forecast period the better structural models perform as opposed to pure time series models. The same result holds,

Table 6. Predictive Power of Various Inflation Forecasting Models for Four-Step Ahead Forecasts.

Model	MSE	RMSE	MAPE	Rel. MAE	Theil's U	ΔTheil's U	CR	χ^2 (1)	χ^2 Yates
AR(1)	1.604	1.267	1.312	1.000	1.000	0.974	0.500	0.000	0.333
ygap_trend	1.074	1.037	0.984	1.005	0.818	0.938	0.417	0.343	0.000
ygap_hp	2.135	1.461	1.446	1.493	1.154	1.311	0.417	0.343	0.000
ygap_cd	2.020	1.421	1.383	1.445	1.122	1.277	0.417	0.343	0.000
monov	0.771	0.878	0.699	0.848	0.693	0.816	0.417	0.343	0.375
markup	1.034	1.017	0.742	0.878	0.803	0.970	0.417	0.343	0.000
mgap	0.569	0.754	0.521	0.692	0.595	0.710	0.500	0.343	0.333
eclectic	1.099	1.048	0.679	0.923	0.828	0.993	0.417	0.343	0.000
g_ygaptrend	1.372	1.171	1.42	1.099	0.925	1.089	0.417	0.343	0.000
g_ygaphp	1.852	1.361	1.075	1.849	1.136	1.174	0.334	1.334	0.333
g_ygapcd	2.233	1.493	1.18	2.000	1.226	1.349	**0.167**	**5.333****	**3.000*****
g_monov	1.269	1.127	0.976	1.015	0.89	0.884	0.250	3.086***	1.371
g_markup	1.166	1.08	1.338	0.988	0.853	0.983	0.250	3.086***	1.371
g_mgap	**0.531**	**0.729**	**0.794**	**0.675**	**0.575**	**0.629**	**0.167**	**5.333****	**3.000*****
g_eclectic	1.114	1.055	0.822	0.968	0.833	0.999	0.667	1.333	0.333
g_eclectic_Δ	0.854	0.924	1.001	0.841	0.73	0.806	0.333	1.5	0.375
g_markup_Δ	0.923	0.961	0.945	0.911	0.759	0.905	0.583	0.343	0.000
g_mgap_Δ	0.921	0.960	1.083	0.854	0.758	0.862	0.250	3.086***	0.371
g_monov_Δ	1.163	1.078	1.312	0.905	0.851	0.941	0.250	3.086***	0.371

**Significance on the 5% significance level.

***Significance on the 10% significance level.

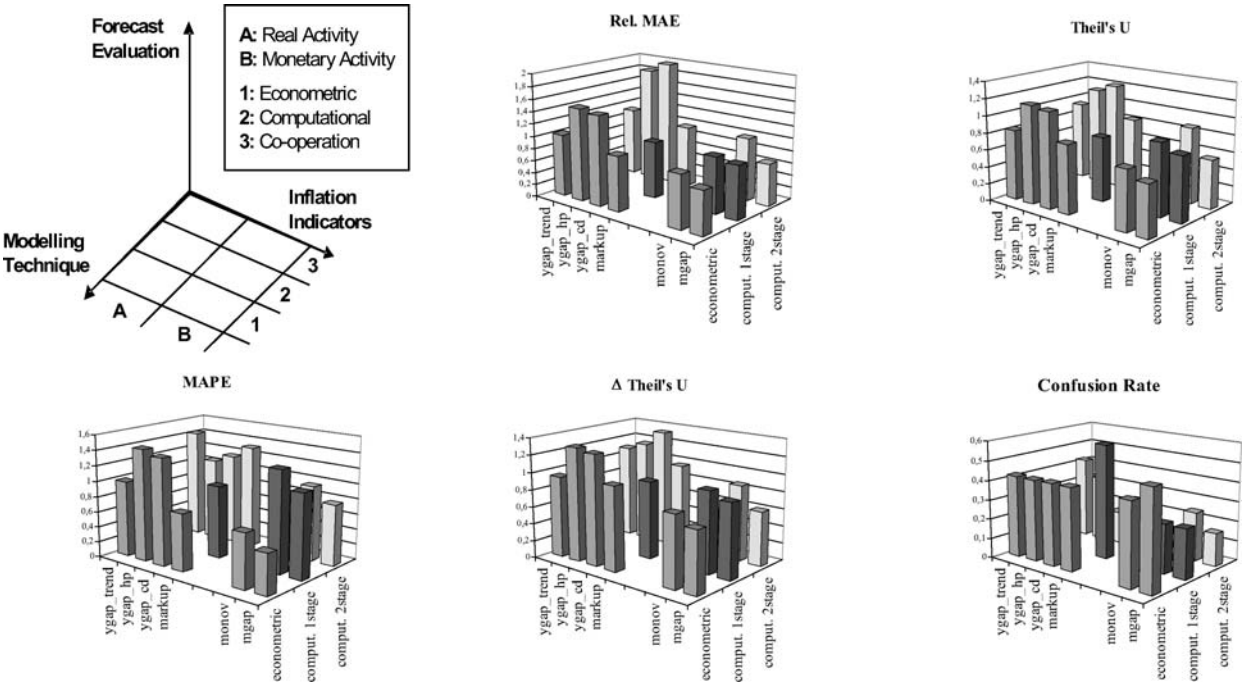


Fig. 3. Multidimensional Evaluation Scheme Four-Step Ahead Forecasts.

if, instead of the simple AR process, a more sophisticated ARIMA model is used (not reported here).[18]

- All models (with the exception of the rather disappointing output gap models) have a Theil's U below the critical value of one and are lower on average, than in the one-step ahead case. GENEFER models in particular, achieve a low value for Theil's U. This indicates that the longer the forecast horizon, the better GENEFER's ability to predict, due to a good structural fit of the underlying system in GENEFER.
- The results of the non-parametric test statistics are even more striking: The confusion rate CR, falls below the values in Table 5 (one-step ahead forecasts) for all models. However, the GENFER models outperform the econometric approaches on average.
- Five GENEFER models pass the χ^2 test based on the confusion matrix (two for the Yates corrected test) while none of the econometric ones does. Significantly, the g_ygapcd model is rather disappointing in its parametric accuracy statistics, but predicts the direction of change in inflation correctly. This result further advocates the hypothesis that GENEFER clearly outperforms linear models in predicting inflationary turning points.
- Regarding the 'winning' model of the forecasting competition, the results of the four-step ahead forecasts are more straightforward than the one step test statistics: The best model is by far the g_mgap. It achieves the best results in all parametric and non-parametric test statistics. The model performs by far the most effectively with respect to Theil's U and passes the Yate's corrected χ^2 test as well as the exact Fisher test at a 10% significance level (not reported in the table).

In comparison to the one-step forecast results, the interpretation of the graphical representation in Fig. 3 is more straightforward: The general trend, already observed when analysing the one-step forecasts, is confirmed, with econometric models having smaller MAPE and rel. MAE values (except for the rather disappointing ygap models), while GENEFER models perform better with test statistics based on RMSE (again disregarding the ygap models). Furthermore, a substantial increase of relative performance of GENEFER models predicting turning points (CR) can be seen from Fig. 3 (lower right). These observations further advocate the hypothesis that computational models are better suited to predict turning points. Finally, regarding the 1- (= using simple growth rates) and 2-stage (= inclusion of error correction terms derived from cointegration analysis) GENEFER models, the performance of the latter is significantly higher and they outclass their 1-stage rivals.

In general, for most evaluation criteria (except the MAPE statistics), both models that give explicit weight to monetary aggregates, perform better than real activity

variables and therefore support monetarist theories of inflation for the EMU. The poor performance of output gap models enhances the findings of the one-step horizon. This also holds for eclectic models, whose relatively poor forecast performance again contrasts with their promising in-sample fit. The results support the above hypothesis that although multiple input models may serve best to identify inflation determinants historically, rather lean models, especially those based on monetary aggregates, are more suitable when extracting leading information about the future path of inflation.

The four-step ahead forecasts are of special importance in determining whether a model captures the general trend of inflation correctly, rather than predicting high frequency movements, which are more dominant for the one-step horizon. The relative dominance of linear and non-linear structural models over univariate time series models argues in favour of the derived models. Nonetheless, the performance of the output gap variants is, on average, rather disappointing. Only the ygap version derived from a simple time trend regression, seems to add useful information beyond that contained in the AR(1) benchmark.

The "winning" model with good results for the one-step ahead forecast horizon and the best performance for four-step forecasts is the GENEFER mgap model, using the derived ec-term from econometric modelling. The combination of both estimation techniques therefore advocates further collaboration between the two forecasting techniques. Because GENEFER delivers the best single forecasting model g_mgap, it is presented in greater detail, because the software's analysis tools for adaptable fuzzy rule-bases are not yet well-known. The g_mgap model consists of 56 generated rules that are reduced to 38 through simplification. It functions through gaussian type fuzzy sets and a degree of granularity of 5 for the inflation rate and 3 for all explanatory variables.

The software can display the rule-base as a list of linked linguistic fuzzy sets – just like the example in Section 4. It represents the rules as a set of activated fuzzy sets in a table grid for a quick overview or shows each single rule in detail. To get an idea of the relationship between one explanatory variable and the inflation rate the software provides a rule-scatter chart that shows a clustering grid, displaying the number of rules that contain the pairwise fuzzy sets in one cluster. The larger the number of rules containing the fuzzy sets in a particular cluster, the larger the gray circles. Figure 4 shows the link between Δpenergy(t$-$1) and the inflation rate within the rule-base of g_mgap (the greater the change in the price index of energy within the last period, the greater the change in the consumer price index).

In order to track GENEFER's forecasts, the software provides several tools that report the details of the underlying algorithms. The "Forecast Manager" is an illustrative tool for analysing and understanding a single forecast value. It contains

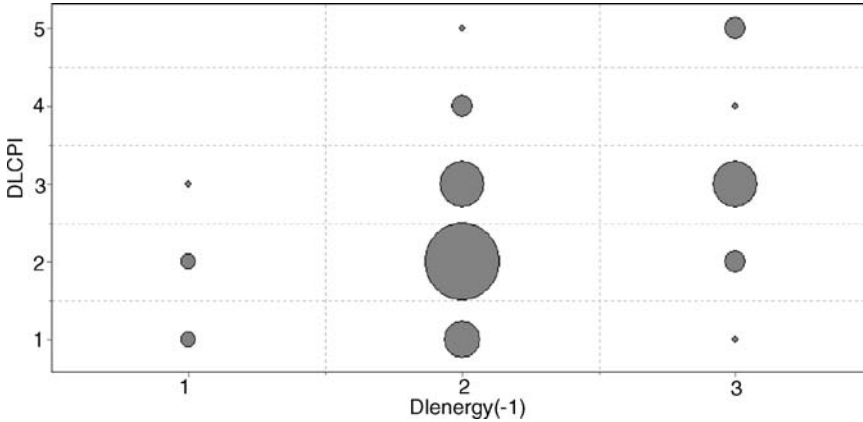


Fig. 4. Rule-Scatter.

and displays all crisp data used in the current forecast period, shows all activated rules not excluded by a user-specific activation filter (since the activation of a fuzzy rule is not a binary value, but a gradual one in the unit interval). It also provides the user with a graph of the complete output fuzzification, the fuzzy inference result (grey area) and an indication of the true value as well as GENEFER's forecast. Figures 5 and 6 shows this graph for g_mgap in period 67.

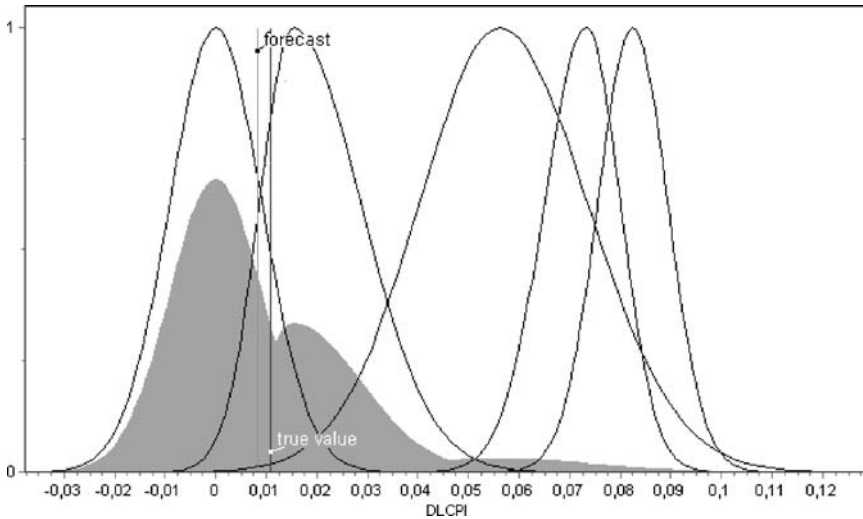


Fig. 5. Fuzzy Inference Result.

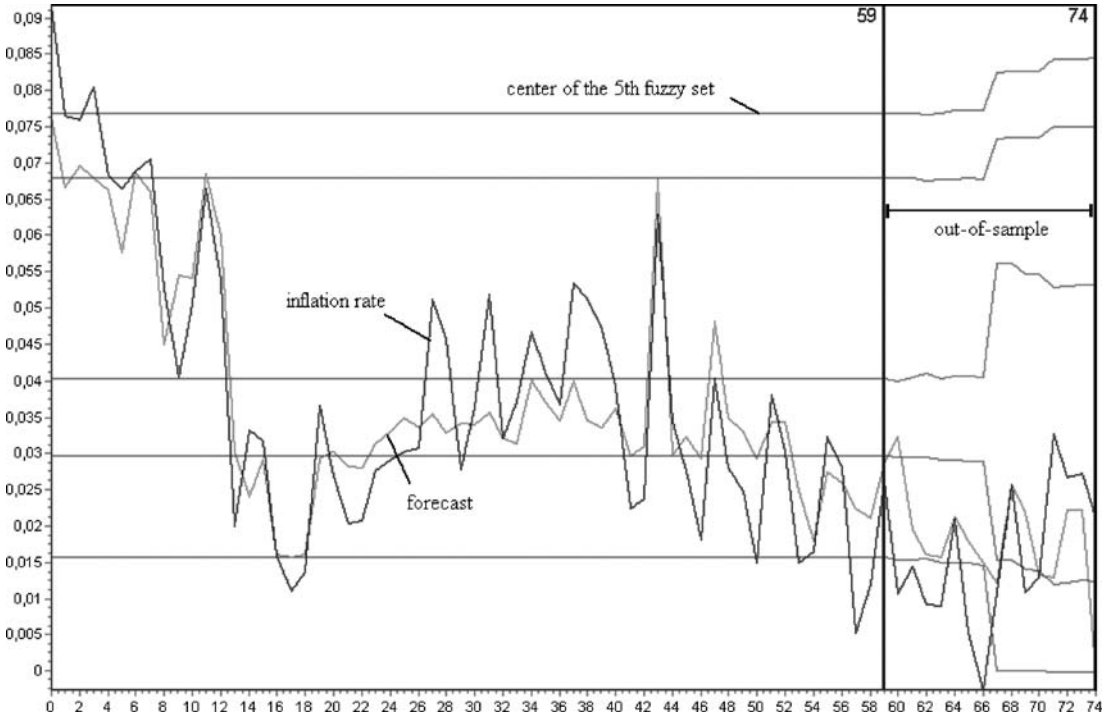


Fig. 6. Time Series of Actual and Forecasted Values in the g_mgap Computational Model.

7. SUMMARY

We have presented a study of the forecasting performance of two types of forecasting techniques: a dynamic, single-equation linear econometric approach, vs. a computational one based on genetic-neural fuzzy rule-bases. We used both approaches to forecast the EMU inflation rate on a one- (one quarter) and four-step ahead basis (one year). We found some superiority for the computational approach according to various different evaluation criteria. Especially striking is the ability of GENEFER models to predict turning points reliably. GENEFER models perform best within a 2-stage approach, where the disequilibrium (error-correction) terms from cointegration analysis are used as input variables. This result proposes a combination of econometric and computational techniques and calls for further research.

The real money gap is found to be the best predictor of future inflation and confirms the current ECB approach of relying on monetary aggregates as a leading indicator for inflation. Nonetheless, some mixed results remain, with respect to the parametric accuracy statistics as opposed to the non-parametric ones. These could possibly be exploited by calculating composites of the forecasts ("thick modelling" e.g. Granger, 2001) that render lower variability and therefore reduce the forecast uncertainty by diversifying over various stable models.

NOTES

1. The details are described in Annex 2 of Fagan et al. (2001) and the database can be downloaded from http://www.ecb.int. We obtained an updated version from Jerome Henry, ECB.

2. The database of Gerlach and Svensson (2003) can be downloaded from http://www.princeton.edu/~svensson/papers/gsdata.xls.

3. The later beginning of our in-sample period is due to the need for a sufficiently large lag length in the estimations.

4. Compared to the static Engle and Granger (1987) approach to cointegration, the ARDL approach has the advantage of allowing for a richer lag structure, while still retaining the specification of one or more explanatory variables. However, weak exogeneity must apply when modelling a single-equation specification, if the model is not to be biased. The ARDL approach to cointegration assumes weak exogeneity to be valid without explicit testing. As argued in Boswijk (1995), it is also possible to test for weak exogeneity ex-post if the derived error correction term of the ARDL model is inserted in equations for the exogenous variables (marginal model in terms of VAR notation) and tested according to their significance with the help of an LM-test. Though some caveats may apply, we treat exogeneity as given in the subsequent modelling process.

5. For a derivation of the markup model see De Brouwer and Ericsson (1998).

6. Next to the trend regression, we also calculate proxies for equilibrium output (i) applying the Hodrick Prescott (HP)-filter with the calibration parameter set to $\lambda = 1600$ as suggested for quarterly data, (ii) estimating a Cobb-Douglas production function as in Fagan et al. (2001). We take all derived versions as possible input variable for the output and money gap model and judge according to statistical criteria which proxy serves best to model and forecast inflation.

7. Though closely related, the monetary overhang and the real money gap may differ from each other, since the money gap would tend to rise for excess output above equilibrium (y^*), while the monetary overhang does not indicate inflationary pressure in excess output situations. The monetary overhang therefore reflects additional information contained in monetary aggregates above its determinants, while the price or real money gap serves more as a summary statistics. See Nicoletti Altimari (2001) for details.

8. Different measures of the equilibrium interest rate turned out to be insignificant and were bypassed in later modelling stages.

9. Table 1 presents the results of standard *F*-tests with Prob-values in brackets. For the mgap, monov, ygap_cd and eclectic model, we obtain a weak indication of heteroskedastic residuals (see White test). If we additionally compute the results of a χ^2-test, the null hypothesis of heteroskedasticity can not be rejected at the 5% significance level only in case of the monetary overhang (monov) and the eclectic model. With respect to all other tests, the models do not show any misspecification with regard to reasonable intervals (again 5% significance value). All models pass stability tests (CUSUM and CUSUMQ) which are not reported in Table 1. These results also provide an ex-post validation of our approach of assuming that there are no breaks in the structural relation, as argued for the data set in Section 2.

10. The Brock-Dechert-Scheinkmann (BDS) test is based on the idea that randomly generated processes are evenly distributed in a n-dimensional space, while processes generated by non-linear models tend to depict geometric structures, if the *n*-dimensional space is reasonably large. Hence, one should observe more clusters when the process is generated by a non-linear model, compared to a randomly created process. Details of the test statistic are reported in Brock et al. (1991). The software to perform the BDS test is available from http://dechert.econ.uh.edu/.

11. Lee et al. (1993) find that the BDS test is quite robust in application and shows low power only in comparison to Neural Networks approaches (e.g., White's neural network test).

12. However, it is difficult to assess whether this is due to non-linearities or heteroskedasticity, as indicated by the White test in Table 1.

13. We restrict the number of fuzzy sets for each variable, in order to maintain interpretability of the fuzzy rule-base.

14. For details see Kooths and Ringhut (2003).

15. One advantage of ex-post forecasts over ex-ante forecasts is that an evaluation of the model's forecasting performance isolates the error component in the forecast of the endogenous variables and prevents a simultaneous error induced by exogenous variables forecasts. Further, most of the forecast accuracy measures applied in this paper are designed to evaluate ex-post forecasts. In order to exploit these advantages, we accept the less realistic ex-post forecasts.

16. For a 2×2 contingency table the χ^2-test has one degree of freedom.

17. The test statistic for the Yates corrected form is

$$\chi^2 = \sum_{i=1}^{4} \frac{(|(x_{oi} - x_{ei})| - 0.5)^2}{x_{ei}}$$

with i ranging over the matrix entries, x_{oi} are observed and x_{ei} expected matrix elements. Since the latter two yield exactly the same results, we report only the Yates corrected χ^2-test in the following.

18. Results are available upon request.

ACKNOWLEDGMENTS

We are grateful to Giulio Nicoletti, Thomas Bittner, Björn Alecke and Gerhard Untiedt for helpful comments and advices. The authors further wish to thank Jerome Henry, who sent us an updated database of the area-wide model for the EMU.

REFERENCES

Banerjee, A., Cockerell, L., & Russell, B. (2001). An I(2) analysis of inflation and the markup. *Journal of Applied Econometrics*, *16*, 221–240.

Berlemann, M., & Nelson, F. (2002). Forecasting inflation via electronic markets – results from a prototype experiment. Working Paper, Dresden University of Technology and University of Iowa.

Boswijk, H. P. (1995). Efficient inference on cointegration parameters in structural error correction models. *Journal of Econometrics*, *69*, 133–158.

Brock, W. A., Hsieh, D. A., & LeBaron, B. (1991). *Nonlinear dynamics, chaos, and instability: Statistical theory and economic evidence*. Cambridge: MIT Press.

De Brouwer, G., & Ericsson, N. R. (1998). Modelling inflation in Australia. *Journal of Business & Economic Statistics*, *16*(4), 433–449.

Claus, I. (2000). Is the output gap a useful indicator of inflation? Reserve Bank of New Zealand Discussion Paper DP2000/05.

Engle, R. F., & Granger, C. W. J. (1987). Co-integration and error correction: Representation, estimation and testing. *Econometrica*, *55*(2), 251–276.

Fagan, G., Henry, J., & Mestre, R. (2001). An area-wide model (AWM) for the Euro area. ECB Working Paper No. 42.

Fic, T. (2003). Identifying determinants of German inflation: An eclectic approach. German Institute for Economic Research (DIW), Discussion Paper No. 334.

Gerlach, S., & Svensson, L. E. O. (2003). Money and inflation in the Euro area: A case for monetary indicators? *Journal of Monetary Economics*, *50*, 1649–1672.

Gordón, O., & Herrera, F. (1997). A three stage evolutionary process for learning descriptive and approximate Fuzzy-logic-controller knowledge bases from examples. *International Journal of Approximate Reasoning*, *17*, 369–407.

Granger, C. (2001). *Thick modelling*. Unpublished manuscript. University of California at San Diego.

Hendry, D. F. (2001). Modelling U.K. inflation, 1875–1991. *Journal of Applied Econometrics*, *16*, 255–275.

Kim, B. Y. (2001). Determinants of inflation in Poland: A structural cointegration approach. Bank of Finland Institute for Economies in Transition, Discussion Paper No. 16.

Kooths, S., & Ringhut, E. (2003). Modelling expectations with GENEFER – an artificial intelligence approach. *Computational Economics*, *21*, 173–194.

Lee, T. H., White, H., & Granger, C. W. J. (1993). Testing for neglected nonlinearity in time series models. *Journal of Econometrics*, *56*, 269–290.

Lin, Y., Cunningham, G. A., & Goggeshall, S. V. (1996). Input variable identification – Fuzzy curves and fuzzy surfaces. *Fuzzy Sets and Systems*, *82*, 65–71.

Nicoletti Altimari, S. (2001). Does money lead Inflation in the Euro Area? ECB Working Paper No. 63.

Pesaran, H. M., & Shin, Y. (1995). An autoregressive distributed lag modelling approach to cointegration analysis. DAE Working Paper No. 9514.

Rech, G. (2002). Forecasting with artificial neural network models. SSE/EFI Working Paper Series in Economics and Finance No. 491.

Roberts, J. M. (1995). New Keynesian economics and the Phillips Curve. *Journal of Money, Credit and Banking*, *27*(4), 975–984.

Rotemberg, J. (1982). Sticky prices in the United States. *Journal of Political Economy*, *60*, 1187–1211.

Stekler, H. O. (1991). Macroeconomic forecast evaluation techniques. *International Journal of Forecasting* (7), 375–384.

Stock, J. J., & Watson, M. W. (1999). Forecasting inflation. NBER Working Paper 7023.

Swansson, N. R., & White, H. (1997). A model selection approach to real time macroeconomic forecasting using linear models and artificial neural networks. *The Review of Economics and Statistics*, *LXXIX*(1), 540–550.

Tödter, K. H. (2002). Monetary indicators and policy rules in the P-star model. Deutsche Bundesbank Discussion Paper No. 18.

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

FINDING OR NOT FINDING RULES IN TIME SERIES

Jessica Lin and Eamonn Keogh

ABSTRACT

Given the recent explosion of interest in streaming data and online algorithms, clustering of time series subsequences has received much attention. In this work we make a surprising claim. Clustering of time series subsequences is completely meaningless. More concretely, clusters extracted from these time series are forced to obey a certain constraint that is pathologically unlikely to be satisfied by any dataset, and because of this, the clusters extracted by any clustering algorithm are essentially random. While this constraint can be intuitively demonstrated with a simple illustration and is simple to prove, it has never appeared in the literature. We can justify calling our claim surprising, since it invalidates the contribution of dozens of previously published papers. We will justify our claim with a theorem, illustrative examples, and a comprehensive set of experiments on reimplementations of previous work.

1. INTRODUCTION

In econometrics, a large fraction of research has been devoted to time series analysis (Enders, 2003). As a recent trend, time series data have also been given a lot of attention in the data mining community (Keogh & Kasetty, 2002; Roddick & Spiliopoulou, 2002). This is highly anticipated since time series data has extended

Applications of Artificial Intelligence in Finance and Economics
Advances in Econometrics, Volume 19, 175–201

ISSN: 0731-9053/doi:10.1016/S0731-9053(04)19007-5

its influence outside of economic applications. It is a by-product in virtually every human endeavor, including biology (Bar-Joseph et al., 2002), finance (Fu et al., 2001; Gavrilov et al., 2000; Mantegna, 1999), geology (Harms et al., 2002b), space exploration (Honda et al., 2002; Yairi et al., 2001), robotics (Oates, 1999) and human motion analysis (Uehara & Shimada, 2002). While traditional time series analysis focuses on modeling and forecasting, data mining researchers focus on discovering patterns (known or unknown) or underlying relationships among the data. These techniques can be very useful in aiding the decision-making process for the econometrics community.

Of all the techniques applied to time series, clustering is perhaps the most frequently used (Halkidi et al., 2001), being useful in its own right as an exploratory technique, and as a subroutine in more complex data mining algorithms (Bar-Joseph et al., 2002; Bradley & Fayyad, 1998). The work in this area can be broadly classified into two categories:

- *Whole Clustering*: The notion of clustering here is similar to that of conventional clustering of discrete objects. Given a set of individual time series data, the objective is to group similar time series into the same cluster.
- *Subsequence Clustering*: Given a single time series, sometimes in the form of streaming time series, individual time series (subsequences) are extracted with a sliding window. Clustering is then performed on the extracted time series subsequences.

Subsequence clustering is commonly used as a subroutine in many other algorithms, including rule discovery (Das et al., 1998; Fu et al., 2001; Uehara & Shimada, 2002; Yairi et al., 2001) indexing (Li et al., 1998; Radhakrishnan et al., 2000), classification (Cotofrei, 2002; Cotofrei & Stoffel, 2002), prediction (Schittenkopf et al., 2000; Tino et al., 2000), and anomaly detection (Yairi et al., 2001). For clarity, we will refer to this type of clustering as STS (Subsequence Time Series) clustering.

In this work we make a surprising claim. Clustering of time series subsequences is meaningless! In particular, clusters extracted from these time series are forced to obey a certain constraints that are pathologically unlikely to be satisfied by any dataset, and because of this, the clusters extracted by any clustering algorithm are essentially random.

Since we use the word "*meaningless*" many times in this paper, we will take the time to define this term. All useful algorithms (with the sole exception of random number generators) produce output that depends on the input. For example, a decision tree learner will yield very different outputs on, say, a credit worthiness domain, a drug classification domain, and a music domain. We call an algorithm "*meaningless*" if the output is independent of the input. As we show in this

paper, the output of STS clustering does not depend on input, and is therefore meaningless.

Our claim is surprising since it calls into question the contributions of dozens of papers. In fact, the existence of so much work based on STS clustering offers an obvious counter argument to our claim. It could be argued: "*Since many papers have been published which use time series subsequence clustering as a subroutine, and these papers produced successful results, time series subsequence clustering must be a meaningful operation*."

We strongly feel that this is not the case. We believe that in all such cases the results are consistent with what one would expect from random cluster centers. We recognize that this is a strong assertion, so we will demonstrate our claim by reimplementing the most successful (i.e. the most referenced) examples of such work, and showing with exhaustive experiments that these contributions inherit the property of meaningless results from the STS clustering subroutine.

The rest of this paper is organized as follows. In Section 2 we will review the necessary background material on time series and clustering, then briefly review the body of research that uses STS clustering. In Section 3 we will show that STS clustering is meaningless with a series of simple intuitive experiments; then in Section 4 we will explain *why* STS clustering cannot produce useful results. In Section 5 we show that the many algorithms that use STS clustering as a subroutine produce results indistinguishable from random clusters. We conclude in Section 6.

2. BACKGROUND MATERIAL

In order to frame our contribution in the proper context we begin with a review of the necessary background material.

2.1. Notation and Definitions

We begin with a definition of our data type of interest, time series:

Definition 1 *(Time Series).* A time series $T = t_1, \ldots, t_m$ is an ordered set of m real-valued variables.

Data mining researchers are typically not interested in any of the global properties of a time series; rather, researchers confine their interest to subsections of the time series, called subsequences.

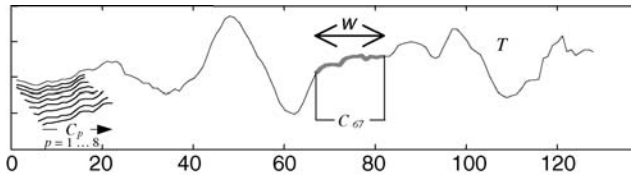


Fig. 1. An Illustration of the Notation Introduced in This Section: A *Time Series T* of Length 128, a *Subsequence* of Length $w = 16$, Beginning at Datapoint 67, and the First 8 Subsequences Extracted by a *Sliding Window*.

Definition 2 *(Subsequence).* Given a time series T of length m, a subsequence C_p of T is a sampling of length $w < m$ of contiguous positions from T, that is, $C = t_p, \ldots, t_{p+w-1}$ for $1 \leq p \leq m - w + 1$.

In this work we are interested in the case where all the subsequences are extracted, and then clustered. This is achieved by use of a sliding window.

Definition 3 *(Sliding Windows).* Given a time series T of length m, and a user-defined subsequence length of w, a matrix S of all possible subsequences can be built by "sliding a window" across T and placing subsequence C_p in the pth row of S. The size of matrix S is $(m - w + 1)$ by w.

Figure 1 summarizes all the above definitions and notations.

Note that while S contains exactly the same information[1] as T, it requires significantly more storage space.

2.2. Background on Clustering

One of the most widely used clustering approaches is hierarchical clustering, due to the great visualization power it offers (Keogh & Kasetty, 2002; Mantegna, 1999). Hierarchical clustering produces a nested hierarchy of similar groups of objects, according to a pairwise distance matrix of the objects. One of the advantages of this method is its generality, since the user does not need to provide any parameters such as the number of clusters. However, its application is limited to only small datasets, due to its quadratic computational complexity. Table 1 outlines the basic hierarchical clustering algorithm.

A faster method to perform clustering is k-means (Bradley & Fayyad, 1998). The basic intuition behind k-means (and a more general class of clustering algorithms known as iterative refinement algorithms) is shown in Table 2.

Table 1. An Outline of Hierarchical Clustering.

Algorithm: Hierarchical Clustering
1. Calculate the distance between all objects. Store the results in a distance matrix.
2. Search through the distance matrix and find the two most similar clusters/objects.
3. Join the two clusters/objects to produce a cluster that now has at least 2 objects.
4. Update the matrix by calculating the distances between this new cluster and all other clusters.
5. Repeat step 2 until all cases are in one cluster.

The *k*-means algorithm for *N* objects has a complexity of O(*kNrD*), where *k* is the number of clusters specified by the user, *r* is the number of iterations until convergence, and *D* is the dimensionality of time series (in the case of STS clustering, *D* is the length of the sliding window, *w*). While the algorithm is perhaps the most commonly used clustering algorithm in the literature, it does have several shortcomings, including the fact that the number of clusters must be specified in advance (Bradley & Fayyad, 1998; Halkidi et al., 2001).

It is well understood that some types of high dimensional clustering may be meaningless. As noted by (Agrawal et al., 1993; Bradley & Fayyad, 1998), in high dimensions the very concept of nearest neighbor has little meaning, because the ratio of the distance to the nearest neighbor over the distance to the average neighbor rapidly approaches one as the dimensionality increases. However, time series, while often having high dimensionality, typically have a low intrinsic dimensionality (Keogh et al., 2001), and can therefore be meaningful candidates for clustering.

2.3. Background on Time Series Data Mining

The last decade has seen an extraordinary interest in mining time series data, with at least one thousand papers on the subject (Keogh & Kasetty, 2002).

Table 2. An Outline of the *k*-Means Algorithm.

Algorithm: *k*-means
1. Decide on a value for *k*.
2. Initialize the *k* cluster centers (randomly, if necessary).
3. Decide the class memberships of the *N* objects by assigning them to the nearest cluster center.
4. Re-estimate the *k* cluster centers, by assuming the memberships found above are correct.
5. If none of the *N* objects changed membership in the last iteration, exit. Otherwise goto 3.

Tasks addressed by the researchers include segmentation, indexing, clustering, classification, anomaly detection, rule discovery, and summarization.

Of the above, a significant fraction use subsequence time series clustering as a subroutine. Below we enumerate some representative examples.

- There has been much work on finding association rules in time series (Das et al., 1998; Fu et al., 2001; Harms et al., 2002a; Uehara & Shimada, 2002; Yairi et al., 2001). Virtually all work is based on the classic paper of Das et al. that uses STS clustering to convert real-valued time series into symbolic values, which can then be manipulated by classic rule finding algorithms (Das et al., 1998).
- The problem of anomaly detection in time series has been generalized to include the detection of surprising or interesting patterns (which are not necessarily anomalies). There are many approaches to this problem, including several based on STS clustering (Yairi et al., 2001).
- Indexing of time series is an important problem that has attracted the attention of dozens of researchers. Several of the proposed techniques make use of STS clustering (Li et al., 1998; Radhakrishnan et al., 2000).
- Several techniques for classifying time series make use of STS clustering to preprocess the data before passing to a standard classification technique such as a decision tree (Cotofrei, 2002; Cotofrei & Stoffel, 2002).
- Clustering of streaming time series has also been proposed as a knowledge discovery tool in its own right. Researchers have suggested various techniques to speed up the STS clustering (Fu et al., 2001).

The above is just a small fraction of the work in the area, more extensive surveys may be found in (Keogh, 2002a; Roddick & Spiliopoulou, 2002).

3. DEMONSTRATIONS OF THE MEANINGLESSNESS OF STS CLUSTERING

In this section we will demonstrate the meaninglessness of STS clustering. In order to demonstrate that this meaninglessness is a result of the way the data is obtained by sliding windows, and not some quirk of the clustering algorithm, we will also do whole clustering as a control (Gavrilov et al., 2000; Oates, 1999). We will begin by using the well-known *k*-means algorithm, since it accounts for the lion's share of all clustering in the time series data mining literature. In addition, the *k*-means algorithm uses Euclidean distance as its underlying metric, and again the Euclidean distance accounts for the vast majority of all published work in this area (Cotofrei, 2002; Cotofrei & Stoffel, 2002; Das et al., 1998; Fu et al., 2001;

Keogh et al., 2001), and as empirically demonstrate in (Keogh & Kasetty, 2002) it performs better than the dozens of other recently suggested time series distance measures.

3.1. K-means Clustering

Because *k*-means is a heuristic, hill-climbing algorithm, the cluster centers found may not be optimal (Halkidi et al., 2001). That is, the algorithm is guaranteed to converge on a local, but not necessarily global optimum. The choices of the initial centers affect the quality of results. One technique to mitigate this problem is to do multiple restarts, and choose the best set of clusters (Bradley & Fayyad, 1998). An obvious question to ask is how much variability in the shapes of cluster centers we get between multiple runs. We can measure this variability with the following equation:

- Let $A = (\bar{a}_1, \bar{a}_2, \ldots, \bar{a}_k)$ be the cluster centers derived from one run of *k*-means.
- Let $B = (\bar{b}_1, \bar{b}_2, \ldots, \bar{b}_k)$ be the cluster centers derived from a different run of *k*-means.
- Let $\text{dist}(\bar{a}_i, \bar{a}_j)$ be the distance between two cluster centers, measured with Euclidean distance.

Then the distance between two sets of clusters can be defined as:

$$\text{cluster\_distance}(A, B) \equiv \sum_{i=1}^{k} \min[\text{dist}(\bar{a}_i, \bar{b}_j)], \quad 1 \leq j \leq k \tag{1}$$

The simple intuition behind the equation is that each individual cluster center in *A* should map on to its closest counterpart in *B*, and the sum of all such distances tells us how similar two sets of clusters are.

An important observation is that we can use this measure not only to compare two sets of clusters derived for the same dataset, but also two sets of clusters which have been derived from *different* data sources. Given this fact, we propose a simple experiment.

We performed 3 random restarts of *k*-means on a stock market dataset, and saved the 3 resulting sets of cluster centers into set $\hat{X}$. We also performed 3 random restarts on random walk dataset, saving the 3 resulting sets of cluster centers into set $\hat{Y}$. Note that the choice of "3" was an arbitrary decision for ease of exposition; larger values do not change the substance of what follows.

We then measured the average cluster distance (as defined in Eq. 1), between each set of cluster centers in $\hat{X}$, to each other set of cluster centers in $\hat{X}$. We call

this number within_set_$\hat{X}$_distance.

$$\text{within\_set\_}\hat{X}\text{\_distance} = \frac{\sum_{i=1}^{3}\sum_{j=1}^{3}\text{cluster\_distance}(\hat{X}_i, \hat{X}_j)}{9} \quad (2)$$

We also measured the average cluster distance between each set of cluster centers in $\hat{X}$, to cluster centers in $\hat{Y}$; we call this number beween_set_$\hat{X}$_and_$\hat{Y}$_distance.

$$\text{beween\_set\_}\hat{X}\text{\_and\_}\hat{Y}\text{\_distance} = \frac{\sum_{i=1}^{3}\sum_{j=1}^{3}\text{cluster\_distance}(\hat{X}_i, \hat{Y}_j)}{9} \quad (3)$$

We can use these two numbers to create a fraction:

$$\text{clustering meaningfulness}\,(\hat{X}, \hat{Y}) \equiv \frac{\text{within\_set\_}\hat{X}\text{\_distance}}{\text{beween\_set\_}\hat{X}\text{\_and\_}\hat{Y}\text{\_distance}} \quad (4)$$

We can justify calling this number "*clustering meaningfulness*" since it clearly measures just that. If, for any dataset, the clustering algorithm finds similar clusters each time regardless of the different initial seeds, the numerator should be close to zero. In contrast, there is no reason why the clusters from two completely different, unrelated datasets should be similar. Therefore, we should expect the denominator to be relatively large. So overall we should expect that the value of *clustering meaningfulness* $(\hat{X}, \hat{Y})$ be close to zero when $\hat{X}$ and $\hat{Y}$ are sets of cluster centers derived from different datasets.

As a control, we performed the exact same experiment, on the same data, but using subsequences that were randomly extracted, rather than extracted by a sliding window. We call this whole clustering.

Since it might be argued that any results obtained were the consequence of a particular combination of k and w, we tried the cross product of $k = \{3, 5, 7, 11\}$ and $w = \{8, 16, 32\}$. For every combination of parameters we repeated the entire process 100 times, and averaged the results. Figure 2 shows the results.

The results are astonishing. The cluster centers found by STS clustering on any particular run of k-means on stock market dataset are not significantly more similar to each other than they are to cluster centers taken from random walk data! In other words, if we were asked to perform clustering on a particular stock market dataset, we could reuse an old clustering obtained from random walk data, and no one could tell the difference!

We re-emphasize here that the difference in the results for STS clustering and whole clustering in this experiment (and all experiments in this work) are due exclusively to the feature extraction step. In particular, both are being tested on the same datasets, with the same parameters of w and k, using the same algorithm.

We also note that the exact definition of *clustering meaningfulness* is not important to our results, since we get the same results regardless of the definition

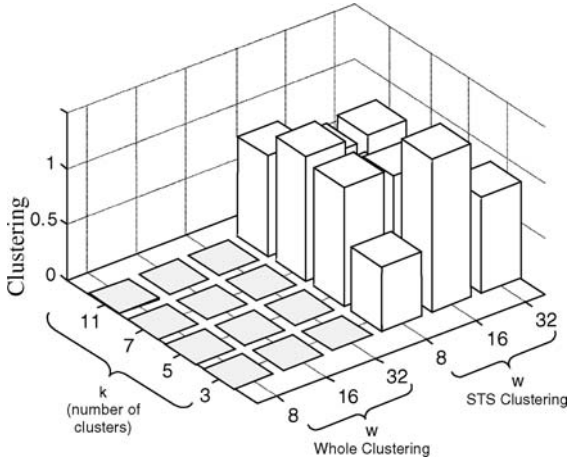


Fig. 2. A Comparison of the Clustering Meaningfulness for Whole Clustering, and STS Clustering, Using *k*-Means With a Variety of Parameters. *Note:* The two datasets used were Standard and Poor's 500 Index closing values and random walk data.

used. In our definition, each cluster center in A maps onto its closest match in B. It is possible, therefore, that two or more cluster centers from A map to one center in B, and some clusters in B have no match. However, we tried other variants of this definition, including pairwise matching, minimum matching and maximum matching, together with dozens of other measurements of clustering quality suggested in the literature (Halkidi et al., 2001); it simply makes no significant difference to the results.

3.2. Hierarchical Clustering

The previous section suggests that *k*-means clustering of STS time series does not produce meaningful results, at least for stock market data. Two obvious questions to ask are, is this true for STS with other clustering algorithms? And is this true for other types of data? We will answer the former question here and the latter question in Section 3.3.

Hierarchical clustering, unlike *k*-means, is a deterministic algorithm. So we can't reuse the experimental methodology from the previous section exactly, however, we can do something very similar.

First we note that hierarchical clustering can be converted into a partitional clustering, by cutting the first k links (Mantegna, 1999). Figure 3 illustrates the idea. The resultant time series in each of the k subtrees can then be merged into

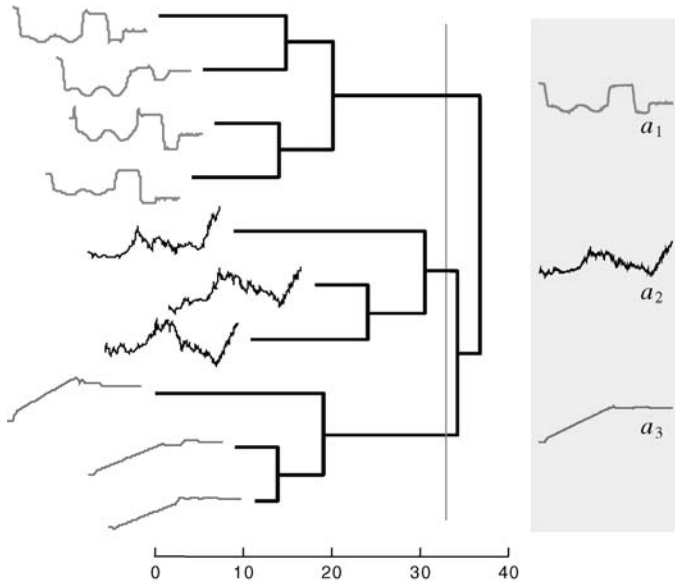


Fig. 3. A Hierarchical Clustering of Ten Time Series. *Note:* The clustering can be converted to a k partitional clustering by "sliding" a cutting line until it intersects k lines of the dendrograms, then averaging the time series in the k subtrees to form k cluster centers (gray panel).

a single cluster prototype. When performing hierarchical clustering, one has to make a choice about how to define the distance between two clusters; this choice is called the linkage method (cf. step 3 of Table 1).

Three popular choices are complete linkage, average linkage and Ward's method (Halkidi et al., 2001). We can use all three methods for the stock market dataset, and place the resulting cluster centers into set X. We can do the same for random walk data and place the resulting cluster centers into set Y. Having done this, we can extend the measure of clustering meaningfulness in Eq. (4) to hierarchical clustering, and run a similar experiment as in the last section, but using hierarchical clustering. The results of this experiment are shown in Fig. 4.

Once again, the results are astonishing. While it is well understood that the choice of linkage method can have minor effects on the clustering found, the results above tell us that when doing STS clustering, the choice of linkage method has as much effect as the choice of dataset! Another way of looking at the results is as follows. If we were asked to perform hierarchical clustering on a particular dataset, but we did not have to report which linkage method we used, we could

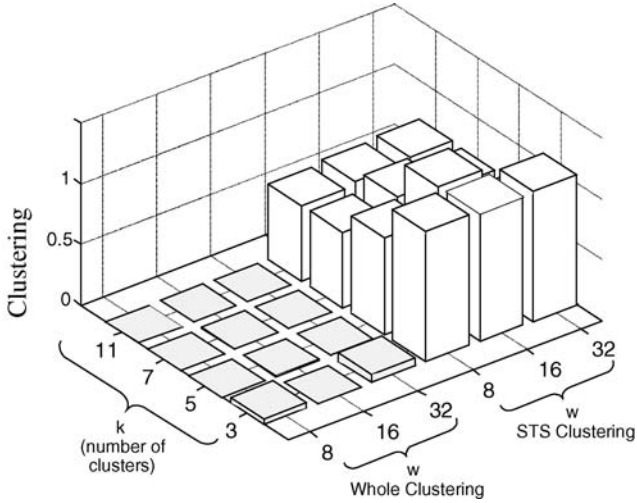


Fig. 4. A Comparison of the Clustering Meaningfulness for Whole Clustering and STS Clustering Using Hierarchical Clustering With a Variety of Parameters. *Note:* The two datasets used were Standard and Poor's 500 Index closing values and random walk data.

reuse an old random walk clustering and no one could tell the difference without re-running the clustering for every possible linkage method.

3.3. Other Datasets and Algorithms

The results in the two previous sections are extraordinary, but are they the consequence of some properties of stock market data, or as we claim, a property of the sliding window feature extraction? The latter is the case, which we can simply demonstrate. We visually inspected the UCR archive of time series datasets for the two time series datasets that appear the least alike (Keogh, 2002b). The best two candidates we discovered are shown in Fig. 5.

We repeated the experiment of Section 3.2, using these two datasets in place of the stock market data and the random walk data. The results are shown in Fig. 6.

In our view, this experiment sounds the death knell for clustering of STS time series. If we cannot easily differentiate between the clusters from these two vastly different time series, then how could we possibly find meaningful clusters in any data?

In fact, the experiments shown in this section are just a small subset of the experiments we performed. We tested other clustering algorithms, including EM

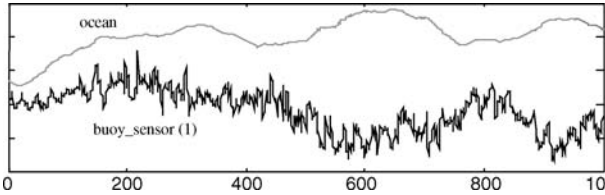


Fig. 5. Two Subjectively Very Dissimilar Time Series from the UCR Archive. *Note:* Only the first 1000 datapoints are shown. The two time series have very different properties of stationarity, noise, periodicity, symmetry, autocorrelation etc.

and SOMs (van Laerhoven, 2001). We tested on 42 different datasets (Keogh, 2002a; Keogh & Kasetty, 2002). We experimented with other measures of clustering quality (Halkidi et al., 2001). We tried other variants of *k*-means, including different seeding algorithms. Although Euclidean distance is the most commonly used distance measure for time series data mining, we also tried other distance measures from the literature, including Manhattan, L_∞, Mahalanobis distance and dynamic time warping distance (Gavrilov et al., 2000; Keogh, 2002a; Oates, 1999). We tried various normalization techniques, including *Z*-normalization, 0–1 normalization, amplitude only normalization, offset only normalization, no normalization etc. In every case we are forced to the inevitable

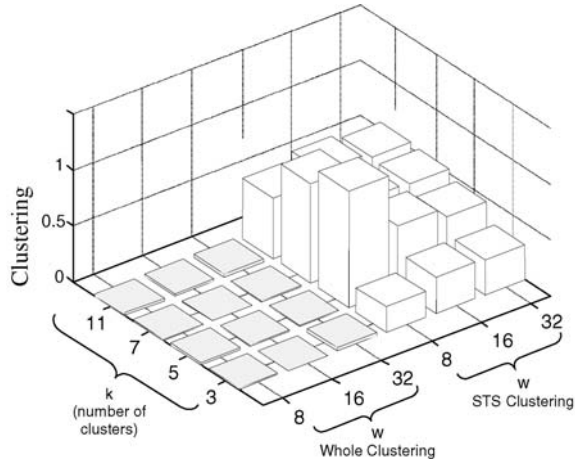


Fig. 6. A Comparison of the Clustering Meaningfulness for Whole Clustering, and STS Clustering, Using *k*-Means With a Variety of Parameters. *Note:* The two datasets used were buoy_sensor(1) and ocean.

conclusion: whole clustering of time series is usually a meaningful thing to do, but sliding window time series clustering is *never* meaningful.

4. WHY IS STS CLUSTERING MEANINGLESS?

Before explaining why STS clustering is meaningless, it will be instructive to visualize the cluster centers produced by both whole clustering and STS clustering. By definition of *k*-means, each cluster center is simply the average of all the objects within that cluster (cf. step 4 of Table 2). For the case of time series, the cluster center is just another time series whose values are the averages of all time series within that cluster. Naturally, since the objective of *k*-means is to group similar objects in the same cluster, we should expect the cluster center to look somewhat similar to the objects in the cluster. We will demonstrate this on the classic Cylinder-Bell-Funnel data (Keogh & Kasetty, 2002). This dataset consists of random instantiations of the eponymous patterns, with Gaussian noise added. Note that this dataset has been freely available for a decade, and has been referenced more than 50 times (Keogh & Kasetty, 2002). While each time series is of length 128, the onset and duration of the shape is subject to random variability. Figure 7 shows one instance from each of the three patterns.

We generated a dataset that contains 30 instances of each pattern, and performed *k*-means clustering on it, with $k = 3$. The resulting cluster centers are shown in

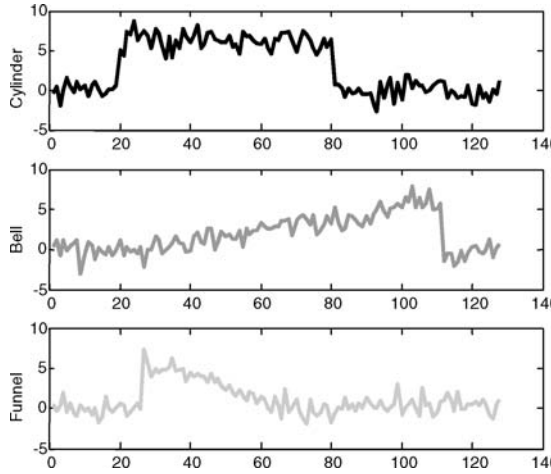


Fig. 7. Examples of Cylinder, Bell, and Funnel Patterns.

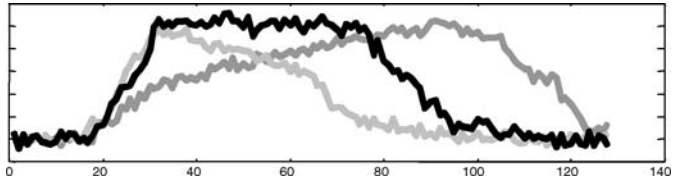


Fig. 8. The Three Final Centers Found by *k*-Means on the Cylinder-Bell-Funnel Dataset. *Note:* The shapes of the centers are close approximation of the original patterns.

Fig. 8. As one might expect, all three clusters are successfully found. The final centers closely resemble the three different patterns in the dataset, although the sharp edges of the patterns have been somewhat "softened" by the averaging of many time series with some variability in the time axis.

To compare the results of whole clustering to STS clustering, we took the 90 time series used above and concatenated them into one long time series. We then performed STS clustering with *k*-means. To make it simple for the algorithm, we used the exact length of the patterns ($w = 128$) as the window length, and $k = 3$ as the number of desired clusters. The cluster centers are shown in Fig. 9.

The results are extraordinarily unintuitive! The cluster centers look nothing like any of the patterns in the data; what's more, they appear to be perfect sine waves.

In fact, for $w \ll m$, we get approximate sine waves with STS clustering regardless of the clustering algorithm, the number of clusters, or the dataset used! Furthermore, although the sine waves are always exactly out of phase with each other by $1/k$ period, overall, their joint phase is arbitrary, and will change with every random restart of *k*-means.

This result explains the results from the last section. If sine waves appear as cluster centers for every dataset, then clearly it will be impossible to distinguish one dataset's clusters from another. Although we have now explained the inability

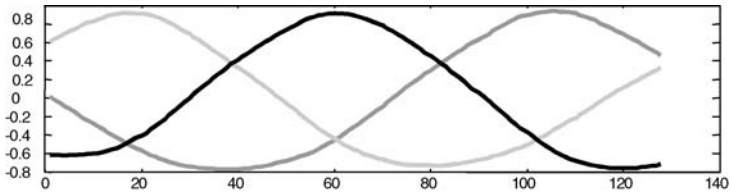


Fig. 9. The Three Final Centers Found by Subsequence Clustering Using the Sliding Window Approach. *Note:* The cluster centers appear to be sine waves, even though the data itself is not particularly spectral in nature. Note that with each random restart of the clustering algorithm, the phase of the resulting "sine waves" changes in an arbitrary and unpredictable way.

of STS clustering to produce meaningful results, we have revealed a new question: why do we always get cluster centers with this special structure?

4.1. A Hidden Constraint

To explain the unintuitive results above, we must introduce a new fact.

Theorem 1. For any time series dataset T with an overall trend of zero, if T is clustered using sliding windows, and $w \ll m$, then the mean of all the data (i.e. the special case of $k = 1$), will be an approximately constant vector.

In other words, if we run STS k-means on *any* dataset, with $k = 1$ (an unusual case, but perfectly legal), we will always end up with a horizontal line as the cluster center. The proof of this fact is straightforward but long, so we have elucidated it in a separate technical report (Truppel et al., 2003). Note that the requirement that the overall trend be zero can be removed, in which case, the $k = 1$ cluster center is still a straight line, but with slope greater than zero. We content ourselves here with giving the intuition behind the proof, and offering a visual "proof" in Fig. 10.

The intuition behind Theorem 1 is as follows. Imagine an arbitrary datapoint t_i somewhere in the time series T, such that $w \leq i \leq m - w + 1$. If the time series is much longer than the window size, then virtually all datapoints are of this type. What contribution does this datapoint make to the overall mean of the STS matrix S? As the sliding window passes by, the datapoint first appears as the rightmost value in the window, then it goes on to appear exactly once in *every* possible location within the sliding window. So the t_i datapoint contribution to the overall shape is the same everywhere and must be a horizontal line. Only those points at the very beginning and the very end of the time series avoid contributing their value to all w columns of S, but these are asymptotically irrelevant. The average of many horizontal lines is clearly just another horizontal line. Another way to look at it is that every value v_i in the mean vector, $1 \leq i \leq w$, is computed by averaging essentially every value in the original time series; more precisely, from t_i to t_{m-w+i}. So for a time series of $m = 1{,}024$ and $w = 32$, the first value in the mean vector is the average of $t[1\ldots993]$; the second value is the average of $t[2\ldots994]$, and so forth. Again, the only datapoints not being included in every computation are the ones at the very beginning and at the very end, and their effects are negligible asymptotically.

The implications of Theorem 1 become clearer when we consider the following well documented fact. For any dataset, the weighted (by cluster membership) average of k clusters must sum up to the global mean. The implication for STS

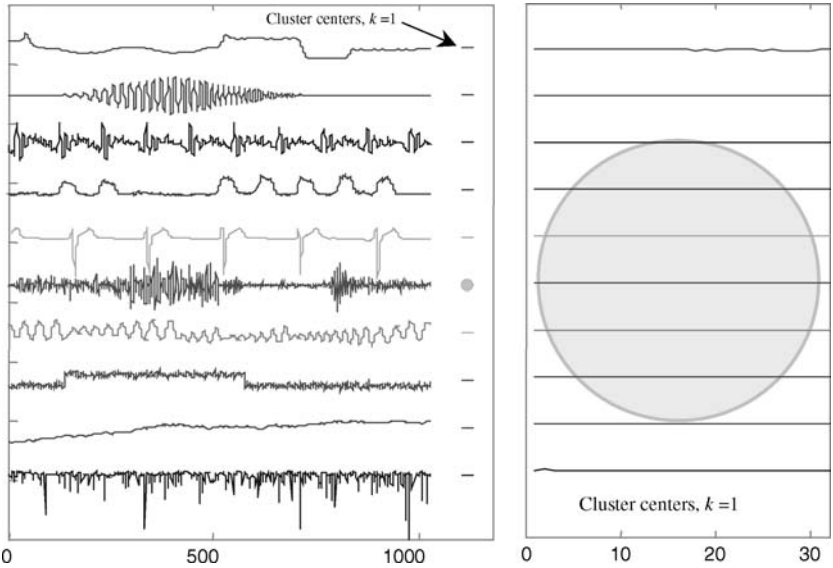


Fig. 10. A Visual "Proof" of Theorem 1. Ten Time Series of Vastly Different Properties of Stationarity, Noise, Periodicity, Symmetry, Autocorrelation etc. *Note:* The cluster centers for each time series, for $w = 32$, $k = 1$ are shown at right. Far right shows a zoom-in that shows just how close to a straight line the cluster centers are. While the objects have been shifted for clarity, they have *not* been rescaled in either axis; note the light gray circle in both graphs. The datasets used are, reading from top to bottom: Space Shuttle, Flutter, Speech, Power_Data, Koski_ecg, Earthquake, Chaotic, Cylinder, Random_Walk, and Balloon.

clustering is profound. Since the global mean for STS clustering is a straight line, then the weighted average of k-clusters must in turn sum to a straight line. However, there is no reason why we should expect this to be true of *any* dataset, much less *every* dataset. This hidden constraint limits the utility of STS clustering to a vanishing small set of subspace of all datasets. The out-of-phase sine waves as cluster centers that we get from the last section conforms to this theorem, since their weighted average, as expected, sums to a straight line.

4.2. The Importance of Trivial Matches

There are further constraints on the types of datasets where STS clustering could possibly work. Consider a subsequence C_p that is a member of a cluster. If we

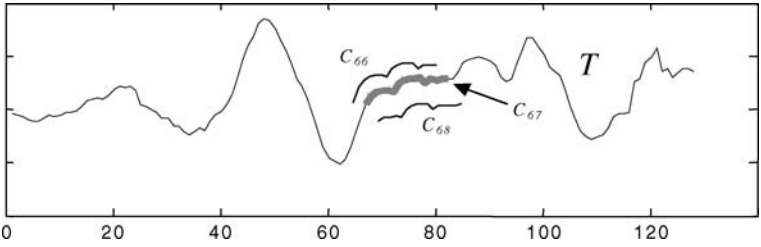


Fig. 11. For Almost Any Subsequence C in a Time Series, the Closest Matching Subsequences are the Subsequences Immediately to the Left and Right of C.

examine the entire dataset for similar subsequences, we should typically expect to find the best matches to C_p to be the subsequences . . ., C_{p-2}, C_{p-1}, C_{p+1}, C_{p+2}, . . . In other words, the best matches to any subsequence tend to be just slightly shifted versions of the subsequence. Figure 11 illustrates the idea, and Definition 4 states it more formally.

Definition 4 *(Trivial Match).* Given a subsequence C beginning at position p, a matching subsequence M beginning at q, and a distance R, we say that M is a *trivial match* to C of order R, if either $p = q$ or there does not exist a subsequence M' beginning at q' such that $D(C, M') > R$, and either $q < q' < p$ or $p < q' < q$.

The importance of trivial matches, in a different context, has been documented elsewhere (Lin et al., 2002).

An important observation is the fact that different subsequences can have vastly different numbers of trivial matches. In particular, smooth, slowly changing subsequences tend to have many trivial matches, whereas subsequences with rapidly changing features and/or noise tend to have very few trivial matches. Figure 12 illustrates the idea. The figure shows a time series that subjectively appears to have a cluster of 3 square waves. The bottom plot shows how many trivial matches each subsequence has. Note that the square waves have very few trivial matches, so all three taken together sit in a sparsely populated region of w-space. In contrast, consider the relatively smooth Gaussian bump centered at 125. The subsequences in the smooth ascent of this feature have more than 25 trivial matches, and thus sit in a dense region of w-space; the same is true for the subsequences in the descent from the peak. So if clustering this dataset with k-means, $k = 2$, then the two cluster centers will be irresistibly drawn to these two "shapes," simple ascending and descending lines.

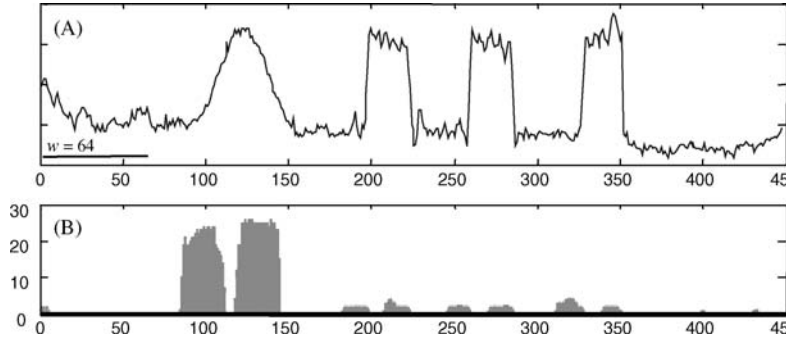


Fig. 12. (A) A Time Series T That Subjectively Appears to Have a Cluster of 3 Noisy Square Waves. (B) Here the ith Value is the Number of Trivial Matches for the Subsequence C_i in T, where $R = 1$, $w = 64$.

The importance of this observation for STS clustering is obvious. Imagine we have a time series where we subjectively see two clusters: equal numbers of a smooth slowing changing pattern, and a noisier pattern with many features. In w-dimensional space, the smooth pattern is surrounded by many trivial matches. This dense volume will appear to any clustering algorithm an extremely promising cluster center. In contrast, the highly featured, noisy pattern has very few trivial matches, and thus sits in a relatively sparse space, all but ignored by the clustering algorithm. Note that it is not possible to simply remove or "factor out" the trivial matches since there is no way to know beforehand the true patterns.

We have not yet fully explained why the cluster centers for STS clustering degenerate to sine waves (cf. Fig. 9). However, we have shown that for STS "clustering," algorithms do not really cluster the data. If not clustering, what are the algorithms doing? It is instructive to note that if we perform singular value decomposition on time series, we also get shapes that seem to approximate sine waves (Keogh et al., 2001). This suggests that STS clustering algorithms are simply returning a set of basis functions that can be added together in a weighted combination to approximate the original data.

An even more tantalizing piece of evidence exists. In the 1920s "data miners" were excited to find that by preprocessing their data with repeated smoothing, they could discover trading cycles. Their joy was shattered by a theorem by Evgeny Slutsky (1880–1948), who demonstrated that any noisy time series will converge to a sine wave after repeated applications of moving window smoothing (Kendall, 1976). While STS clustering is not exactly the same as repeated moving window smoothing, it is clearly highly related. For brevity we will defer future discussion of this point to future work.

4.3. Is there a Simple Fix?

Having gained an understanding of the fact that STS clustering is meaningless, and having developed an intuition as to why this is so, it is natural to ask if there is a simple modification to allow it to produce meaningful results. We asked this question, not just among ourselves, but also to dozens of time series clustering researchers with whom we shared our initial results. While we considered all suggestions, we discuss only the two most promising ones here.

The first idea is to increment the sliding window by more than one unit each time. In fact, this idea was suggested by Das et al. (1998), but only as a speed up mechanism. Unfortunately, this idea does not help. If the new step size s is much smaller than w, we still get the same empirical results. If s is approximately equal to, or larger than w, we are no longer doing subsequence clustering, but whole clustering. This is not useful, since the choice of the offset for the first window would become a critical parameter, and choices that differ by just one timepoint can give arbitrarily different results. As a concrete example, clustering weekly stock market data from "Monday to Sunday" will give completely different cluster patterns and cluster memberships from a "Tuesday to Monday" clustering.

The second idea is to set k to be some number much greater than the true number of clusters we expect to find, then do some post-processing to find the *real* clusters. Empirically, we could not make this idea work, even on the trivial dataset introduced in the last section. We found that even if k is extremely large, unless it is a significant fraction of T, we still get arbitrary sine waves as cluster centers. In addition, we note that the time complexity for k-means increases with k.

It is our belief that there is no simple solution to the problem of STS-clustering; the definition of the problem is itself intrinsically flawed.

4.4. Necessary Conditions for STS Clustering to Work

We conclude this section with a summary of the conditions that must be satisfied for STS clustering to be meaningful.

Assume that a time series contains k approximately or exactly repeated patterns of length w. Further assume that we happen to know k and w in advance. A necessary (but not necessarily sufficient) condition for a clustering algorithm to discover the k patterns is that the weighted mean of the patterns must sum to a horizontal line, and each of the k patterns must have approximately equal numbers of trivial matches.

It is obvious that the chances of both these conditions being met is essentially zero.

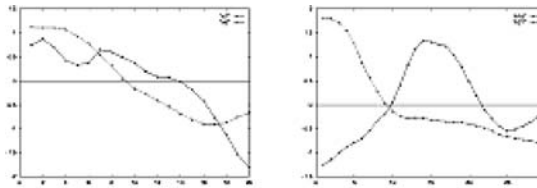
5. A CASE STUDY ON EXISTING WORK

As we noted in the introduction, an obvious counter argument to our claim is the following. "*Since many papers have been published which use time series subsequence clustering as a subroutine, and these papers produce successful results, time series subsequence clustering must be a meaningful operation*." To counter this argument, we have reimplemented the most influential such work, the Time Series Rule Finding algorithm of Das et al. (1998) (the algorithm is not named in the original work, we will call it TSRF here for brevity and clarity).

5.1. (Not) Finding Rules in Time Series

The algorithm begins by performing STS clustering. The centers of these clusters are then used as primitives to convert the real-valued time series into symbols, which are in turn fed into a slightly modified version of a classic association rule algorithm (Agrawal et al., 1993). Finally the rules are ranked by their *J*-measure, an entropy based measure of their significance.

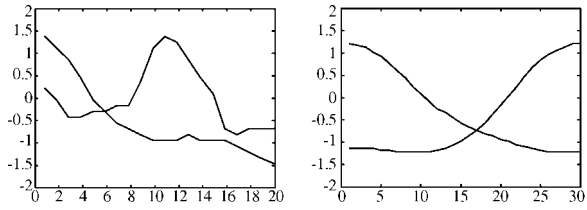
The rule finding algorithm found the rules shown in Fig. 13 using 19 months of NASDAQ data. The high values of support, confidence and *J*-measure are offered as evidence of the significance of the rules. The rules are to be interpreted as follows. In Fig. 13(b) we see that "*if stock rises then falls greatly, follow a smaller*



(a) (b)

w	d	Rule	Sup %	Conf %	J-Mea.	Fig
20	5.5	$7 \Rightarrow^{15} 8$	8.3	73.0	0.0036	(a)
30	5.5	$18 \Rightarrow^{20} 21$	1.3	62.7	0.0039	(b)

Fig. 13. Above, Two Examples of "Significant" Rules Found by Das et al. (This is a Capture of Fig. 4 from Their Paper.) Below, a Table of the Parameters They Used and Results They Found.



w	d	Rule	Sup %	Conf %	J-Mea	Fig
20	5.5	$11 \Rightarrow {}^{15}3$	6.9	71.2	0.0042	(a)
30	5.5	$24 \Rightarrow {}^{20}19$	2.1	74.7	0.0035	(b)

Fig. 14. Above, Two Examples of "Significant" Rules Found in *Random Walk* Data Using the Techniques of Das et al. Below, We Used Identical Parameters and Found Near Identical Results.

rise, then *we can expect to see within 20 time units, a pattern of rapid decrease followed by a leveling out*" (Das et al., 1998).

What would happen if we used the TSRF algorithm to try to find rules in random walk data, using exactly the same parameters? Since no such rules should exist by definition, we should get radically different results.[2] Figure 14 shows one such experiment; the support, confidence and *J*-measure values are essentially the same as in Fig. 13!

This one experiment might have been an extraordinary coincidence; we might have created a random walk time series that happens to have some structure to it. Therefore, for every result shown in the original paper we ran 100 recreations using different random walk datasets, using quantum mechanically generated numbers to insure randomness (Walker, 2001). In every case, the results published cannot be distinguished from our results on random walk data.

The above experiment is troublesome, but perhaps there are simply no rules to be found in stock market. We devised a simple experiment in a dataset that does contain known rules. In particular, we tested the algorithm on a normal healthy electrocardiogram. Here, there is an obvious rule that one heartbeat follows another. Surprisingly, even with much tweaking of the parameters, the TSRF algorithm cannot find this simple rule.

The TSRF algorithm is based on the classic rule mining work of Agrawal et al. (1993); the only difference is the STS step. Since the rule mining work has been carefully vindicated in 100s of experiments on both real and synthetic datasets, it

seems reasonable to conclude that the STS clustering is at the heart of the problems with the TSRF algorithm.

These results may appear surprising, since they invalidate the claims of a highly referenced paper, and many of the dozens of extensions researchers have proposed (Das et al., 1998; Fu et al., 2001; Harms et al., 2002a, b; Hetland & Satrom, 2002; Jin et al., 2002a, b; Mori & Uehara, 2001; Osaki et al., 2000; Sarker et al., 2002; Uehara & Shimada, 2002; Yairi et al., 2001). However, in retrospect, this result should not really be too surprising. Imagine that a researcher claims to have an algorithm that can differentiate between three types of Iris flowers (Setosa, Virginica and Versicolor) based on petal and sepal length and width[3] (Fisher, 1936). This claim is not so extraordinary, given that it is well known that even amateur botanists and gardeners have this skill (British Iris Society, 1997). However, the paper in question is claiming to introduce an algorithm that can find rules in stock market time series. There is simply no evidence that any human can do this, in fact, the opposite is true: every indication suggests that the patterns much beloved by technical analysts such as the "calendar effect" are completely spurious (Jensen, 2000; Timmermann et al., 1998).

6. DISCUSSION AND CONCLUSIONS

As one might expect with such an unintuitive and surprising result, the original version of this paper caused some controversy when first published. Some suggested that the results were due to an implementation bug. Fortunately, many researchers have since independently confirmed our findings; we will note a few below.

Dr. Loris Nanni noted that she had encountered problems clustering economic times series. After reading an early draft of our paper she wrote "*At first we didn't understand what the problem was, but after reading your paper this fact we experimentally confirmed that (STS) clustering is meaningless*!!" (Nanni, 2003). Dr. Richard J. Povinelli and his student Regis DiGiacomo experimentally confirmed that STS clustering produces sine wave clusters, regardless of the dataset used or the setting of any parameters (Povinelli, 2003). Dr. Miho Ohsaki re-examined work she and her group had previously published and confirmed that the results are indeed meaningless in the sense described in this work (Ohsaki et al., 2002). She has subsequently been able to redefine the clustering subroutine in her work to allow more meaningful pattern discovery (Ohsaki et al., 2003). Dr. Frank Höppner noted that he had observed a year earlier than us that ". . . *when using few clusters the resulting prototypes appear very much like dilated and translated trigonometric functions* . . ." (Hoppner, 2002); however, he did not attach

any significance to this. Dr. Eric Perlman wrote to tell us that he had begun to scaling up a project of astronomical time series data mining (Perlman & Java, 2003); however, he abandoned it after noting that the results were consistent with being meaningless the sense described in this work. Dr. Anne Denton noted, "I've experimented myself, (and) the central message of your paper – that subsequence clustering is meaningless – is very right," and "it's amazing how similar the cluster centers for widely distinct series look!" (Denton, 2003).

7. CONCLUSIONS

We have shown that clustering of time series subsequences does not produce meaningful results. We have demonstrated that the choice of clustering algorithms or the measurements of clustering meaningfulness is irrelevant to our claim. We have shown, theoretically and empirically, that the clusters extracted from these time series are forced to obey a certain constraint that is pathologically unlikely to be satisfied by any dataset. More specifically, in order to discover k patterns in any dataset using subsequence clustering, at least two conditions must be satisfied:

(1) the weighted mean of the patterns must sum to a constant line, and
(2) each of the k patterns must have approximately the same number of trivial matches.

Needless to say, the chance of any dataset to exhibit these two properties is very slim.

In future work we intend to consider several related questions; for example, whether or not the weaknesses of STS clustering described here have any implications for model-based, streaming clustering of time series, or streaming clustering of nominal data (Guha et al., 2000). In addition, we plan to investigate alternatives for finding clusters in time series data. One promising direction is towards time series motif discovery algorithms (Chiu et al., 2003; Lin et al., 2002), which identify frequently occurring patterns in time series.

NOTES

1. S contains the same information as T, except that the subsequences are usually normalized individually before inserting to S. Normalization is an important and indispensable step in the sense that it allows identification of similar patterns in time series with different amplitude, scaling, etc.

2. Note that the shapes of the patterns in Figs 13 and 14 are only very approximately sinusoidal. This is because the time series are relatively short compared to the window

length. When the experiments are repeated with longer time series, the shapes converge to pure sine waves.

3. This of course is the famous Iris classification problem introduced by R. A. Fischer. It is probably the most referenced dataset in the world.

ACKNOWLEDGMENTS

We gratefully acknowledge the following people who looked at an early draft of this work. Some of these people were justifiably critical of the work, and their comments lead to extensive rewriting and additional experiments. Their criticisms and comments greatly enhanced the arguments in this paper: Christos Faloutsos, Frank Höppner, Howard Hamilton, Daniel Barbara, Magnus Lie Hetland, Hongyuan Zha, Sergio Focardi, Xiaoming Jin, Shoji Hirano, Shusaku Tsumoto, Loris Nanni, Mark Last, Richard J. Povinelli, Zbigniew Struzik, Jiawei Han, Regis DiGiacomo, Miho Ohsaki, Sean Wang, and the anonymous reviewers of the earlier version of this paper (Keogh et al., 2003). Special thanks to Michalis Vlachos for pointing out the connection between our work and that of Slutsky.

REFERENCES

Agrawal, R., Imielinski, T., & Swami, A. (1993, May 26–28). Mining association rules between sets of items in large databases. In: *Proceedings of the 1993 ACM SIGMOD International Conference on Management of Data* (pp. 207–216). Washington, DC.

Bar-Joseph, Z., Gerber, G., Gifford, D., Jaakkola, T., & Simon, I. (2002, April 18–21). A new approach to analyzing gene expression time series data. In: *Proceedings of the 6th Annual International Conference on Research in Computational Molecular Biology* (pp. 39–48). Washington, DC.

Bradley, P. S., & Fayyad, U. M. (1998, July 24–27). Refining initial points for K-means clustering. In: *Proceedings of the 15th International Conference on Machine Learning* (pp. 91–99). Madison, WI.

British Iris Society, Species Group Staff (1997). *A guide to species irises: Their identification and cultivation*. Cambridge University Press.

Chiu, B., Keogh, E., & Lonardi, S. (2003, August 24–27). Probabilistic discovery of time series motifs. In: *Proceedings of the 9th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (pp. 493–498). Washington, DC, USA.

Cotofrei, P. (2002, August 24–28). Statistical temporal rules. In: *Proceedings of the 15th Conference on Computational Statistics – Short Communications and Posters*. Berlin, Germany.

Cotofrei, P., & Stoffel, K. (2002, April 21–24). Classification rules + time = Temporal rules. In: *Proceedings of the 2002 International Conference on Computational Science* (pp. 572–581). Amsterdam, Netherlands.

Das, G., Lin, K., Mannila, H., Renganathan, G., & Smyth, P. (1998, August 27–31). Rule discovery from time series. In: *Proceedings of the 4th International Conference on Knowledge Discovery and Data Mining* (pp. 16–22). New York, NY.

Denton, A. (2003, December). Personal communication.

Enders, W. (2003). *Applied econometric time series* (2nd ed.). New York: Wiley.

Fisher, R. A. (1936). The use of multiple measures in taxonomic problems. *Annals of Eugenics*, *7*(2), 179–188.

Fu, T. C., Chung, F. L., Ng, V., & Luk, R. (2001, August 26–29). Pattern discovery from stock time series using self-organizing maps. In: *Workshop Notes of the Workshop on Temporal Data Mining, at the 7th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (pp. 27–37). San Francisco, CA.

Gavrilov, M., Anguelov, D., Indyk, P., & Motwani, R. (2000, August 20–23). Mining the stock market: Which measure is best? In: *Proceedings of the 6th ACM International Conference on Knowledge Discovery and Data Mining* (pp. 487–496). Boston, MA.

Guha, S., Mishra, N., Motwani, R., & O'Callaghan, L. (2000, November 12–14). Clustering data streams. In: *Proceedings of the 41st Annual Symposium on Foundations of Computer Science* (pp. 359–366). Redondo Beach, CA.

Halkidi, M., Batistakis, Y., & Vazirgiannis, M. (2001). On clustering validation techniques. *Journal of Intelligent Information Systems (JIIS)*, *17*(2–3), 107–145.

Harms, S. K., Deogun, J., & Tadesse, T. (2002a, June 27–29). Discovering sequential association rules with constraints and time lags in multiple sequences. In: *Proceedings of the 13th International Symposium on Methodologies for Intelligent Systems* (pp. 432–441). Lyon, France.

Harms, S. K., Reichenbach, S., Goddard, S. E., Tadesse, T., & Waltman, W. J. (2002b, May 21–23). Data mining in a geospatial decision support system for drought risk management. In: *Proceedings of the 1st National Conference on Digital Government* (pp. 9–16). Los Angeles, CA.

Hetland, M. L., & Satrom, P. (2002). Temporal rules discovery using genetic programming and specialized hardware. In: *Proceedings of the 4th International Conference on Recent Advances in Soft Computing* (December 12–13). Nottingham, UK.

Honda, R., Wang, S., Kikuchi, T., & Konishi, O. (2002). Mining of moving objects from time-series images and its application to satellite weather imagery. *The Journal of Intelligent Information Systems*, *19*(1), 79–93.

Hoppner, F. (2002, Sept/Okt). Time series abstraction methods – A survey. In: Tagungsband zur 32. GI Jahrestagung 2002, Workshop on Knowledge Discovery in Databases (pp. 777–786). Dortmund.

Jensen, D. (2000). Data snooping, dredging and fishing: The dark side of data mining. 1999 SIGKDD Panel Report. *ACM SIGKDD Explorations*, *1*(2), 52–54.

Jin, X., Lu, Y., & Shi, C. (2002a, May 6–8). Distribution discovery: Local analysis of temporal rules. In: *Proceedings of the 6th Pacific-Asia Conference on Knowledge Discovery and Data Mining* (pp. 469–480). Taipei, Taiwan.

Jin, X., Wang, L., Lu, Y., & Shi, C. (2002b, August 12–14). Indexing and mining of the local patterns in sequence database. In: *Proceedings of the 3rd International Conference on Intelligent Data Engineering and Automated Learning* (pp. 68–73). Manchester, UK.

Kendall, M. (1976). *Time-series* (2nd ed.). London: Charles Griffin and Company.

Keogh, E. (2002a, August 20–23). Exact indexing of dynamic time warping. In: *Proceedings of the 28th International Conference on Very Large Data Bases* (pp. 406–417). Hong Kong.

Keogh, E. (2002b). The UCR time series data mining archive. http://www.cs.ucr.edu/∼eamonn/TSDMA/index.html. Computer Science & Engineering Department, University of California, Riverside, CA.

Keogh, E., Chakrabarti, K., Pazzani, M., & Mehrotra, S. (2001). Dimensionality reduction for fast similarity search in large time series databases. *Journal of Knowledge and Information Systems*, *3*(3), 263–286.

Keogh, E., & Kasetty, S. (2002, July 23–26). On the need for time series data mining benchmarks: A Survey and empirical demonstration. In: *Proceedings of the 8th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (pp. 102–111). Edmonton, Alta., Canada.

Keogh, E., Lin, J., & Truppel, W. (2003, November 19–22). Clustering of time series subsequences is meaningless: Implications for past and future research. In: *Proceedings of the 3rd IEEE International Conference on Data Mining* (pp. 115–122). Melbourne, FL.

Li, C., Yu, P. S., & Castelli, V. (1998, November 3–7). MALM: A framework for mining sequence database at multiple abstraction levels. In: *Proceedings of the 7th ACM International Conference on Information and Knowledge Management* (pp. 267–272). Bethesda, MD.

Lin, J., Keogh, E., Patel, P., & Lonardi, S. (2002, July 23–26). Finding motifs in time series. In: *Workshop Notes of the 2nd Workshop on Temporal Data Mining, at the 8th ACM International Conference on Knowledge Discovery and Data Mining*. Edmonton, Alta., Canada.

Mantegna, R. N. (1999). Hierarchical structure in financial markets. *European Physical Journal*, *B11*, 193–197.

Mori, T., & Uehara, K. (2001, September 18–21). Extraction of primitive motion and discovery of association rules from human motion. In: *Proceedings of the 10th IEEE International Workshop on Robot and Human Communication* (pp. 200–206). Bordeaux-Paris, France.

Nanni, L. (2003, April 22). Personal communication.

Oates, T. (1999, August 15–18). Identifying distinctive subsequences in multivariate time series by clustering. In: *Proceedings of the 5th International Conference on Knowledge Discovery and Data Mining* (pp. 322–326). San Diego, CA.

Ohsaki, M., Sato, Y., Yokoi, H., & Yamaguchi, T. (2002, December 9–12). A rule discovery support system for sequential medical data, in the case study of a chronic hepatitis dataset. In: *Workshop Notes of the International Workshop on Active Mining, at IEEE International Conference on Data Mining*. Maebashi, Japan.

Ohsaki, M., Sato, Y., Yokoi, H., & Yamaguchi, T. (2003, September 22–26). A rule discovery support system for sequential medical data, in the case study of a chronic hepatitis dataset. In: *Workshop Notes of Discovery Challenge Workshop, at the 14th European Conference on Machine Learning/the 7th European Conference on Principles and Practice of Knowledge Discovery in Databases*. Cavtat-Dubrovnik, Croatia.

Osaki, R., Shimada, M., & Uehara, K. (2000). A motion recognition method by using primitive motions. In: H. Arisawa & T. Catarci (Eds), *Advances in Visual Information Management: Visual Database Systems* (pp. 117–127). Kluwer.

Perlman, E., & Java, A. (2003). Predictive mining of time series data. In: H. E. Payne, R. I. Jedrzejewski & R. N. Hook (Eds), *ASP Conference Series, Vol. 295, Astronomical Data Analysis Software and Systems XII* (pp. 431–434). San Francisco.

Povinelli, R. (2003, September 19). Personal communication.

Radhakrishnan, N., Wilson, J. D., & Loizou, P. C. (2000). An alternative partitioning technique to quantify the regularity of complex time series. *International Journal of Bifurcation and Chaos*, *10*(7), 1773–1779. World Scientific Publishing.

Roddick, J. F., & Spiliopoulou, M. (2002). A survey of temporal knowledge discovery paradigms and methods. *Transactions on Data Engineering*, *14*(4), 750–767.

Sarker, B. K., Mori, T., & Uehara, K. (2002). Parallel algorithms for mining association rules in time series data. CS24–2002–1, Tech. Report.

Schittenkopf, C., Tino, P., & Dorffner, G. (2000, July). The benefit of information reduction for trading strategies. Report Series for Adaptive Information Systems and Management in Economics and Management Society. Report# 45.

Timmermann, A., Sullivan, R., & White, H. (1998). The dangers of data-driven inference: The case of calendar effects in stock returns. FMG Discussion Papers dp0304, Financial Markets Group and ESRC.

Tino, P., Schittenkopf, C., & Dorffner, G. (2000, July). Temporal pattern recognition in noisy non-stationary time series based on quantization into symbolic streams: Lessons learned from financial volatility trading. Report Series for Adaptive Information Systems and Management in Economics and Management Sciences. Report# 46.

Truppel, W., Keogh, E., & Lin, J. (2003). A hidden constraint when clustering streaming time series. UCR Tech. Report.

Uehara, K., & Shimada, M. (2002). Extraction of primitive motion and discovery of association rules from human motion data. In: *Progress in Discovery Science, Lecture Notes in Artificial Intelligence* (Vol. 2281, pp. 338–348). Springer-Verlag.

van Laerhoven, K. (2001). Combining the Kohonen self-organizing map and K-means for on-line classification of sensor data. In: G. Dorffner, H. Bischof & K. Hornik (Eds), *Artificial Neural Networks, Lecture Notes in Artificial Intelligence* (Vol. 2130, pp. 464–470). Springer-Verlag.

Walker, J. (2001). HotBits: Genuine random numbers generated by radioactive decay. http://www.fourmilab.ch/hotbits.

Yairi, Y., Kato, Y., & Hori, K. (2001, June 18–21). Fault detection by mining association rules in house-keeping data. In: *Proceedings of the 6th International Symposium on Artificial Intelligence, Robotics and Automation in Space*. Montreal, Canada.

A COMPARISON OF VAR AND NEURAL NETWORKS WITH GENETIC ALGORITHM IN FORECASTING PRICE OF OIL

Sam Mirmirani and Hsi Cheng Li

ABSTRACT

This study applies VAR and ANN techniques to make ex-post forecast of U.S. oil price movements. The VAR-based forecast uses three endogenous variables: lagged oil price, lagged oil supply and lagged energy consumption. However, the VAR model suggests that the impacts of oil supply and energy consumption has limited impacts on oil price movement. The forecast of the genetic algorithm-based ANN model is made by using oil supply, energy consumption, and money supply (M1). Root mean squared error and mean absolute error have been used as the evaluation criteria. Our analysis suggests that the BPN-GA model noticeably outperforms the VAR model.

1. INTRODUCTION

Oil, one of the strategic commodities, has played a critical role in affecting the world's economic conditions, before and after the appearance of new economic currents. In his commentary, Evans (2000) contends that the single most influential

Applications of Artificial Intelligence in Finance and Economics
Advances in Econometrics, Volume 19, 203–223

ISSN: 0731-9053/doi:10.1016/S0731-9053(04)19008-7

factor in causing recessions has been oil shocks. For that reason, reliable prediction of the price of oil would be important to researchers and policy makers. Emerson (2000) argues that while OPEC, a crisis manager, tends to maintain price stability; financial market speculators and hedgers have a propensity to exacerbate oil price fluctuations.

Various econometric techniques have been used for predicting the price movement of oil. Economists are unable to agree which is the most suitable method for making reliable oil price predictions. One of the widely used econometric techniques is vector autoregression (VAR).

More recently, artificial neural networks (ANN) techniques have gained much deserved popularity in engineering because of their flexibility and accuracy. Many algorithms have been used in ANN models. Among them is the genetic algorithm which is considered to be superior for searching global optimality.

The purpose of this study is to incorporate oil supply, oil demand, and government policy into forecasting models of oil prices. We compare the results of a vector autoregression model with that of a genetic algorithm-based ANN model.

This paper begins with a review of the most recent literature on oil price movements and some forecasting techniques. This is followed by a description of data bases. An application of VAR is given in the next section. The use of ANN models in forecasting is then elaborated. Findings and discussions are given in the two sections.

2. LITERATURE REVIEW

The global oil glut of the 1990s has been viewed as the key to the low rate of inflation and the high rate of growth in the United States. The reversal of the oil price movement in early 1999, due to OPEC's decision to reduce the level of exports on the one hand and the limited refinery capacity worldwide on the other, has led some policy makers and economists to wonder whether changes in energy price have long lasting macroeconomic impacts.

Gisser and Goodwin (1986), Hamilton (1996), and Bernanke et al. (1997) have identified a negative correlation between oil price and economic growth. On the contrary, Raymond and Rich (1997) suggest that oil price fluctuations may not present a significant threat to economic growth in the long run.

Concerning inflation, the price of oil has been known to be one of the main supply-side factors that have contributed to general price increases in most industrialized countries. Oil price as an inflationary factor in the United States was examined by Pain et al. (2000), and in European economies by Sarantis and

Stewart (2000). Both studies concluded that oil price has a strong influence on the inflationary expectation.

On the other hand, changes in general economic conditions may also affect the price of oil. In a comprehensive study of global energy demand for the next two decades, Matsui (1998) projects a moderate increase in demand with little change in international oil prices. Taking the political economy tack, Jaffe and Manning (2000) are more optimistic and suggest a prolonged surplus and lower prices for the next 20 years.

Many in the oil business do not share this optimistic view. Linden (1998) forecasts oil shortages and consequently rising prices. In a more recent study, Johnson (2002) paints a gloomy picture of the future, making references to global oil shortages. On the other hand, with a qualitative approach, Lawrence (2002) incorporates supply and demand forces to analyze oil and natural gas price in the future. The focus of the study is on the potential political tensions and socio-economic instability among OPEC members rooted in the September 11th attack on the U.S. and presents a more optimistic future for oil prices. In another study, Weinert (2002) points out that the attack will have little effect on oil prices, due to lower demand for oil.

In the academic community, the estimation of oil price and its relationship with other variables has brought up methodological issues. Hooker (1996) points out that the existence of breakpoints and the changes in Granger-causality in sub-samples necessitate the use of non-linear analysis. In analyzing long-run energy price fluctuations, Pindyck (1999) suggests the use of a non-structural model. On the other hand, Morana (2001) points out that the GARCH model allows for better forecasting without requiring the specification of the structural model. As expected, changes in modeling technique provide the public with mixed results in forecasting oil price movements.

In addition to econometric methods, engineering-based techniques, such as neural networks and genetic algorithms, have gradually found their way into economic and business modeling (Curry & Peel, 1998; Dawid, 1996; Kyungdoo et al., 1997; Smith & Mason, 1997; Zhang & Hu, 1998). A thorough review of literature pertaining to neural network applications in finance and business is provided by Wong and Yakup (1997, 1998) and Chen (2002). Although we have not been able to find any application of neural networks in analyzing oil price movement, McMenamin and Monforte (1998) have applied neural networks in a related field, the forecasting of short-term energy loads. They find that because of nonlinearities, neural networks are especially well suited for the analysis, and provide improvements in forecast accuracy when compared with regression models.

The methodological linkage between time series analysis and neural networks is given by Cheng and Titterington (1994). This linkage suggests that a neural

network estimator can be viewed as a sieve extremum estimator. The consistency of sieve extremum estimators has been recognized by Geman and Hwang (1982) and White and Wooldridge (1991). The convergence and asymptotic normality of these estimators have been established by Barron (1993, 1994), Chen and Shen (1998), Shen (1997) and Shen and Wong (1994).

In comparison with standard econometric techniques, neural networks make less restrictive assumptions on the underlying distributions, allow non-parametric functional forms, and provide a higher degree of robustness (Bode, 1998; Lippmann, 1992; Ripley, 1993). As pointed out by Kuo and Reitsch (1995/1996), neural networks provide meaningful predictions when independent variables were correlated or missing. It is also known that neural networks tend to outperform the conventional regression analysis in the presence of ambiguity in independent variables (Denton, 1995). Cooper (1999) compares the multi-layer feed-forward neural networks results with that of multivariate statistics in the international economics field and confirms the superiority of the technique. Similar conclusions are drawn by Kudyba (1998) in forecasting electricity demand. A comparison between VAR and ANN models for economic forecasting has been made by Moshiri et al. (1999).

Our motivation to use neural networks in forecasting the price of oil is justified on the basis that the standard random walk assumption is often misapplied to cover a noisy nonlinear process (Grudnitski & Osburn, 1993). When a nonlinear model is adopted, the functional form of a traditional model is often arbitrarily imposed on rather than dictated by the data. The use of neural networks provides needed flexibility and prevents the adoption of other stringent assumptions (Denton, 1995; Kuo & Reitsch, 1995/1996). Thus, neural networks are superior to traditional approaches in terms of parsimony of parameterization (Cheng & Titterington, 1994). Finally, a network structure is trained by using part of the data and then tested by using the rest of the data. A well-trained network is expected to provide robust predictions. Therefore, it is no surprise to learn that neural network models outperform non-linear regression in terms of predictability (Cheng & Titterington, 1994; Grudnitski & Osburn, 1993).

3. DATA

Following price theory, market price of oil should primarily be determined by oil demanded and oil supplied, subject to the intervention of government policy. It is therefore reasonable to estimate the price movement of oil by using petroleum consumption, oil supply, and a policy variable. The use of petroleum consumption and oil supply, respectively, to reflect oil demanded and supplied seems to be

straightforward. The identification of policy variables which have strong and steady impacts on oil price, however, poses quite a challenge. In addition to the theoretical adequacy of the policy variable, the availability of data is also a concern. Because oil price movement interacts with both inflation and economic growth, which in turn are subject to the constraint of monetary policy, we have arbitrarily selected money supply as a proxy of government policy.

The monthly data covers the period of 1980:01 to 2002:11, a total of 275 sample observations of oil price (P), petroleum consumption (CO), oil supply (S), and $M1$ money supply ($M1$). P is the daily NYMEX light sweet crude oil futures prices (U.S. dollars per barrel). Measured in trillion BTU, petroleum consumption includes the oil consumed monthly by both businesses and consumers in the U.S. Monthly oil supply, measured in million barrels per day, is derived by adding the domestic production to the oil imported from overseas. $M1$ is measured in billions of dollars. Data on nominal oil prices, oil supply, and petroleum consumption, all seasonally unadjusted, have been obtained from the Energy Information Administration, U.S. Department of Energy. Seasonally adjusted money supply, measured in $M1$, has been made available by the Federal Reserve System.

The price of oil experienced a steady increase since the late 1970s. Despite several significant decreases, oil price maintained relative high levels for the first half of the 1980s. The sharp plunge in 1986 marked the end of a price movement pattern. The formation of this unique plateau pattern was attributable to international politics. Since the middle of the 1980s, oil prices fluctuated around $15 per barrel. Towards the end of the century, however, price volatility seemed to have increased. It is obvious that a linear model is inadequate for predicting oil price movement.

Figure 2 shows the petroleum consumption by households and businesses from 1980 to 2002. The time series exhibits a gradual decrease in the early 1980s. Despite its volatility, the demand for petroleum has gradually increased since 1983.

As shown in Fig. 3, the supply of oil, which is the sum of domestic production and imports, follows the general pattern as shown by petroleum consumption. However, a close examination reveals that oil supply has higher seasonal volatility than petroleum consumption.

As shown in Fig. 4, the supply of $M1$ in the United States is obviously not stationary. Over a span of twenty years, money supply, as measured by $M1$, has grown three folds. The rate of increase of money supply far exceeds that of petroleum consumption or oil supply. The issue of stationarity requires the immediate attention of the modeler. The $M1$ series reveals another significant difference from other series in this study. That is, money supply exhibits much less volatility over time.

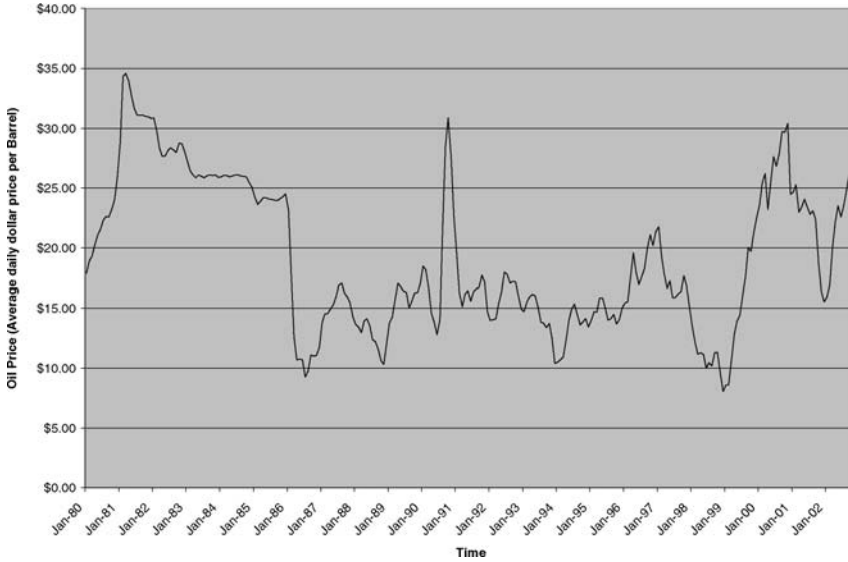


Fig. 1. Oil Price (P). Original Data, January 1980 to November 2002. *Source:* U.S. Dept. of Energy, Energy Information Administration, Monthly Energy Review.

Because both oil price and money supply are measured in nominal values, there is no need to convert any variable from real values to nominal values. To obtain unbiased estimates, oil price, petroleum consumption, and oil supply have been processed to obtain seasonally adjusted data. In addition, the outlier effects have been adjusted in all four monthly series by applying the Tramo program. The four series after seasonal and outlier adjustments are given in Figs 1 through 8.

It is well known that oil price movement reflects changes in OPEC action. Since our intent is not to estimate the price impact of any OPEC policy, nor to model any structural changes, only monthly data after January 1986 have been used in our analysis. As a result, the comparison between VAR- and ANN-based forecasts becomes more meaningful.

Table 1 reveals changes in the mean, the standard deviation, and the coefficient of variation of each series resulting from seasonal and outlier adjustments. The summary statistics are derived from monthly data from January 1986 to November 2002. Data before January 1986 have been removed to avoid modeling complications. Note that the statistics of $M1$ stay the same under the columns of "no adjustments" and "seasonally adjusted," because $M1$ is a seasonally adjusted series. After seasonal and outlier adjustments, the coefficient of variation of oil

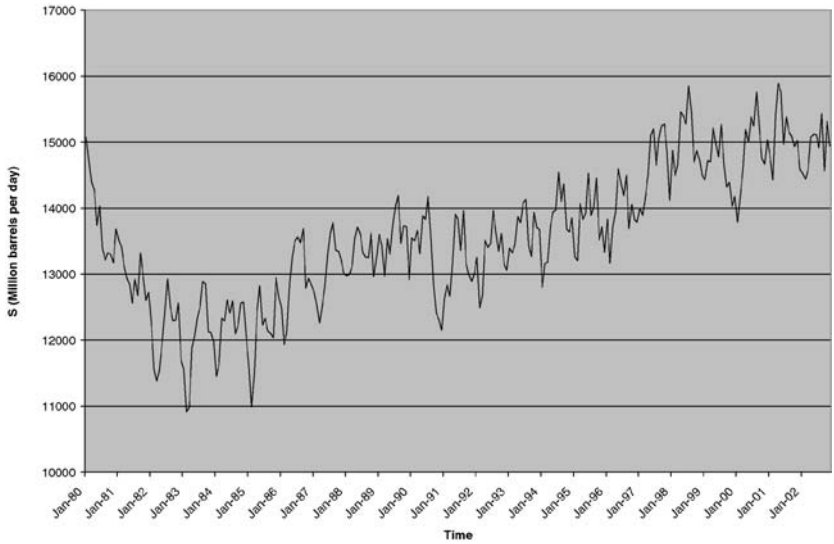


Fig. 2. Oil Supply (*S*). Original Data, January 1980 to November 2002. *Source:* U.S. Dept. of Energy, Energy Information Administration, Monthly Energy Review.

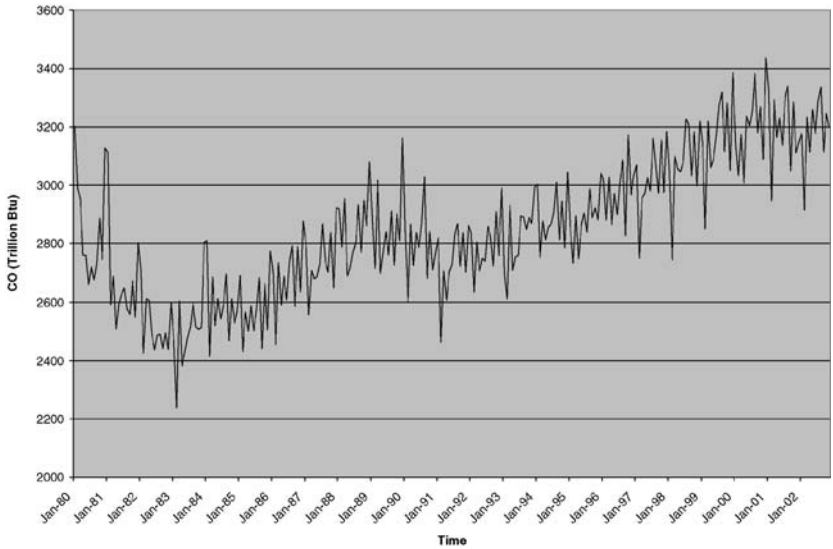


Fig. 3. Petroleum Consumption (CO). Original Data, January 1980 to November 2002. *Source:* U.S. Dept. of Energy, Energy Information Administration, Monthly Energy Review.

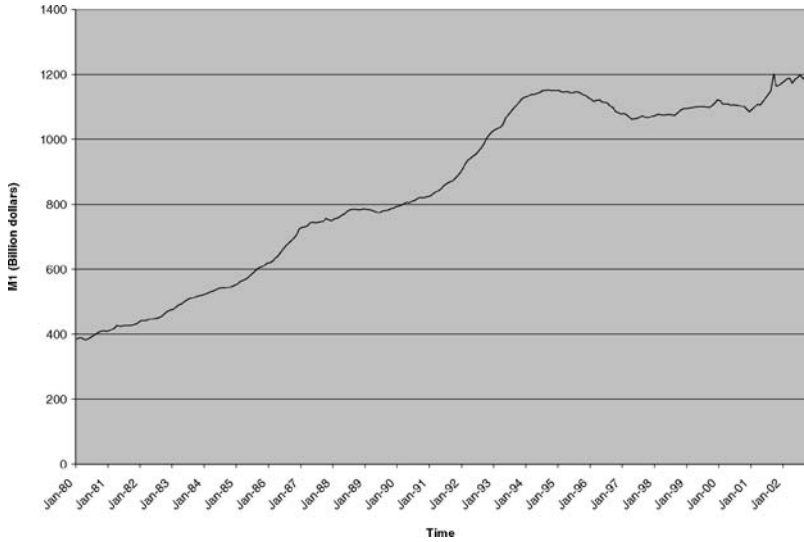


Fig. 4. Money Supply (*M*1). Original Data, January 1980 to November 2002. *Source:* The Federal Reserve System.

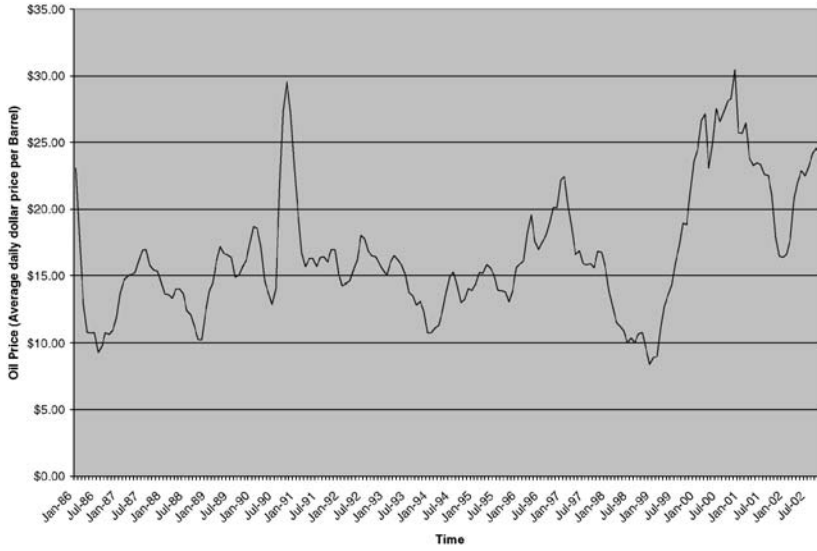


Fig. 5. Oil Price (*P*). Seasonally and Outlier Adjusted Data, January 1986 to November 2002.

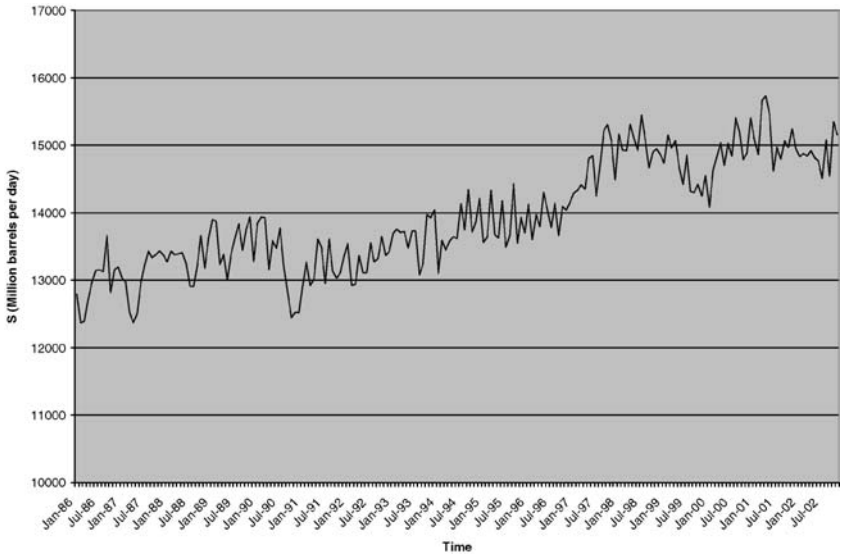


Fig. 6. Oil Supply (*S*). Seasonally and Outlier Adjusted Data, January 1986 to November 2002.

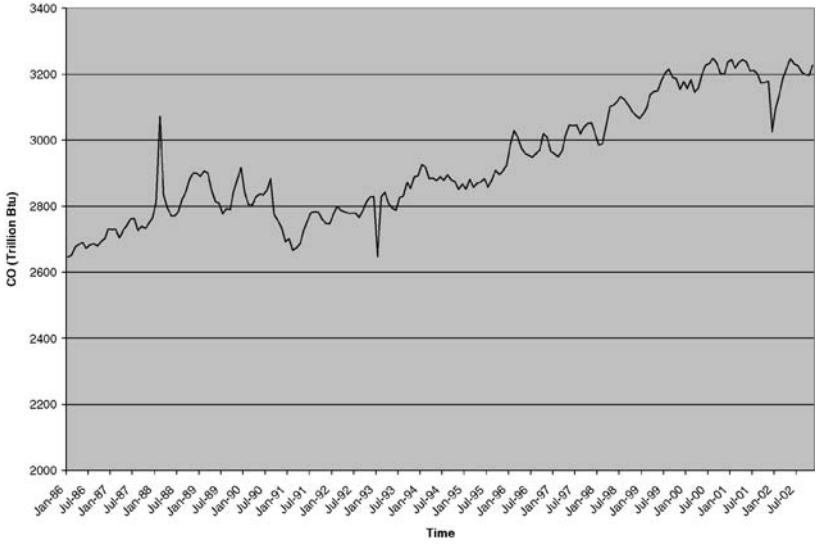


Fig. 7. Petroleum Consumption (CO). Seasonally and Outlier Adjusted Data, January 1986 to November 2002.

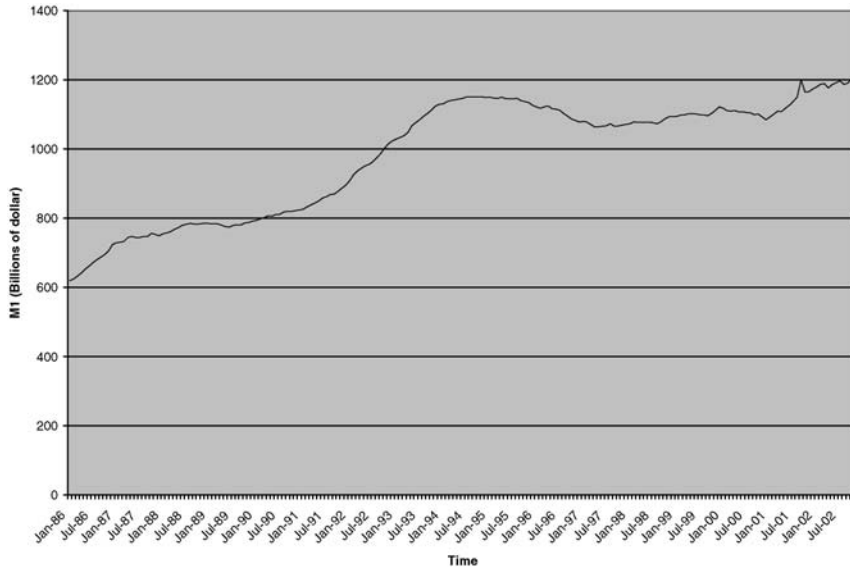


Fig. 8. Money Supply ($M1$). Seasonally and Outlier Adjusted Data, January 1986 to November 2002.

prices changes from 0.2857 to 0.2536. Statistics of other series shows only minor changes before and after the adjustments.

For comparing forecasting results, the data set has been divided into two subsets. The first sub-sample, covering a period from January 1986 to November 2001, is used for obtaining fitted values of the VAR and ANN models. To generate long-term ex-post forecasts, the data from December 2001 to November 2002 are used.

4. VAR MODELS

A VAR(p) model can be expressed as

$$\mathbf{X}_t = \boldsymbol{\alpha} + \boldsymbol{\beta}_1 \mathbf{X}_{t-1} + \cdots + \boldsymbol{\beta}_p \mathbf{X}_{t-p} + v_t, \tag{1}$$

where $\boldsymbol{\alpha} = (\alpha_1, \ldots, \alpha_k)'$ is a vector of intercepts; $\boldsymbol{\beta}_1, \ldots, \boldsymbol{\beta}_p$ are vectors of coefficients, $\mathbf{X}_{t-1}, \ldots, \mathbf{X}_{t-p}$ are vectors of variables with lags $1, \ldots, p$, respectively; and $v_t, \sim \text{WN}(0, \Sigma)$. The number of lags for each variable is determined by the Akaike Information and Schwartz criteria.

VAR has been considered as a very useful tool for economic forecasting. For a large sample, VAR estimates are fully efficient. Furthermore, the properties of the

Table 1. Variable Statistics.

Statistic	No Adjustments				Seasonally Adjusted				Seasonally Adjusted + Outliers Removed			
	P	*S*	CO	*M*1	*P*	*S*	CO	*M*1	*P*	*S*	CO	*M*1
Mean	16.66	13,943.07	2,931.99	984.65	16.65	13,941.28	2,932.52	984.65	16.35	13,955.28	2,929.48	985.67
Standard deviation	4.76	859.91	203.67	168.32	4.68	802.92	176.79	168.32	4.15	840.27	204.07	170.22
Coefficient of variation	0.2857	0.0617	0.0695	0.1709	0.2814	0.0576	0.0603	0.1709	0.2536	0.0602	0.0697	0.1727

Table 2. Tests of Unit Roots Hypothesis.

Variable	ADF Test	PP Test
P	−2.323187	−2.255479
CO	−1.644940	−1.107161
S	−1.910515	−2.539142
$M1$	−1.197474	−2.030140
ΔP	−8.630371	−8.390454
ΔCO	−16.60505	−21.49671
ΔS	−15.60040	−29.29778
$\Delta M1$	−2.858863	−12.96359

estimates are reliable. A VAR model also has the advantage that the estimate of its covariance matrix is independent of the estimates of its parameters. To examine the dynamic interactions among oil prices, petroleum consumption, oil supply and monetary policy, a VAR analysis is performed. E-Views is the software chosen for this task. To investigate the order of integration of these variables, the augmented Dickey-Fuller (ADF) test, the Phillips-Perron (PP) test, and the Kwiatkowski-Phillips-Schmidt-Shin (KPSS) test have been applied to the in-sample data.

Table 2 gives the results of augmented Dickey-Fuller test, Phillips-Perron test, and Kwiatkowski-Phillips-Schmidt-Shin test. Each variable and its first difference are subject to these tests. The ADF and the PP tests fail to reject the hypothesis that the P, CO, S, or $M1$ contains an autoregressive unit root. However, these tests reject the hypothesis that ΔP, ΔCO, or ΔS contains a unit root. For $\Delta M1$, the PP

Table 3. Johansen Cointegration Test.

Maximum Eigenvalues

Null Hypothesis	Eigenvalue	Maximum Eigenvalue Statistic	5% Critical Value	1% Critical Value
$r = 0$	0.205043	43.36929	20.97	25.52
$r = 1$	0.091872	18.21385	14.07	18.63
$r = 2$	0.006799	1.289427	3.76	6.65

Trace Statistics

Null Hypothesis	Eigenvalue	Trace Statistic	5% Critical Value	1% Critical Value
$r = 0$	0.205043	62.87258	29.68	35.65
$r = 1$	0.091872	19.50328	15.41	20.04
$r = 2$	0.006799	1.289427	3.76	6.65

Table 4. Vector Autoregression Estimates.

	P
P_{t-1}	0.896101
	(0.02503)
	[35.7980]
S_{t-1}	−0.001005
	(0.00027)
	[−3.69780]
CO_{t-1}	0.005355
	(0.00134)
	[3.99524]
R-squared	0.904227
Adj. *R*-squared	0.903203
Log likelihood	−335.8326
Akaike AIC	3.566659
Schwartz SC	3.617928

Note: Standard errors in ( ) & *t*-statistics in [ ].

test statistic is smaller than the *t*-statistic at both 5 and 10% levels, while the ADF statistic is greater than the *t*-statistic at the 5% level but smaller at the 10%. Results from all these tests suggest that each of the four series is an I(1) process.

The Johansen test has been applied to whether there exists any cointegrating equation. Results of this test are given in Table 3. The hypothesized number of cointegration equations is given in column 1. The next column shows eigenvalues. Maximum eigenvalue statistics and trace statistics are given in the third column. The 5% and 1% critical values are given in the last two columns. Results of this test suggest that the null hypothesis of no cointegrating equation is rejected. Both the maximum-eigenvalue test and the trace test suggest the presence of two cointegrating equations at the 5% level.

Based upon Akaike Information criterion, Schwartz criterion, and log likelihood ratio statistic, the best VAR model has been selected. This model includes *P*, CO, and *S* as endogenous variables. It is noteworthy that both the constant term and *M*1 have been excluded from the model. The VAR estimates which are used for forecasting oil price are shown in Table 4.

5. NEURAL NETWORK STRUCTURE AND GENETIC ALGORITHM

Perceptrons are the building blocs of a neural network. Each perceptron (also known as a neuron or node), alone, is capable of solving only linearly independent

problems. A feed-forward net is constructed by arranging banks of nodes into rows and linking elements of successive rows via weighted summation operators. A node in a higher row computes a linear combination of inputs sent from a lower row, and then forwards the result to an activation function (a nonlinear function if needed) to generate output. The activation function forces the output to be contained within finite bounds, with a range usually between 0 and 1, or −1 and 1. In a feed-forward net, the signal pathway is one direction – from the input layer to the output layer. Kinchem (1998) argues that the development of the back propagation paradigm in conjunction with the popular use of inexpensive high-powered personal computers brought neural networks technology to the mainstream.

Consider a three-layer network with an input layer, a hidden layer, and an output layer. Let $\mathbf{X} \in \Re^d$ be a d-dimensional stationary process. At iteration step t, $t = 1, 2, \ldots, n$, the input vector $x(t) \in \mathbf{X}$ at the input layer is connected with $r(t)$ nodes at the hidden layer to generate outputs $z(t) \in \mathbf{Z}$. This output vector is then fed to a node at the output layer. The output resulted from the output layer at iteration step t, $y(t) \in y$, is the outcome of the feed-forward system. If each neuron has an activation value that is the weighted sum of input values, then the network-based estimation function of $y(t)$, $\theta_n(\cdot)$, becomes

$$\begin{aligned}\theta_n(x(t), b(t), a(t)) &= \sum_i S_i\{Z_i(t), b_i(t)\} \\ &= \sum_i S_i \left(\sum_{ji} T_{ji}[x_{ji}(t), a_{ji}(t)], b_i(t) \right), \end{aligned} \tag{2}$$

where $b_i(t) \in b(t)$, $i = 1, 2, \ldots, r(t)$, is the hidden-to-output weight assigned to ith hidden node with $b_0(t)$ as the bias, and $a_{ji}(t) \in a(t)$, $i = 1, 2, \ldots, r(t)$ and $j = 1, 2, \ldots, d$, is the input-to-hidden weight assigned to the connection between jth input and ith hidden node with a_{0i} as the corresponding bias. Let $\theta \in \Theta$ be the parameter space, and $x_{ji}(t) \in x(t)$. Let $\theta_0 \in \Theta$ be the true parameter, and a, $b \in \theta$ be the estimates of θ_0. Here, S_i and T_{ji} are activation functions, a bounded function with infinite domain. A function may take the form of a signum, a logistic sigmoid, a cumulative distribution, a threshold logic, or a hyperbolic tangent function. Any change in activation functions would result in a different estimation of y.

Before a network can be used as a forecasting tool, it must be trained. Given a sample of $\{y(t), x(t)\}$ at iteration step t, the training problem is to select a, $b \in \theta$ so that an evaluative criterion such as the sum of squared errors is optimized. Note the difference between estimated and actual y at iteration step t by:

$$D(t) = y(t) - \theta_n(x(t), b(t), a(t)). \tag{3}$$

The weights assigned to inputs of the output layer at iteration step $t + 1$ become

$$b_i(t+1) = b_i(t) - \eta D(t) z_i(t), \quad i = 0, 1, 2, \ldots, r(t), \tag{4}$$

where η is the learning rate. For the hidden layer, the estimation error at iteration step t is

$$\delta(t) = D(t) S'\{z_i(t), b_i(t+1)\}, \quad i = 0, 1, 2, \ldots, r(t), \tag{5}$$

where S' is the first derive of S. The weights assigned to inputs of the hidden layer at iteration step $t + 1$ are

$$a_i(t+1) = a_{ji}(t) - \eta \delta(t) x_{ji}(t), \quad i = 0, 1, 2, \ldots, r(t), \quad \text{and} \quad j = 0, 1, 2, \ldots, d. \tag{6}$$

An architecture following this type of learning process is known as a back propagation network. The optimal estimate of y is obtained when the following risk functional is minimized:

$$L_n(\theta) = \int (y - \theta_n(x, b, a))^2 p(x, y)\, \mathrm{d}x\, \mathrm{d}y, \tag{7}$$

where $p(x, y)$ is the joint probability density function.

A neural net estimator contains an approximation error and an estimation error. The approximation error refers to the distance between the target function and the approximation function, while the estimation error measures the distance between the approximation function and the estimates. Cybenko (1989) and Hornik et al. (1989) suggest that a feed-forward network with sufficient hidden units is able to eliminate the approximation error. That is, feed-forward networks have the property of universal approximation. Barron (1993), Makovoz (1996), and Chen and White (1999) have proved that the approximation function approaches the target function at a convergence rate to be determined primarily by the number of nodes in a feed-forward network.

As aforementioned, the neural network estimator θ^* is a sieve extremum estimator, which satisfies

$$L_n(\theta^*) \geq \sup_{\theta \in \Theta_n} \quad L_n(\theta^*) - O(\varepsilon_t^2), \tag{8}$$

where Θ_n is a sieve such that $\{\Theta_n\}$ is dense in Θ as $n \to \infty$, and $\epsilon_t \to 0$, as $t \to \infty$. The neural network sieve θ_n, as shown in Eq. (1), is an approximation function in Θ_n. We can restate the approximation function in a more familiar form

$$\theta_n(x) = b_0 + \sum_{j=1}^{m} b_j \psi(a_j^T x + a_{0j}), \tag{9}$$

where $\psi(\cdot)$ is the activation function. Chen and Shen (1998) show that the neural network sieve convergence rate is $O_P([n/\log n])^{-(1+1/d)/[4(1+1/2d)]}$, and this convergence rate induces a root-n asymptotic normality of plug-in estimates.

Given the above theoretical structure, many algorithms can be applied for searching the optimal network architecture. Among them are linear programming, integer programming, constrained non-linear programming, unconstrained non-linear programming, non-linear least-square methods, heuristic search and genetic algorithms (GAs). Economists often use structural models based on economic theory. But when the structure of the problem domains is large and not well known, GAs have an advantage over traditional modeling methods (Polani & Uthman, 1993). That is, when the researcher has little a priori knowledge of the structure of problem domains, the combined power of GAs can still explore the search space efficiently (Al-Tabtabai & Alex, 1998). When GAs are incorporated into neural networks, the learning power and robustness of the network are greatly enhanced (Hansen & Meservey, 1996).

Holland (1975) proposed an iterative genetic algorithm to search the optimal solution to a fitness function or objective function. Each member in the population of possible solutions is a binary string of chromosomes with a fixed length. To initiate the search process, a population is created at random or in a heuristic manner. Each iteration of the search algorithm consists of an evaluation of the current population, and the formation of a replacement population. To evaluate a population, a selection operator that optimizes a fitness function is applied. A genetic algorithm can therefore be viewed as a repetitive optimization process that uses a selection operator to choose the most-fit population. There are several ways to end the search algorithm. The process can come to an end by simply fixing the total number of iterations, by reaching an approximate value within some tolerance level of the true solution to fitness function, or by establishing a criterion relevant to the application under consideration.

The creation of a replacement population at each iteration is essential to the search for the optimal solution. Variations to an existing population are generated by means of some chromosome recombination operators. The crossover operator is a powerful device for recombining chromosomes. During the crossover, two members of an existing population exchange portions of their binary chromosomes. To achieve the crossover effect, a crossover point on a string of chromosomes of each member is chosen. Then, two members exchange their binary sub-strings of chromosomes to the right of this crossover point. After the crossover, a new population has been created because each of its members contains a different string of chromosomes. Through this process, each iteration evaluates a different set of possible solutions to the objective function.

The population convergence over the iteration process implies that an optimal solution to the objective function is being approached. If the convergence rate is too high, however, the solution could be sub-optimal. To prevent fast convergence, a mutation operator can be applied to provide members with new genetic combinations. Thus, the mutation operator prevents premature convergence through population diversity.

A genetic algorithm can be viewed as a search process for identifying the optimal net structure. In this sense, each population corresponds to a neural network structure. Following Polani and Uthman (1993), each member of a population has the chromosomes consisting of two header bytes and "m" double bytes. The first header byte indicates the number of neurons or node in the net. The second header byte specifies the maximum number of transcription steps allowed by the modeler. A transcription step reflects a connection between one node and another. The double bytes are used to designate the possible node connections. The transcription process comes to a halt before the allowed maximum number of steps whenever it tries to connect a node with itself, or to reconnect the pre-connected nodes.

For the purpose of making an accurate forecast, the risk functional in our neural network model, as shown in Eq. (7), can serve as the fitness function of the GA search. That is, a neural network structure is considered optimal when its root mean squared error between predicted and observed output is the smallest among all nets.

This study uses Neuro Genetic Optimizer (NGO) to train and to select the optimal network. Among many types of applications that NGO provides, the time series prediction was selected. The software also offers a selection of neural network architectures, such as Back Propagation (BPN), Continuous Adaptive Time Delay (CATNN), Time Delay (TDNN), Probabilistic (PNN), Generalized Regression Neural Network (GRNN), and Self Organizing Map (SOM). For consistency purposes, BPN has been chosen for this study. Out of six available measures for fitness and accuracy, root mean squared error and mean absolute error have been chosen.

The databases are scaled by the software with the range of -1 to $+1$ for inputs. For output, depending on the transfer function, the scaling is from 0.1 to 0.9 if logistic sigmoid, and -1 to $+1$ if other functions are used. The initial weights are randomly selected between ± 0.3. This software allows a maximum of 256 hidden neurons per hidden layer. In addition, each network is allowed to have a maximum of two hidden layers. NGO has a capacity to support 32 input and 100 output variables.

Each iteration or run of a GA search involves a full cycle of evaluation/selection/mutation process. If not limited, GA would search for the optimum solution continuously without stopping. To prevent that, a minimum of 20 training

Table 5. Forecast Evaluation.

	$P_t = f(\cdot)$	Root Mean Squared Error	Mean Absolute Error
VAR	P_{t-1}, S_{t-1}, CO_{t-1}	4.69861	4.18883
BPN – GA	S_t, CO_{t-1}, $M1_t$	1.2354	0.8629

passes for each network were used. A new population is generated by cloning the survivors of an old population. We adopt the mutation technique of Tail Swap with a random exchange at a rate of 25%.

The neural network methodology requires the allocation of a portion of the data for training purpose. The training set includes the monthly data of each series from January 1986 to November 2001. The last twelve data points of each series are used for ex-post forecasting.

6. FINDINGS

Root mean squared error and mean absolute error have been chosen as the criteria for comparing VAR and BPN forecasts. The estimated coefficients of the VAR model is

$$P_t = \underset{(0.02503)}{0.896101}\,P_{t-1} - \underset{(0.00027)}{0.001005}\,S_{t-1} + \underset{(0.00134)}{0.005355}\,CO_{t-1}. \tag{10}$$

R-squared = 0.904227, Adjusted R-squared = 0.903203.

The GA-based back propagation (BPN-GA) model, however, employs all three variables, S, CO and $M1$, to forecast the price of oil (P). There is one hidden layer with 1 logistic, and 1 tan h neurons. There is one output neuron using the logistic transfer function.

Table 5 shows root mean squared error and mean absolute error of VAR and BPN-GA models. Variables used for different modeling techniques are given in the second column. The subsequent columns provide root mean squared error and mean absolute error of each model. Based on the forecast evaluation statistics, the BPN-GA model noticeably outperforms the VAR model.

7. DISCUSSION

The Middle East conflict has heightened the concern about global oil supply. At the heart of this concern is the future energy prices and their reliable forecasts. The

selection of a forecasting technique becomes an important component of policy making.

The VAR model selected for forecasting suggests that lagged oil price is the most important variable in forecasting price movement. Interestingly, $M1$ does not play any role in this model. A possible explanation is that money supply is not an appropriate proxy for energy policy. For future studies, the search for an alternative policy variable is in order.

A closer look of the VAR model further reveals that the coefficient of lagged oil supply is -0.001005, and that of lagged energy consumption is 0.005355. Both coefficients are statistically significant. The results also confirm the economic theory that an increase in oil supply has a dampening effect on, while the energy consumption is positively related to the oil price. However, these coefficients suggest that there are minimal impacts of energy consumption and oil supply on oil price.

Recently, the application of the artificial neural networks in forecasting has gained popularity in the fields of business and economics. Models based on neural networks have the advantage of making less restrictive assumptions. Neural networks are also capable of approximating any continuous mapping with desirable accuracy. This study suggests that a BPN-GA model can outperform a VAR model in forecasting. However, our study does not prove that ANN is always a better forecasting technique than VAR.

ANN techniques, however, also have some shortcomings. One of the problems is that they do not emphasize the functional form of a model. Unlike a VAR model, coefficients of individual variables are not known. In addition, the selection of variables to be included does not follow the common practices in econometrics. For example, unit root tests do not find a place in ANN models.

ACKNOWLEDGMENTS

We wish to thank anonymous reviewers for their valuable suggestions. Also we would like to extend our gratitude to Mr. Matthias L. Lippmann for his research assistance.

REFERENCES

Al-Tabtabai, H., & Alex, A. P. (1998). An evolutionary approach to the capital budgeting of construction projects. *Cost Engineering*, *40*, 28–34.

Barron, A. R. (1993). Universal approximation bounds for superposition of a sigmoidal function. *IEEE Transactions on Information Theory*, *39*, 930–945.

Barron, A. R. (1994). Approximation and estimation bounds for artificial neural networks. *Machine Learning*, *14*, 115–133.

Bernanke, B., Gertler, M., & Watson, M. (1997). Systematic monetary policy and the effect of oil price shocks. *Brookings Papers on Economic Activity*, *1*, 91–142.

Bode, J. (1998). Neural networks for cost estimation. *Cost Engineering*, *40*, 25–30.

Chen, S.-H. (Ed.) (2002). *Genetic algorithms and genetic programming in computational finance*. Boston: Kluwer.

Chen, X., & Shen, X. (1998). Sieve extremum estimates for weakly dependent data. *Econometrica*, *66*, 289–314.

Chen, X., & White, H. (1999). Improved rates and asymptotic normality for nonparametric neural network estimators. *IEEE Transactions on Information Theory*, *45*, 682–691.

Cheng, B., & Titterington, D. M. (1994). Neural networks: A review from a statistical perspective. *Statistical Science*, *9*, 2–30.

Cooper, J. B. (1999). Artificial neural networks vs. multivariate statistics: An application from economics. *Journal of Applied Statistics*, *26*, 909–921.

Curry, B., & Peel, M. J. (1998). Neural networks and business forecasting: An application to cross-sectional audit fee data. *International Journal of Commerce and Management*, *8*, 94–120.

Cybenko, G. (1989). Approximation by superposition of a sigmoidal function. *Mathematics of Control, Signals and Systems*, *2*, 303–314.

Dawid, H. (1996). *Adaptive learning by genetic algorithms*. Berlin: Springer.

Denton, J. W. (1995). How good are neural networks for causal forecasting? *The Journal of Business Forecasting Methods & Systems*, *14*, 17–23.

Emerson, S. (2000). Oil price: The new dynamic – recent oil price trends underscore OPEC's unwieldy market power. *Oil and Gas Journal*, *98*, 89–90.

Evans, M. (2000). Don't oil prices matter any more? *Industry Week*, *249*, 76.

Geman, S., & Hwang, C. R. (1982). Nonparametric maximum likelihood estimation by the method of sieves. *Annals of Statistics*, *10*, 401–414.

Gisser, M., & Goodwin, T. H. (1986). Crude oil and the macroeconomy: Test of some popular notions. *Journal of Money, Credit and Banking*, *18*, 94–103.

Grudnitski, G., & Osburn, L. (1993). Forecasting S&P and gold futures prices: An application of neural networks. *The Journal of Futures Markets*, *13*, 631–644.

Hamilton, J. (1996). This is what happened to the oil price-macroeconomy relationship. *Journal of Monetary Economics*, *38*, 215–220.

Hansen, J. V., & Meservey, R. D. (1996). Learning with the genetic optimization of a generalized regression neural network. *Decision Support Systems*, *18*, 317–325.

Holland, J. H. (1975). *Adaptation in natural and artificial systems*. Ann Arbor: University of Michigan Press.

Hooker, M. (1996). What happened to the oil price-macroeconomy relationshp? *Journal of Monetary Economics*, *38*, 195–213.

Hornik, K., Stinchcombe, M., & White, H. (1989). Multilayer feed-forward networks are universal approximators. *Neural Networks*, *2*, 259–366.

Jaffe, A. M., & Manning, R. A. (2000). The shocks of a world of cheap oil. *Foreign Affairs*, *79*, 16–29.

Johnson, D. (2002). Running out of oil. *The Futurist*, *36*, 14–15.

Kinchem, K. (1998). Neural network technology moves toward mainstream control use. *Pulp & Paper*, *72*, 63–68.

Kudyba, S. (1998). Are neural networks a better forecaster? *Futures*, *27*, 52–59.

Kuo, C., & Reitsch, A. (1996). Neural networks vs. conventional methods of forecasting. *The Journal of Business Forecasting Methods & Systems*, *14*, 17–25.

Kyungdoo, N., Yi, J., & Prybutok, V. R. (1997). Predicting airline passenger volume. *The Journal of Business Forecasting Methods & Systems*, *16*, 14–16.

Lawrence, F. (2002). Learning from the past – strategizing for future. *Oil and Gas Investor*, 9–11.

Linden, H. R. (1998). Flaws seen in resource models behind crisis forecast for oil supply, price. *Oil and Gas Journal*, *96*, 33–37.

Lippmann, R. P. (1992). Introduction to computing with neural nets. In: C. Lau (Ed.), *Neural Networks: Theoretical Foundations and Analysis* (pp. 5–23). New York: IEEE Press.

Makovoz, Y. (1996). Random approximants and neural networks. *Journal of Approximation Theory*, *85*, 98–109.

Matsui, K. (1998). Global demand growth of power generation, input choices and supply security. *Energy Journal*, *19*, 93–107.

McMenamin, S., & Monforte, F. A. (1998). Short term energy forecasting with neural networks. *The Energy Journal*, *19*, 43–61.

Morana, C. (2001). A semiparametric approach to short-term oil price forecasting. *Energy Economics*, *23*, 325–338.

Moshiri, S., Cameron, N. E., & Scuse, D. (1999). Static, dynamic, and hybrid neural networks in forecasting inflation. *Computational Economics*, *14*, 219–235.

Pain, N. et al. (2000). The world economy. *National Institute Economic Review*, *171*, 36–69.

Pindyck, R. (1999). The long run evolution of energy price. *The Energy Journal*, *20*, 1–27.

Polani, D., & Uthman, T. (1993). Training Kohonen feature maps in different topologies: An analysis using genetic algorithms. In: S. Forrest (Ed.), *Proceedings of the Fifth International Conference on Genetic Algorithms* (pp. 326–333). San Mateo, CA: Morgan Kaufmann.

Raymond, J. E., & Rich, R. W. (1997). Oil and the macroeconomy: A markov state-switching approach. *Journal of Money, Credit, and Banking*, *29*, 193–213.

Ripley, B. D. (1993). Statistical aspects of neural networks. In: J. Barndorff-Nielsen, L. Jensen & W. S. Kendall (Eds), *Networks and Chaos – Statistical and Probabilistic Aspects* (pp. 40–123). London: Chapman & Hall.

Sarantis, N., & Stewart, C. (2000). The ERM effect, conflict and inflation in the European Union. *International Review of Applied Economics*, *14*, 25–43.

Shen, X. (1997). On methods of sieves and penalization. *Annals of Statistics*, *25*, 2555–2591.

Shen, X., & Wong, W. H. (1994). Convergence rate of sieve estimates. *Annals of Statistics*, *25*, 580–615.

Smith, A. E., & Mason, A. (1997). Cost estimation predictive modeling: Regression vs. neural network. *The Engineering Economist*, *42*, 137–161.

Weinert, G. (2002). Global economy in the doldrums. *Intereconomics*, *37*, 59–64.

White, H., & Wooldridge, J. (1991). Some results for sieve estimation with dependent observations. In: W. Barnett, J. Powell & G. Tauchen (Eds), *Nonparametric and Semiparametric Methods in Economics* (pp. 459–493). New York: Cambridge University Press.

Wong, B. K., & Yakup, S. (1997). Neural networks applications in business: A review and analysis of literature (1988–1995). *Decision Support Systems*, *34*, 301–320.

Wong, B. K., & Yakup, S. (1998). Neural network applications in finance: A review and analysis of literature (1990–1996). *Information and Management*, 129–139.

Zhang, G., & Hu, M. (1998). Neural network forecasting of the British pound/U.S. dollar exchange rate. *Omega*, *26*, 495–506.

SEARCHING FOR DIVISIA/INFLATION RELATIONSHIPS WITH THE AGGREGATE FEEDFORWARD NEURAL NETWORK

Vincent A. Schmidt and Jane M. Binner

ABSTRACT

Divisia component data is used in the training of an Aggregate Feedforward Neural Network (AFFNN), a general-purpose connectionist system designed to assist with data mining activities. The neural network is able to learn the money-price relationship, defined as the relationships between the rate of growth of the money supply and inflation. Learned relationships are expressed in terms of an automatically generated series of human-readable and machine-executable rules, shown to meaningfully and accurately describe inflation in terms of the original values of the Divisia component dataset.

1. INTRODUCTION

If macroeconomists ever agree on anything, it is that a relationship exists between the rate of growth of the money supply and inflation. According to the Quantity Theory tradition of economics, the money stock will determine the general level of

Applications of Artificial Intelligence in Finance and Economics
Advances in Econometrics, Volume 19, 225–241

ISSN: 0731-9053/doi:10.1016/S0731-9053(04)19009-9

prices (at least in the long term) and according to the monetarists it will influence real activity in the short run. This relationship has traditionally played an important role in macroeconomic policy as governments try to control inflation.

Measuring money supply is no easy task. Component assets range from "narrow money," which includes cash, non interest bearing demand deposits and sight deposits[1] on which cheques can be drawn, to "broader" money, which includes non-checkable liquid balances and other liquid financial assets such as Certificates of Deposit. In practice, official measures have evolved over time according to policy needs. Obviously many of these assets yield an interest rate and could thus be chosen as a form of savings as well as being available for transactions. Financial innovation, in particular liberalisation and competition in banking, has led to shifts in demand between the components of "money" which have undermined earlier empirical regularities and made it more difficult to distinguish money which is held for transactions purposes from money which is held for savings purposes (Mullineux, 1996).

Secondly, there is the question of how to combine the different components, since they are not perfect substitutes for one another. They provide different levels of monetary services for transactions (liquidity) and different yields (interest). As payments technology progresses, so the monetary services provided will change disproportionately – not only do interest rates vary with time but liquidity can be enhanced too. Despite this variation in the characteristics of assets, the conventional way of measuring the amount of money circulating in an economy is to simply sum the various constituent liquid liabilities of commercial and savings banks. However the belief is widely established that this method of arriving at broad money aggregates is seriously flawed and based on untenable assumptions, as shown by Belongia (1996). From a micro-demand perspective it is hard to justify adding together component assets having differing yields that vary over time, especially since the accepted view allows only perfect substitutes to be combined as one "commodity."

Barnett (1978, 1980) pioneered the use of the Divisia monetary aggregate as an alternative to simple sum aggregation. By drawing on statistical index number theory and consumer demand and aggregation theory, he advocated the use of the Divisia chain-linked index numbers as a means of constructing a sophisticated weighted index number measure of money. The Divisia index formulation has been well documented in the literature and so need not detain us here (see Fisher et al., 1993; Mullineux, 1996, for detailed discussions on the construction of Divisia monetary aggregates and associated problems). Proponents of weighted index number aggregation contend that Divisia M4 endogenizes at least some of the major innovations that clearly distorted simple sum M4 in the 1980s, especially the payment of competitive interest rates on checking deposits.

The Divisia approach attempts to allow for the varying transactions and liquidity properties of the components of a monetary aggregate by giving them different weights. The share weight of each component depends on its size relative to the other components of the monetary aggregate and on its user cost. Consequently, currency and non-interest bearing deposits receive the highest weights because they are highly liquid assets and (correspondingly) have high user costs (in terms of interest foregone). Interest-bearing time deposits by contrast pay a relatively high rate of interest and are less liquid, and so attract a lower weight than might be expected from the size of such deposits alone. To the extent that these weights reflect the differences in transactions services provided by various monetary assets, the resulting Divisia index should be more closely related to the overall level of liquidity in the economy (and therefore to total spending in the economy) than are conventional monetary aggregates.

Clearly, the foundations of the construction of monetary aggregates are well rooted in monetary aggregation theory and require extremely strong assumptions. (Barnett and Serletis (2000) give a detailed treatment of the theory of monetary aggregation.) However, the underlying philosophy of the current research is that all assumptions can be weakened and the Divisia formulation can still be improved. Our hypothesis is that the interest rate weights are not ideal in Divisia construction and the Divisia index number formula has not yet been optimized. This is also borne out by recent research which has focused on accounting for the riskiness of the asset in the index construction; see Barnett et al. (Barnett, 1995; Barnett et al., 1997) for such efforts in the USA and Drake et al. (1999) and Elger and Binner (2004) for approaches in the UK.

To work toward this hypothesis, our primary interest is to identify the relationships between Divisia components and inflation. One promising approach is the use of artificial neural networks, or connectionist models, which can model relations from examples, and are robust with respect to noise and random fluctuations in training data. This robustness allows neural solutions to generalize such that the final state is accurate across a wide range of data. We believe neural networks can be used as a tool to identify the true weights and relationships between monetary assets and prices. If the discovered relationships prove to be reliable indicators, a construction of superior weighted monetary indices could ultimately reinstate monetary targeting as the mainstay of macroeconomic control.

Trained neural networks are ideal solutions in a variety of domains, and the popularity of connectionist systems has been a catalyst in the introduction of a large collection of inexpensive or freely available neural modeling tools. Coupled with rapid and continuous advances in computer technology and the availability of inexpensive, but powerful, desktop computers, many casual researchers are able to take advantage of these tools. In many cases, reasonable experimental results

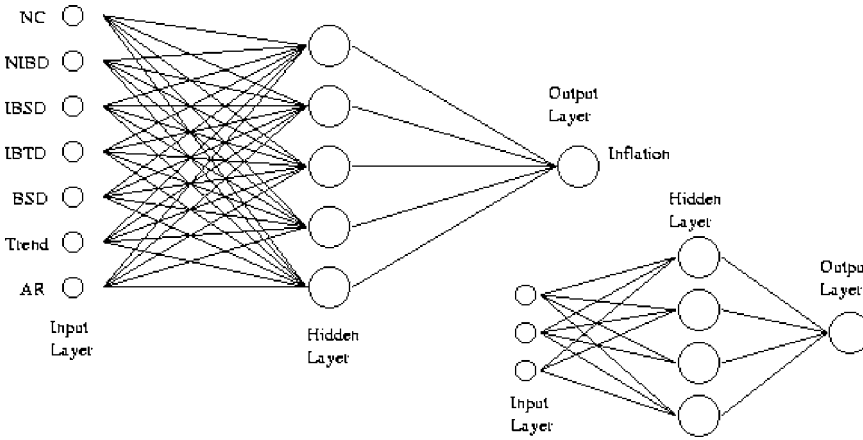


Fig. 1. Feedforward Connectionist Models.

can be quickly produced. The resulting models and their solutions can be useful in lending insight into the behavior and relationships among data elements and attributes.

The use of neural networks to learn the relationship between Divisia asset components and inflation was first attempted very successfully for UK Divisia M4 by Gazely and Binner (2000). Their 7–5–1 and 3–4–1 feedforward models are shown graphically in Fig. 1, and include Divisia components, a time trend variable, and an autoregressive term. Weights from the trained network were used in a sensitivity analysis to determine the importance of attribute inputs. They demonstrated that a properly generated neural network is expressive enough to learn relationships between Divisia components and inflation to the extent that even a reasonably simple architecture can outperform traditional Divisia measurements under many circumstances, and performs comparably at its worst.

These connectionist models were sufficiently expressive to learn the relationships from Divisia components to inflation. However, we are not currently interested in only devising a system that learns such relationships, but in a system that also describes them in useful and practical terms. This is the role of a solution such as the Aggregate Feedforward Neural Network (AFFNN) approach, which specializes in learning the relationships using a customized neural network, and then presents the knowledge as human-readable and machine-executable rules.

The purpose of the AFFNN architecture is to assist in the exploration and explanation of data, supporting data mining activities. It was selected for this effort because of its generality and existing rule extraction subsystem. The AFFNN specializes in examining a collection of K input attributes and discovering the

relationships amongst all attributes simultaneously. The resulting relationships are expressed as sets of rules. Key descriptions of this novel model were introduced, with applications, in Schmidt and Chen (Schmidt & Chen, 2002a, b). Characteristic features of this model include:

- The AFFNN is a novel architecture that behaves as if it were an aggregation of a collection of individual neural networks.
- The AFFNN uses all attributes in both the source and target sets, but is not autoassociative because of the way that the inputs are managed.
- The AFFNN can use contemporary supervised training algorithms and performance functions suitable for feedforward connectionist models with little or no modification.
- A customized decompositional rule extraction algorithm was developed for use with the AFFNN. Discovered relationships are algorithmically transformed into readable and executable rules.

The remainder of this paper briefly summarizes key points of the AFFNN, and then describes the model's application to learning the relationships among the monetary component asset data of UK Divisia M4.[2] Application-specific AFFNN architectural and training details are presented, along with a brief comparison to a traditional non-rule-bearing feedforward neural model and a discussion of the experimental results. This study is a continuation of a collaborative research effort originating in Schmidt and Binner (2003).

2. LEARNING DIVISIA RELATIONSHIPS USING THE AFFNN

The AFFNN is a fully-connected feedforward connectionist model that relies on the use of traditional supervised training algorithms. Although the architecture is not constrained, the most commonly used architecture contains a single hidden layer with an empirically determined number of hidden layer nodes. Both the input and output layers have the same number of nodes, represented by the size of the vector encoding all K attributes. The nature of the AFFNN design allows the user to select practically any supervised training algorithm and performance function.

By design, the AFFNN accepts all K attributes as network inputs, and produces all K attributes as outputs. However, the network is not trained autoassociatively. Each attribute A_i, $1 \leq i \leq K$, is learned by the network such that all other attributes are permitted to contribute to the solution for A_i except for A_i itself. Expressed more formally, if A_{i^*} represents the set of all attributes such that all attributes except

for A_i are included in A_{i^*}, and if $O_{\hat{i}}$ defines the set of attributes only including A_i, then the AFFNN strives to simultaneously learn all relationships:

$$O_{\hat{i}} = f(A_{i^*}), \quad \forall 1 \leq i \leq K \tag{1}$$

for all values of i for all such existing functions, given adequate expressive power in the hidden layer and a sufficient amount and quality of training.

The AFFNN is a general-purpose connectionist architecture that is able to perform such learning with the assistance of an exclusive performance function wedge and a set of pre-and post-processing functions. These functions, not accessible by the user, manage the internal functionality of the AFFNN model. This unique feature allows it to operate in a variety of domains without restricting the selection of supervised training functions or network performance functions available to the end user. More detail regarding the specific implementation of the AFFNN can be found in the definitive work by Schmidt (2002).

In this effort the AFFNN is used as a tool for discovering relationships within the Divisia component data, specifically the relationship between Divisia components and inflation. The raw dataset is obtained as a collection of quarterly figures from Q1 of 1977 through Q1 of 2001, yielding 97 exemplar vectors. This data is preprocessed by computing the percentage of change for each successive and corresponding quarter (the same quarter of the previous year), which reduces the data to 94 exemplars. Once the component data is preprocessed, it is clustered and thermometer-encoded in preparation for use in the AFFNN. The clustering algorithm used for this effort is fully automated (completely hands-off), and is a customized approach based on the standard deviation of the differences among the numbers to be grouped. The algorithm is documented in Section 5.2.1 of Schmidt (2002). Applying this algorithm to the Divisia dataset, the number of clusters for each attribute is:

- Notes and Coin (NC) encoded into 7 levels;
- Non Interest Bearing Bank Deposits (NIBD) encoded into 14 levels;
- Interest Bearing Bank Sight Deposits (IBSD) encoded into 4 levels;
- Interest Bearing Bank Time Deposits (IBTD) encoded into 7 levels; and
- Building Society Deposits (BSD) encoded into 7 levels.

The six attributes used to train the AFFNN consist of these five, along with inflation rate (INFL), also expressed as a percentage of increase, automatically clustered and encoded into 4 levels. Since these attributes are thermometer-encoded, the input vector is 43 binary elements: 39 from the original encoded components and four from the encoded inflation rate.

Figure 2 shows a block diagram depicting a 43–12–43 AFFNN topology very similar to the architecture used in this study. After empirically determining that

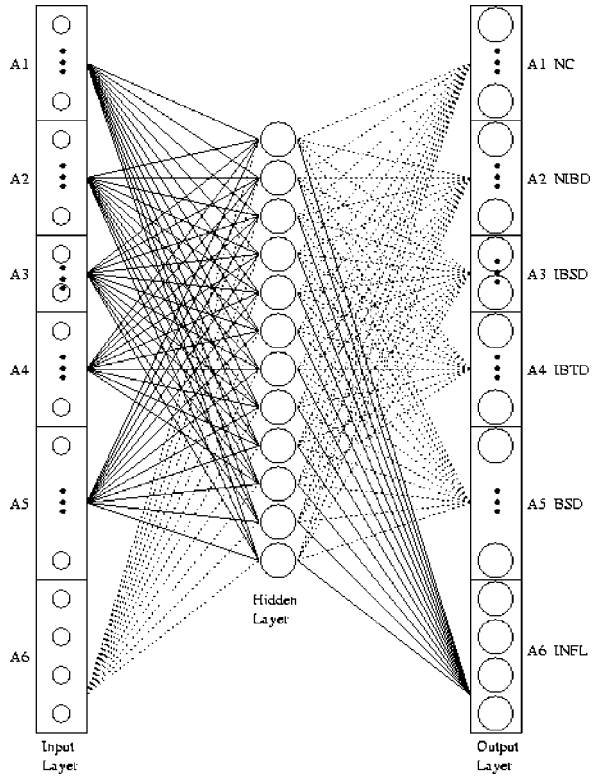


Fig. 2. Divisia AFFNN Block Diagram.

15 nodes in the hidden layer is sufficiently expressive, the AFFNN used for this study is a 43–15–43 model (43 nodes in the input and output layers, with 15 nodes in the hidden layer). Note that the AFFNN uses all six encoded attributes as both inputs and outputs (the input and output layer sizes are the same). The nodes in the hidden layer use sigmoid activation functions and the output layer nodes use linear activation functions. Both the number of network layers and the selection of activation functions used in this effort conform to research demonstrating that a feedforward neural network containing a single hidden layer (using sigmoid activation functions) and a single output layer (using linear activation functions) can be used to approximate virtually any function of interest to any degree of accuracy when using an adequate number of nodes in the hidden layer (Hornick et al., 1989). Thus, defining a network with a single hidden layer is a practical decision that is not a limitation of the AFFNN, which can be created with an arbitrary number of layers and any set of activation functions.

The nature of the AFFNN allows inflation (INFL) to be learned purely as a function of the other attribute variables. Although the AFFNN attempts to learn all of the relationships for all attributes, it is always possible to examine the paths leading to a single attribute, effectively "lifting" the desired portion (inflation, in this case) out of context following network training. This allows the rule extraction routines to concentrate on describing the relationships in this more limited series of outputs. For example, rules are generated using the weights and values propagated along the solid lines of the figure, while weights on the dashed lines are ignored. Here, A6 (INFL) is the only attribute receiving weights at the output nodes, but it does not contribute as an input.

3. GENERATING RULES FROM LEARNED RELATIONSHIPS

It is not sufficient to merely learn the relationships; they must also be expressed in some format that is both understandable and reasonable. Neural networks excel at learning relationships, but are frequently declared to be black boxes since casual examination of trained network weights is not practical. Thus, it is often assumed that the inner workings of connectionist systems are incapable of being practically expressed. Fortunately this is not necessarily true. There are numerous research activities that strive to describe the neural model in understandable terms, although many contemporary systems require specialized architectures or operate in limited domains (see Lu et al., 1995; Towell & Shavlik, 1993, for particularly interesting systems). In fact, examination of the relevant literature shows the existence of a variety of techniques for transforming trained connectionist models into intelligible forms. For example, some methods generate decision trees or statistical models from the trained neural network. The decompositional approach, where the trained model's connections and internal weights are examined, is also a popular form.

Rule extraction in the AFFNN uses a technique which is similar to basic decompositional strategies frequently employed to describe neural networks. The algorithm uses a multistaged approach to inspect discretized activation values within the trained network model. Rules are then created by identifying similar weighting structures and combining the activation values within the trained AFFNN. The final product of the decompositional extraction is a collection of *if-then* rules based on the solution set of satisfied expressions, where each rule describes a single expression. A *set* of rules describes the entire solution, and inputs satisfying *any* rule belong to the solution set.

As an additional advantage, the rules are expressed in an appropriate syntax such that they are directly executable by a computer. Thus it is not necessary to "code" or "translate" the rules into a valid computer language. These rules can be executed against the training or testing data to confirm rule validity and can also be used to process previously unseen data. In the case of the Divisia component data, this makes the extracted rules available for performing predictions and other forecasting tasks.

Figure 3 illustrates the basics of this technique, as applied to a typical AFFNN architecture. For this example, assume that and AFFNN is trained for three attributes, A, B, and C, encoded into multiple input variables, and that rules are to be produced for attribute C, with values encoded into two nodes. The only weights used in extracting these rules are indicated by the solid lines in the figure. Note how entries related to attribute C are not participating as inputs to the network, and only those nodes related to attribute C are considered at the output layer (since the AFFNN is not autoassociative).

Each node in the hidden layer produces values that are passed on to the output layer. These values can be clustered into sets, as represented by the sets labeled h1–h7 in the figure. Certain combinations of these values yield the desired output at the selected output node, exemplified as the two sets (expressions) at the right of the figure. The extraction algorithm is able to identify these "candidate expressions" by working in reverse. Starting with the selected output node, the algorithm identifies potential solutions forming the specified output range. The clusters of activation

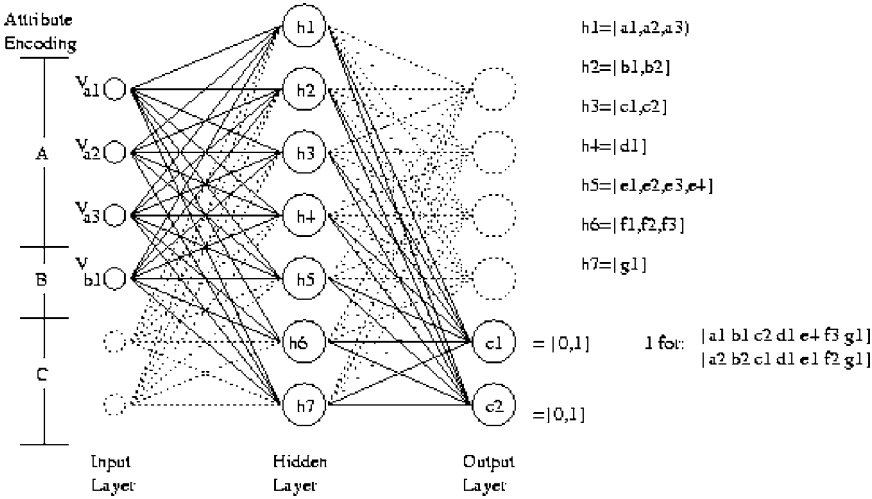


Fig. 3. Decompositional Extraction.

values (represented by h1–h7 in the figure) generated by the nodes in the hidden layer that produce the desired results are gathered into combinations that form "candidate expressions," matching the solution set.

This procedure is repeated again to determine the inputs supporting the individual terms in the candidate expressions. Combinations of inputs producing all terms in the candidate expression will satisfy the candidate expression, and therefore also satisfy the solution set. This is a brief overview of the extraction technique, which is a repeatable, deterministic, and algorithmic approach. Specific implementation and more detailed discussion can be found in Schmidt (2002) and Schmidt and Chen (2002b). This stage of the process yields a set of satisfied expressions, which are encoded by an automated encoding process.

The custom rule generation function algorithmically translates the satisfied expressions into MATLAB code describing the solution set. The rule generator was deliberately coded to produce rules describing outputs with respect to the original input values, expressing the rules in a notation that is easy to read and validate by subject-matter experts. The rules are represented using MATLAB as a notation, allowing them to be executed as an independent product, separate from the original AFFNN.

The Divisia data exhibits the expected results when processed by the AFFNN. An automated clustering algorithm assists in dividing the inflation data (percentage change in growth rate of price level) into four distinct levels:

$$L_1 = (-0.0033, 0.0124]$$
$$L_2 = (0.0124, 0.0558]$$
$$L_3 = (0.0558, 0.1116]$$
$$L_4 = (0.1116, \infty)$$

When trained on the dataset, the AFFNN learns the relationships leading to each attribute,[3] including the function describing the relationships of the Divisia components to the level of inflation:

$$f(\text{NC, NIBD, IBSD, IBTD, BSD}) \Rightarrow L_x, L_x \in \{L_1, L_2, L_3, L_4\} \tag{2}$$

Once the AFFNN learns this relationship, the decompositional rule extraction mechanism is used to produce a series of *if-then* rules describing the discovered relationships in terms of original attribute values. This makes the resulting rule base very easy to comprehend by econometricians or similar subject-matter experts, a much more practical task than attempting to directly understand the internal weights of a trained neural network model.

The example rule shown in Fig. 4 is a single expression for a solution the AFFNN learned to describing L_3, the third level of the inflation relationship. The

```
if {
        (nc <= –0.004833)
        & ((nibd <= –0.091788)
                | (nibd > –0.010042
                        & nibd <= 0.049695)
                | (nibd > 0.082643))
        & (ibsd <= 0.148376)
        & (ibtd > 0.008503)
        & (bsd > 0.082032)
} then return true;
```

Fig. 4. Example Expression for L_3.

rule extraction algorithm yields 408 distinct expressions for completely describing this relationship. Any set of values of NC, NIBD, IBSD, IBTD, and BSD meeting the requirements of this expression makes the expression true. If one or more of the 408 expressions comprising the solution set for L_3 is true, then that set of values is a valid solution for L_3 and satisfies the statement "Inflation percent change is in the range (0.0558, 0.1116]."

4. EXPERIMENTAL METHOD AND RESULTS

The AFFNN topology was designed for 43 inputs, as described in Section 2, representing the encoding of all attributes of the Divisia dataset. Training was performed using an AFFNN model developed with the MATLAB Neural Network toolbox, and executed on a Linux-based dual Pentium-II (300 MHz) computer with 128 Mb of RAM and 1 Gb of swap space.

Following the example of the earlier experiments by Gazely and Binner (2000), the dataset was randomly divided into a large training set of 75 exemplar vectors (80% of the data) and a smaller testing set of 19 exemplar vectors (20% of the data). The training and test sets were held constant for all experiments described in this paper.

Two distinctly different AFFNN architectures were trained using the Divisia dataset, identical except for the representation of the encoding used for the INFL attribute. This test examines the effects of using slightly different encoding mechanisms on AFFNN architectures that are otherwise similar. In both cases, attribute data for NC, NIBD, IBSD, IBTD, and BSD were thermometer-encoded based on the outcome of the automated data clustering algorithm. The first AFFNN also thermometer-encoded the INFL attribute, while the second AFFNN used 1-of-4 encoding. Both AFFNN models used sigmoid activation functions in the hidden layer and linear activation functions in the output layer. The number of hidden nodes for each model was determined empirically.

Table 1. AFFNN Experimental Results.

Network	Topology	INFL Encoding	Train/Test Accuracy in %	# Rules (L_1 / L_2 / L_3 / L_4)
1st AFFNN	43–15–43	Thermometer	93.3/84.2	573 / 566 / 408 / 0
2nd AFFNN	43–16–43	1-of-N	89.3/8.9	39 / 365 / 149 / 0

Table 1 shows the experimental results of creating and training these two AFFNN models. The first AFFNN was designed with a 43–15–43 topology. Training mean-squared error (MSE) was 0.0079 after 4000 epochs (943) seconds of training using the custom MATLAB AFFNN model. As indicated in the table, the trained network properly classified 93.3% of the training data and 84.2% of the testing data for the INFL attribute. The number of rules extracted for the four levels of INFL were, respectively, 573, 566, 408, and 0.

A second AFFNN was designed with a 43–16–43 topology (using one additional node in the hidden layer). For this model, the best candidate of 500 networks was selected, where each network was trained for 2500 iterations (approximately 550 seconds of training per trial). The best candidate was able to correctly classify 89.3% of the training data and 78.9% of the test data. Although accuracy was marginally reduced, rule comprehensibility improved due to a decrease in the number of rules: the rule extraction algorithm generated 39, 365, 149, and 0 rules for each of the four respective levels of INFL.

5. DISCUSSION OF RESULTS

In addition to being easily readable by subject-matter experts, the rules generated by the rule extraction algorithm can be executed directly by the computer. The advantages to this product are twofold: (1) The rules can be as validation of the AFFNN's learning ability, since the training and testing data can be classified independently by the rules; (2) the rules can be used to forecast or predict future outcomes when used with previously unseen exemplar vectors. The rules produced for the Divisia dataset generate outcomes that are very similar to the output of the AFFNNs, as expected. However, the rule extraction algorithm relaxes several constraints as it progresses, which permits the rules to process borderline data cases differently than the AFNNN would compute the same data under some circumstances. Identical AFFNN and rule outcomes are neither guaranteed nor expected in all cases.

Examination of the rules produced by both AFFNNs yields interesting observations. In general, rules tended to use all five attributes to describe INFL,

suggesting that all attributes have some significance or relative importance. For the NIBD attribute, the expression simplification step of the rule extraction algorithm commonly folded two or more levels of this component together (see Fig. 4). This indicates that the rule would be true for multiple levels of the NIBD attribute, suggesting that NIBD values play a key role in the prediction of INFL, but specific levels may not be critical contributors. This was also true in many cases for the BSD attribute. In contrast, the extraction algorithm rarely folded IBSD or IBTD together, allowing for the possibility that specific levels of these attributes may significantly impact INFL.

These results, which might initially seem to conflict with earlier sensitivity analysis findings by Gazely and Binner (2000), are not alarming. The AFFNN observations are based on the importance of particular levels of attributes in certain situations and equations, rather than weight sensitivities in the overall network.

The most surprising result of the rule generation process was observing the number of rules generated by both AFFNN models. Although modification of the encoding for the INFL target attribute did reduce the number of rules, there was still an unexpectedly large quantity of expressions for such a small dataset. This suggests one or more of the following conclusions regarding the nature of the dataset:

(1) the learning of appropriate relationships is hindered due to suboptimal data encoding or clustering practices,
(2) decompositional rule extraction is a computationally complex task that does not tend to generate optimal numbers of rules by default,
(3) the relationships among these data are extremely complex and cannot be adequately captured within the expressive power provided to the network models, or
(4) there is not a sufficient mechanism to accurately capture the relationships, and the network is merely learning a somewhat random set of relationships based on the data available.

Since techniques such as those used in Gazely and Binner (2000) have been able to successfully model the relationships within the Divisia component data, it is most likely that the first point (suboptimal encoding and/or data clustering) has merit. In addition, it is clearly true that rule extraction is a nontrivial process, lending support to the second conclusion as well. Since both the first and second AFFNN models classify the testing and training data with reasonable classification accuracy, there seems to be an existing relationship among the Divisia components, which casts doubt on the latter two conclusions. Regardless of the reason for the number of rules, this simple experiment demonstrates the potential complexity of attempting to describe relationships, even in small datasets.

Table 2. AFFNN vs. Tailored Network Models.

Network	Topology	INFL Encoding	Train/Test Accuracy in %
1st AFFNN	43–15–43	Thermometer	93.3/84.2
1st Tailored	39–6–4	Thermometer	97.3/78.9
2nd AFFNN	43–16–43	1-of-N	89.3/78.9
2nd Tailored	39–6–4	1-of-N	98.7/73.7

As a final basis of comparison for network training accuracy, two individual supervised feed-forward neural networks were created, each corresponding to the encoding method used by of one of the AFFNN models. For each network, the five encoded Divisia components were used as the network exemplar vectors (39 encoded elements), with the INFL attribute (4 encoded elements) as the target. These tailored 39–6–4 feedforward models were trained with the identical 80% training dataset records used with the AFFNN models and were tested with the identical 20% test dataset records. Both tailored networks used sigmoid activation functions in the hidden layer and linear activation functions in the output layer. After 4000 epochs (94 seconds, MSE = 0.004789, although MSE was stable near 1000 epochs) of training, the first tailored model correctly classified 97.3% of the training data and 78.9% of the test data. The second tailored model was similarly trained, and correctly classified 98.7 and 73.7% of the training and testing data, respectively. These results are shown in Table 2 along with the values of their respective AFFNN models.

As the table indicates, the accuracy of the AFFNN and corresponding tailored individual network was very close. In fact, each tailored model exhibits a higher training accuracy than the related AFFNN, and lower testing dataset accuracy than the related AFFNN. This suggests that the tailored networks might be suffering from some overfitting, while the AFFNN models are still able to generalize reasonably well on the test data. Good generalization in the face of potentially long training times is a strong argument in favor of using the AFFNN on the Divisia component dataset and similar monetary component asset data, especially when performing preliminary data mining investigations.

6. SUMMARY

This preliminary, yet very promising, research demonstrates how neural network models (such as the Aggregate Feedforward Neural Network) provide beneficial information in the domain of discovering and describing the money-price relationship using Divisia component data. The AFFNN is demonstrated as being

straightforward to design and use with encoded Divisia component and inflation data, and the model is able to effectively learn the relationships within the dataset. A simple decompositional rule extraction technique examines the learned knowledge and automatically generates a collection of *if-then* rules in terms of the original attribute values. These Divisia rules are suitable for examination by subject-matter experts (specifically, econometricians). The rules potentially provide interesting and useful insight into monetary aggregation theory, particularly when exploring the relationships between various monetary assets and the corresponding growth rate of prices. As an additional advantage, the resulting rules are expressed in well-documented computer code, capable of being executed for validation or used for forecasting purposes.

There are a number of enhancements and additional paths requiring attention as the research continues. It is almost painfully evident from the preliminary results that selection of encoding and clustering methods significance impacts the final solution. This is certainly not a new observation with respect to connectionist models. It merely underscores the importance of finding a reasonable representation before investing significant amounts of time or effort in network training. The use of appropriate and representative data encoding and preprocessing heuristics is critical with respect to producing reasonable and properly expressed relationships. Inappropriate encoding or data clustering can hinder the discovery of practical relationships. Only a selection of obvious preprocessing and encoding mechanisms were examined here, but others may be more appropriate. Preprocessing and encoding also impacts rule generation.

The final rule set is reasonably large, and should be reduced in both size and complexity. To promote confidence in the rule set, it should also be compared with results from a greater variety of systems. Perhaps the use of additional datasets would also be helpful in this regard, especially as we continue to examine and draw from other current research in monetary aggregation theory. The rule extraction algorithms, although functional, must also be improved to avoid issues in combinatorial explosion. This requires more intelligent and efficient extraction techniques and early simplification of the rule sets. Finally, the AFFNN excels when there are multiple relationships in the dataset, and tends to present weaker solutions when such relationships do not exist. Since it is unclear if there are multiple relationships within the Divisia component dataset, it would be interesting to disengage the rule generation capability from the AFFNN system and execute additional rule generation experiments with simpler feedforward connectionist models, such as with the tailored models reported herein.

The process outlined in this research clearly demonstrates the utility of connectionist models, particularly data-mining models such the Aggregate Feedforward Neural Network, in discovering the relationships among Divisia

component data. Ongoing research will examine alternative encoding mechanisms and extensions to the decompositional rule extraction algorithms. It will be especially interesting to more rigorously compare the upcoming results with contemporary techniques used in describing the money-price relationship as this revolutionary work in macroeconomics continues.

NOTES

1. Interest-bearing savings account with instant access to the money.
2. Component data is available on the Internet at http://www.bankofengland.co.uk/mfsd/index.htm (Bank of England Statistical Abstracts, Part 2, Section A, Tables 12.1 and 12.2).
3. Traditional feedforward neural networks would also be capable of learning such relationships; the chief advantage of the AFFNN is that it learns the other five sets of relationships, for NC, NIBD, IBSD, IBTD, and BSD, simultaneously within the same network. The functions for the Inflation attribute, represented by L_x in Eq. (2), are the only ones of immediate interest in this study.

REFERENCES

Barnett, W. A. (1978). The user cost of money. *Economic Letters*, *1*(2), 145–149. Reprinted in: W. A. Barnett & A. Serletis (Eds) (2000), *The Theory of Monetary Aggregation* (Chap. 1, pp. 6–10). Amsterdam: North-Holland.

Barnett, W. A. (1980). Economic monetary aggregates; an application of index number and aggregation theory. *Journal of Econometrics*, *14*(1), 11–48. Reprinted in: W. A. Barnett & A. Serletis (Eds) (2000), *The Theory of Monetary Aggregation* (Chap. 2, pp. 11–48). Amsterdam: North-Holland.

Barnett, W. A. (1995). Exact aggregation under risk. In: W. A. Barnett, M. Salles, H. Moulin & N. Schofield (Eds), *Social Choice, Welfare and Ethics* (pp. 353–374). Cambridge. Proceedings of the Eighth International Symposium in Economic Theory and Econometrics: Cambridge University Press. Reprinted in: W. A. Barnett & A. Serletis (Eds) (2000), *The Theory of Monetary Aggregation* (Chap. 10, pp. 195–216). Amsterdam: North-Holland.

Barnett, W. A., Liu, Y., & Jensen, M. (1997). The CAPM risk adjustment for exact aggregation over financial assets. *Macroeconomic Dynamics*, *1*, 485–512. Reprinted in: W. A. Barnett & A. Serletis (Eds) (2000), *The Theory of Monetary Aggregation* (Chap. 12, pp. 245–273). Amsterdam: North-Holland.

Barnett, W. A., & Serletis, A. (Eds) (2000). *The theory of monetary aggregation*. Amsterdam: North-Holland.

Belongia, M. T. (1996). Measurement matters: Recent results from monetary economics reexamined. *Journal of Political Economy*, *104*(5), 1065–1083.

Drake, L. M., Fleissig, A., & Mullineux, A. W. (1999). Are 'risky' assets substitutes for 'monetary' assets? Evidence from an aim demand system. *Economic Inquiry*, *37*, 510–526.

Elger, T., & Binner, J. M. (2004). The UK household demand for risky money. *Topics in Macroeconomics*, *4*(1). The Berkeley Electronics Press (Bepress). http://www.bepress.com/bejm.

Fisher, P., Hudson, S., & Pradhan, M. (1993). Divisia indices for money: an appraisal of theory and practice. Technical Report 9, Bank of England Working Paper Series.

Gazely, A. M., & Binner, J. M. (2000). Optimal weights for Divisia aggregation using a neural network approach. In: *5th Biennial Conference on Alternative Perspectives on Finance*. Dundee.

Hornick, K., Stinchcombe, M., & White, H. (1989). Multilayer feedforward networks are universal approximators. *Neural Networks*, *2*(5), 359–366.

Lu, H., Setiono, R., & Liu, H. (1995). NeuroRule: A connectionist approach to data mining. In: *Proceedings of the 21st VLDB Conference* (pp. 478–489). Zurich, Switzerland.

Mullineux, A. W. (Ed.) (1996). *Financial innovation, banking and monetary aggregates* (Chap. 1, pp. 1–12). Cheltenham, UK: Edward Elgar.

Schmidt, V. A. (2002, May). *An aggregate connectionist approach for discovering association rules*. Ph.D. thesis, Wright State University, Dayton, OH.

Schmidt, V. A., & Binner, J. M. (2003). Determining Divisia rules using the aggregate feedforward neural network. In: *Proceedings of the 2003 International Conference on Artificial Intelligence* (pp. 68–74). Las Vegas, NV.

Schmidt, V. A., & Chen, C. L. P. (2002a). Extracting rules from the aggregate feedforward neural network. In: *Proceedings of the 2002 International Conference on Artificial Intelligence* (pp. 188–193). Las Vegas, NV.

Schmidt, V. A., & Chen, C. L. P. (2002b). Using the aggregate feedforward neural network for rule extraction. *International Journal on Fuzzy Systems*, *4*(3), 795–807.

Towell, G. G., & Shavlik, J. W. (1993). Extracting refined rules from knowledge-based neural networks. *Machine Learning*, *13*, 71–101.

PREDICTING HOUSING VALUE: GENETIC ALGORITHM ATTRIBUTE SELECTION AND DEPENDENCE MODELLING UTILISING THE GAMMA TEST

Ian D. Wilson, Antonia J. Jones, David H. Jenkins and J. A. Ware

ABSTRACT

In this paper we show, by means of an example of its application to the problem of house price forecasting, an approach to attribute selection and dependence modelling utilising the Gamma Test (GT), a non-linear analysis algorithm that is described. The GT is employed in a two-stage process: first the GT drives a Genetic Algorithm (GA) to select a useful subset of features from a large dataset that we develop from eight economic statistical series of historical measures that may impact upon house price movement. Next we generate a predictive model utilising an Artificial Neural Network (ANN) trained to the Mean Squared Error (MSE) estimated by the GT, which accurately forecasts changes in the House Price Index (HPI). We present a background to the problem domain and demonstrate, based on results of this methodology, that the GT was of great utility in facilitating a GA based approach to extracting a

Applications of Artificial Intelligence in Finance and Economics
Advances in Econometrics, Volume 19, 243–275

ISSN: 0731-9053/doi:10.1016/S0731-9053(04)19010-5

sound predictive model from a large number of inputs in a data-point sparse real-world application.

1. INTRODUCTION

Development of data-derived models (for example, Artificial Neural Networks) of smooth systems, where the objective is to construct a model directly from a set of measurements of the system's behaviour is often problematic. This is especially true of systems where the relative utility of each metric is unclear, where a large number of potentially useful inputs exist and where data is either sparse or contains a high level of noise. These problems are often addressed iteratively, with data-driven models being constructed, analysed and adjusted before repeating the cycle until either a useful model is constructed or it becomes apparent that the system being modelled is not smooth. Clearly, this iterative procedure is labour intensive requiring considerable experience and skill on the part of the practitioner.

The Gamma Test (first noted by Stefánsson et al., 1997) (GT) was originally developed to facilitate the exercise of constructing data-derived models of smooth systems, i.e. a model where the transformation from input to output is continuous and has bounded first partial derivatives over the input space (e.g. Jones et al., 2002). The GT procedure provides an estimate for the noise level present in a data set computed directly from the data without assuming any a priori knowledge about the system. The GT provides a measure of the quality of the data that, in cases of high noise, indicates when a smooth model does not exist within the data. ANN derived models generalise from useful data first and reach a point where continuing the process tends to "over train" the network, a situation where noise is incorporated into the model. Therefore, providing a measure of the noise inherent within a data set before the modelling exercise begins supplies a point at which the training exercise should terminate. However, the utility of a data-derived measure of noise within a non-linear system extends further. That is, supplying an estimate of the level of noise within a data set provides a means for extracting useful features before the iterative procedure of deriving a data-driven model begins.

In this paper, the authors demonstrate, by means of a real-world example that is familiar to many, the practical utility of the GT to the area of attribute selection and dependence modelling. Specifically, the authors show how a GA driven by a GT derived objective function can heuristically generate a sample of different attribute selections that, when analysed, strongly indicates which attributes are salient. In addition, the authors show how these salient attributes were utilised to generate an ANN model of the underlying relationship that resulted in good actual forecasts

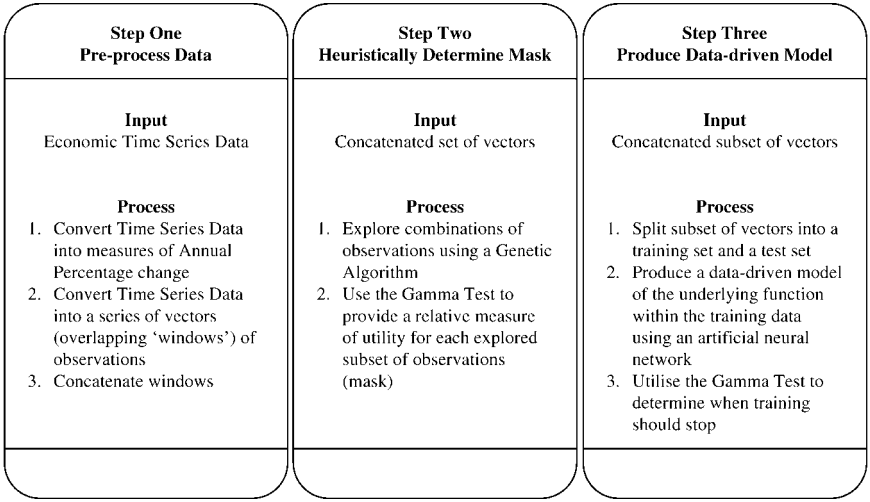


Fig. 1. Summary of the Modelling Procedure.

and trends that closely followed the actual. The salient steps within this process are illustrated in Fig. 1.

First, the reader is presented with an overview of the GT, GA and ANN procedures and the problem domain. Next, an explanation is given of how these procedures were applied to the domain problem to extract predictively useful features and associated forecasts. Finally, conclusions are drawn and plans for future work explained.

2. THE GAMMA TEST (GT): OVERVIEW

The Gamma Test is a non-linear data analysis algorithm that estimates that part of the variance of the output of an input/output data set

$$\{(\mathbf{x}_i, y_i) | 1 \leq i \leq M\} \tag{1}$$

that cannot be accounted for by the existence of any smooth model based on the inputs, even though the model is unknown. Here $\mathbf{x}_i = (x_1(i),\ldots, x_m(i))$ represents the ith data input vector and y_i represents the associated output.

We imagine that the data is derived from an underlying smooth function f: $P^m \rightarrow P$ which is *unknown* and that the measured output values y are given by

$$y = f(x) + r \tag{2}$$

where r is a random variable with mean zero and bounded variance $\mathrm{Var}(r)$.

The Gamma test estimates $\mathrm{Var}(r)$ in $O(M \log M)$ time by first constructing a *kd*-tree using the input vectors $\mathbf{x}_i$ $(1 \le i \le M)$ and then using the *kd*-tree to construct lists of the kth $(1 \le k \le p)$ nearest neighbours $\mathbf{x}_{\mathrm{N}[i,k]}$ $(1 \le i \le M)$ of $\mathbf{x}_i$. Here p is fixed and bounded, typically $p = 10$. The algorithm next computes

$$\delta_m(k) = \frac{1}{M}\sum_{i=1}^{M} |x_{N[i,k]} - x_i|^2 \quad (1 \le k \le p) \tag{3}$$

where $|\cdot|$ denotes Euclidean distance, and

$$\gamma_M(k) = \frac{1}{2M}\sum_{i=1}^{M} (y_{N[i,k]} - y_i)^2 \quad (1 \le k \le p). \tag{4}$$

Note here that $y_{\mathrm{N}[i,k]}$ is not necessarily the kth nearest neighbour of y_i in output space. Finally the regression line $\gamma = \Gamma + A\delta$ of the points $(\delta_M(k), \gamma_M(k))$ $(1 \le k \le p)$ is computed and the vertical intercept Γ returned as the estimate for $\mathrm{Var}(r)$. The slope parameter A is also returned as this normally contains useful information regarding the complexity of the unknown surface $y = f(\mathbf{x})$.

Evans and Jones (2002) and Evans et al. (2002) provide a formal proof that $\Gamma \to \mathrm{Var}(r)$ in probability as $M \to \infty$ under a wide range of circumstances. The idea is based on the remarkable observation that the relationship between $\gamma_M(k)$ and $\delta_M(k)$ is *approximately linear* in probability as M becomes large, i.e.

$$\gamma_M(k) \approx \mathrm{Var}(r) + A\delta_M(k) + o(\delta_M(k)) \tag{5}$$

with probability one as $M \to \infty$.

If linear regression is characterised as the ability to provide an estimate of "goodness of fit" against the class of linear models, then the Gamma test is *non-linear regression*, because it provides an estimate of "goodness of fit" against the class of non-linear smooth models that have bounded partial derivatives.

If possible the Gamma test requires the number of data points M to be relatively large, with an asymptotic estimate being reached for incrementally increasing quantities of M indicating sufficient data is available to provide a reliable Γ estimate. With high dimensional input data we may, of necessity, require orders of magnitude more data. However, this should not surprise us, it is intrinsic to the nature of the undertaking. A linear model is determined by very few parameters and naturally requires less data to fit, whereas here we seek to quantify the goodness of fit against a huge class of potential models, each of which may be determined by an infinite set of parameters. What is surprising is that this can be done at all.

Although the Gamma test gives very little information about the best fitting function from the allowed class it nevertheless *facilitates* the construction of such a model. To actually build the model we use information from the Gamma test combined with other non-parametric techniques, such as local linear regression or neural networks.

However, the Gamma test has other implications that are of relevance to the present study: it can be used for *model identification*. In this context we might say that the goal of model identification for a particular output is to choose a selection of input variables that best models the output y. Although *mathematically* the inclusion of an irrelevant variable in the list of inputs makes no difference to the fact that f: $P^m \to P$ is a *function*, nevertheless *in practice* it is very important to eliminate counter-productive inputs. This reduces training time for neural networks and can substantially improve the resulting model.

Some input variables may be irrelevant, or subject to high measurement error, so their inclusion as inputs into the model may be counter-productive, leading to a higher *effective* noise level on the desired output. Since a single Gamma test is a relatively fast procedure it is possible (provided m is not too large) to find that selection of inputs which minimises the (asymptotic) value of the Gamma statistic and thereby make the "best selection" of inputs. Moreover, for the purpose of comparing one selection of input variables with another, even if M is smaller than we would prefer, it may not be critical that individual Gamma test results are rather less accurate than one would like *provided* they are all computed using the same data. This is because we are primarily interested in ranking selections of inputs in order of their Gamma statistics rather than the Gamma statistic per se.

3. GENETIC ALGORITHM OVERVIEW

This section provides an introduction to the, general, GA search procedure highlighting the design methodology adopted. Genetic Algorithms are adaptive search methods that can be used to solve optimisation problems. They are based on the genetic process of evolution within biological organisms. Which is to say that, over many generations, populations have evolved according to the principles of natural selection. By adopting this process, a GA is able to "evolve" solutions to real world problems (for a fuller discussion the reader is directed to Golberg, 1989).

Solutions are evolved utilising a genome (or structure of the problem, where a single instance of which represents a solution to the problem) and a genetic algorithm (the procedure utilised to control how evolution takes place). The GA makes use of genome operators (associated with the genome)

and selection/replacement strategies (associated with the GA) to generate new individuals. The GA uses an objective function to determine how fit each of these individual genomes is for survival. So, given a choice of GA, three things are required to solve a problem:

- A representation for the genome (determined from the definition of the problem);
- Suitable genetic operators;
- An objective function that measures the relative quality of a solution.

In summary, when using a GA to solve an optimisation problem, a series of variables are combined to form a single solution to the problem within a single genome. The GA creates a population of solutions based on the genome. The GA then operates on this population to evolve an optimum, or near optimum, solution to the problem utilising the objective function. Given this overview, the following sections expand upon each of these components.

3.1. The Genome

This section outlines the decision making process that determines how an individual solution (the genome) should be modelled and physically represented. When defining a representation appropriate to the problem at hand, a data structure that is minimal but also completely expressive should be selected (see Section 7.3 for the description of the implemented genome). Although it may appear beneficial to include extra genetic material beyond that which is required to express a solution fully, this tends to increase the size of the search space and hinder the performance of the algorithm. Finally, each genome will have a "fitness" score associated with it that determines its prospects for selection. This representation is an independent component of the general GA procedure, allowing separate decision making processes to be made. For example, a different GA procedure might be adopted without any need to change the structure of the genome. This is possible because the operators necessary to evolve new solutions, described in the next section, are associated with the genome and not the GA itself.

3.2. The Genome Operators

Given a general GA procedure and genome (described in Section 3.1), it is also necessary to determine how operators specific to the genome should behave. This section describes how these operators act upon the genome within a general GA procedure. Three operators can be applied to the genome, these being initialisation,

mutation and crossover. These operators allow a population to be given a particular bias, and allow for mutations or crossovers specific to the problem representation. The initialisation operator determines how each genome is initialised. Here, the genome is "filled" with the genetic material from which all new solutions will evolve. Next, the mutation operator defines the procedure for mutating the genome. Mutation is only rarely applied and randomly alters a gene in a selected "child." It provides a small amount of random search that facilitates convergence at the global optimum. Finally, the crossover operator defines the procedure for generating a child from two parent genomes. The crossover operator produces new individuals as "offspring," which share some features taken from each "parent."

These operators are independent functions in themselves, specific to the structure of the genome, which may be altered in isolation to the other components described in these sections. For example, the crossover operator might be changed from a single point (adopted by the authors and explained in Section 8.3) to a two-point implementation without any need to adjust the other components.

3.3. Objective Functions and Fitness Scaling

This section describes how an objective-function and fitness-scaling fits into a general GA procedure. Genetic algorithms are often more attractive than gradient search methods because they do not require complicated differential equations or a smooth search space. The genetic algorithm needs only a single measure of how good an individual is compared with the other individuals. The objective function provides this, needing only a genome, or solution, and genome specific instructions for assigning and returning a measure of the solution's quality. The objective score is the raw value returned by the objective function. The fitness score is the possibly transformed objective score used by the genetic algorithm to determine the fitness of individuals for mating. Typically, the fitness score is obtained by a linear scaling of the raw objective scores. Given this, the objective function can be altered in isolation from the GA procedure and genome operators, and, once a representation for the problem has been decided upon, without any need to change the structure of the genome.

3.4. The Genetic Algorithm

Here, we present an overview of a general GA procedure, explaining how each phase fits into the evolutionary process. The GA procedure determines when the population is initialised, which individuals should survive, which should

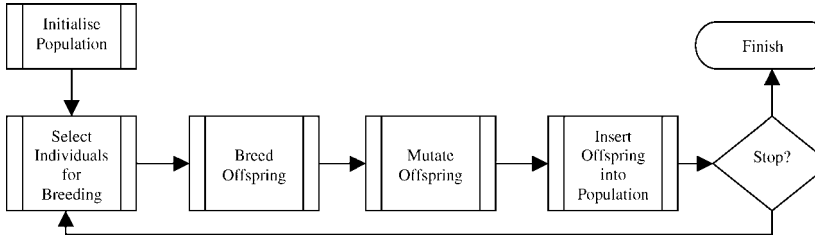


Fig. 2. Genetic Algorithm Evolution Procedure.

reproduce, and which should die. At each generation certain, highly fit, individuals (determined by the scaled, objective function) are allowed to reproduce (through selection) by "breeding" (using the crossover operator described in Section 8.3) with other individuals within the population. Offspring may then undergo mutation, which is to say that a small part of their genetic material is altered.

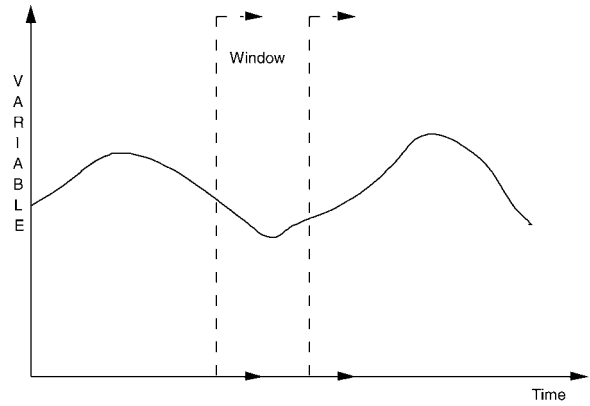
Offspring are then inserted into the population, in general replacing the worst members of the existing population although other strategies exist (e.g. random). Typically, evolution stops after a given number of generations, but fitness of best solution, population convergence, or any other problem specific criterion can be used. Given this overview of the general design methodology adopted, the following sections describe the domain problem and how the problem was modelled (Fig. 2).

4. FORECASTING WITH ANNS

The forward projection of a value that varies with time can be approximated from a model of the time series using a variety of traditional techniques. Most such schemes rely on the assumption of linearity, or on some kind of transformation of the data such as a logarithmic or a cosine. The necessary function has to be postulated however by the user for each individual time series.

Neural networks are pattern recognition devices that can be used effectively to fit any smooth non-linear function in data. They have the advantage that there is no need to assume any transformation or functional form in the data. Windowing is a technique often used to enable ANN's to model a time series. The network has multiple inputs, $x_1 \ldots x_m$ and an output y. The data is arranged in to $m + 1$ columns, each column being a repeat of the one on its right, but displaced down by one time unit. If the successive values of the time series are represented by y_t, y_{t+1}, $y_{t-2} \ldots$, then Table 1 shows the arrangement of the data, assuming for example that $m = 4$. The number of columns is decided by the length of the data set, and

Table 1. The Time Series is Organised as a Data Table, Each Record Being a Small Window of the Complete Time Series.



Record	Inputs				Output or Targets
1	y_{t+2}	y_{t+3}	y_{t+4}	y_{t+5}	y_{t+6}
2	y_{t+1}	y_{t+2}	y_{t+3}	y_{t+4}	y_{t+5}
3	y_t	y_{t+1}	y_{t+2}	y_{t+3}	y_{t+4}
4	y_{t-1}	y_t	y_{t+1}	y_{t+2}	y_{t+3}
5			*etc.*		

Note: The full table comprising of the time series back to a convenient point in time is used to train and validate the neural network. In this example, y_{t+6} is the most recent value of the time series.

the width of window considered necessary to include all the major features of the time series, a figure that is often found by trial and error.

Once a model has been trained and validated, it can be used to project the series forward in time. The most recent n values of the time series (up to y_{t+6} in this case) are applied to the neural network and the output gives the first projected value of the time series, y_{t+7}. Next, the last $n-1$ values of the series, plus the value y_{t+7} are applied to the neural network, which now gives y_{t+8}.

Often it is desirable to predict a time-series that is thought to be dependent on other time series. The input to the network is formed by concatenating the data from appropriate "windows" of the independent time series while the output is the predicted value for the dependent time series. However, mere concatenation often leads to lengthy (which is not a good feature when training neural networks) input vectors that may contain non-salient information. The aim of feature selection is to reduce the length of the training vector by removing non-salient columns from the concatenated vector.

5. THE PROBLEM DOMAIN: HOUSE PRICE FORMATION

Periodic overheating of residential property markets is a feature of developed economies. It is a feature that has the potential to inflict substantial socio-economic damage. Avoiding the worst effects has become a policy objective for governments' central banks (Aoki et al., 2001). Given that approximately 65% of U.K. residential property is mortgaged, the activity of lenders provides a potential brake to curb the worst excesses of the market. In particular it would be helpful if lending did not fuel speculation. At the moment, lenders rely on "open market value," a bid price which by definition reflects market froth, as the key metric when determining a ceiling figure for lending. A move away from current bid price to a metric that reflected a more sustainable value would realise the policy objective without heavy regulation. Such an alternative metric – that restrains the speculative component – would need to be forward looking. Whereas the "open market value" is determined by reference to comparable concurrent transactions, any measure of sustainable value would need to be predictive. This is not a trifling pursuit; the complexity is perhaps on a par with setting central bank lending rates. Incidentally, we might anticipate that the analysis identifies one or more heuristics similar to the Taylor rule that assists in setting the "appropriate" interest rate at the American Treasury Department. We call this more prudent metric "sustainable market value" and given that, historically, the highest risk in residential lending occurs during the first three years of a mortgage, we define "sustainable" using a three year horizon (Jenkins, 2002). The development of models that provide a sustainable valuation for properties would be of great usefulness to the lender and consumer. The measure would provide the lender with a more sober reflection of risk. The consumer would be able to make informed decisions based upon pricing models that give a clearer indication of the real, sustainable, value of property. "Sustainable market value" would forestall negative equity and facilitate movement between jobs in today's mobile labour market.

Two major problems exist when model building in this domain. The first problem is theoretical, the second computational. Economic theory is hardly a finished epistemological category. Writing of the "Death of Economics" in 1994 then forecaster at Henley Management College, Professor Paul Ormerod, suggested of current economic theory that it "should be abandoned or at least suspended until it can find a sounder economic base." And of economic forecasting in particular, "By ignoring non-linearity, forecasters constantly get things wrong – missing, for example, the contagion of fear that infected Asia and the world after the fall of the Thai baht in 1997" (Ormerod, 1998). What appears true of economic theory in general is also likely to be true of theoretical market models of residential markets in particular (Meen & Andrew, 1998).

The extent of the computational problem can be gauged from the fact that outside the subjective, comparison heuristic used by professional valuers, no coherent model exists even for the calculation of current bid price; the countless factors that influence value have nowhere been systematised (Jenkins et al., 1998). Furthermore, unless it can be proven that Takens' Theorem applies in this domain, the prediction of sustainable values is likely to be more rather than less complex than the calculation of open market values. This suggests that the kind of modelling strategy pursued here is relevant and may be significant.

6. DESCRIPTION OF THE DATA

Competing theories are used to explain the behaviour of markets at national, regional and urban aggregates. However, U.K. national and regional level models have developed primarily from the modelling of the market at the macroeconomic level and these models have not been integrated with modelling at the urban level, where professional valuers operate. In order to make some headway, we decided that the initial focus should be on the prediction of values at a national level where data has been systematically recorded for 30 years and sometimes longer.

In this paper we have not chosen data on the basis of a particular theory, rather data sets were chosen primarily because of their consistent use across the various models described in the literature, which we classify as follows:

- General, related, models (e.g. Ball & Grilli, 1997; Barkham & Geltner, 1996);
- Hedonic (a measure of general, overall opinion) regression analysis models (e.g. Adair et al., 1996; Antwi, 1995; Lam, 1996);
- Artificial intelligence (e.g. McCluskey & Anand, 1999; Wang, 1999), including ANNs (e.g. Lewis, 1999; McGreal et al., 1998; Vemuri & Rogers, 1994; Worzala et al., 1995).

Such models indicate that the main variables expected to influence house prices at both the national and regional levels include incomes; interest rates (real or nominal); the general level of prices; household wealth; demographic variables; the tax structure; financial liberalisation and the housing stock. Miles and Andrew (1997) developed a highly condensed forecasting model using just three variables widely thought to play a significant causal role together with a house price index:

- Real House Price (log of house price index divided by Retail Price Index – RPI);
- Real incomes (log of real disposable incomes at constant prices);

- Retail prices (log of RPI);
- Mortgage interest rate (tax-adjusted interest rate).

In this pilot we have included these last variables in the form of quarterly percentage changes in the Bank Rate (BR),[1] the Retail Price Index (RPI),[2] and the Average Earnings Index (AEI).[3]

However, given the data-driven nature of this model and given too that variables will be ranked in terms of their significance to model building; we were not constrained by the exigencies of economic theory. In addition to data that are thought to have a strong causal connection to house price changes, we were able to select additional time series that may have a weak (or no) connection with house price formation, or which may simply be associated with changing levels of activity/pricing in housing markets. In fact we restricted ourselves to quarterly percentage changes in the following additional variables, which a priori were assumed to be associated with if not causes of house price changes: claimant count (CC);[4] consumption of durable goods (DG); GDP household consumption (GDPHC); household savings rate (HHSR) and rates of mortgage equity withdrawal (MEW).

We also injected a degree of anonymity into the modelling by not disclosing to the non-economist model builder a priori expectations of the likely strength of these variables, nor the lagged observation(s) in which they might be expected to feature. The house price index data in this paper relates to quarterly changes in the All U.K.: Average House Price (£) (House Price Index – HPI) Nationwide Building Society (see Appendix A further information).

7. PROBLEM COMPOSITION AND MODELLING

In this section, considerations relating to the attribute generalisation procedure are described, and an overview of the model's underlying representation and physical implementation is provided, along with a detailed discussion about the objective function and its mathematical formulation.

7.1. Problem Size and Complexity

The eight attributes contained in the economic indicator database (quarterly measurements from 1975) that are converted into windows of length six and concatenated together provide a vector of 48 inputs and one output value. This configuration provides a search space of approximately 2.8 e+20 combinations.

7.2. Data Pre-Processing

Data is often manipulated using pre-processing routines, such as data scaling, sectioning, smoothing, and removal of outliers, before being used to construct an underlying model. However, the authors made the deliberate decision to allow the data-mining procedure described in this paper to drive the decision making process without any a priori assumptions about the data being made. Therefore, the data was simply transformed from a series of actual values into one containing the, quarterly, annual percentage movements of each series (see Appendix B for more information). This is important, given that the object was to reduce the windowed input vector for each independent time-series where possible to a single lagged observation. This decision was based upon an assumption that a combination of single, differently lagged, observations of the direction and magnitude of the input time-series could, in combination, be used to determine the change in the output time series. In other words, was it possible to replace a window of discrete observations with a single measure of the magnitude and direction of the time-series (i.e. the annual percentage movement)? To this end, each economic time-series was independently "windowised" (described in Section 4) and then concatenated together to provide a set of input vectors. Then, for each input vector was concatenated with a single value of the annual movement in the HPI, four quarters in advance of the end of the period covered by each input vector.

7.3. Genome Representation

Our state representation, the genome (introduced in Section 3.1), is physically stored as a string of Boolean values (a "mask") and its corresponding fitness value. The mask is effectively a series of on/off switches that correspond with the columns within our database (described in Section 7.2) of input vectors (illustrated in Fig. 3).

The fitness value for any given genome is calculated using only the active columns determined by its mask (e.g. the mask provided by the partial genome shown in Fig. 3 indicates that only columns $\{1, 3, 5, 7, 8, 9, 10, 13, \ldots, 48\}$ were utilised when determining its fitness score). This physical implementation is easily manipulated using the GA procedures introduced in Section 3 and explained further in Section 8.

7.4. Mask Evaluation

The success of any discrete optimisation problem rests upon its objective function, the purpose of which is to provide a measure for any given solution that represents

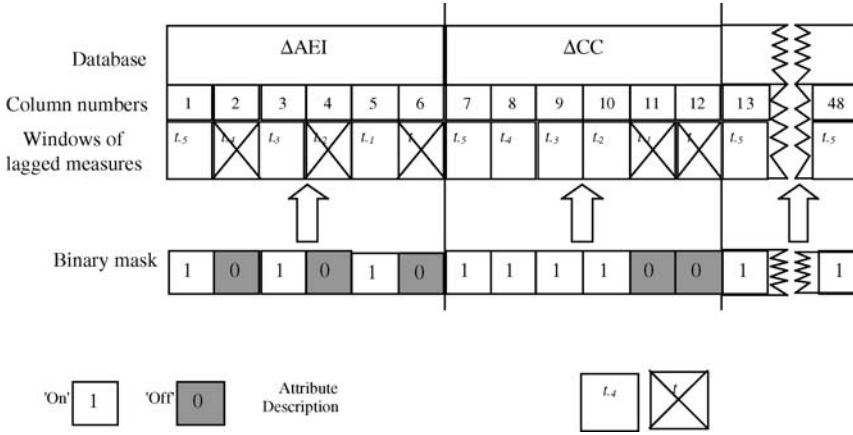


Fig. 3. Mapping within the State Representation.

its relative quality. In this section, we present the attribute selection strategies and their formulation within an objective function.

The objective function used here works by calculating the Gamma statistic for the data for a given attribute mask within our state representation and then summing metrics associated with the quality of the mask. Hence, the objective score associated with a given configuration is an abstraction of the penalties analogous with the relationships between a set of attributes determined by the configuration. In full, we consider the three measures of relative quality proposed by Durrant (2001), namely:

- the amount of noise within the database (should be reduced to a minimum);
- the underlying complexity of any underlying relationship between input and output data (should be minimised);
- the complexity of any ANN architecture utilised (should be optimally-minimal in terms of the number of inputs into the architecture).

7.4.1. Underlying Definitions and Constraints

The objective function utilised to evaluate solutions requires a number of definitions, namely:

- $P(t)$: $\{m_{t1}, \ldots, m_{tn}\}$ is the population of masks at generation t;
- M: $\{m_1, \ldots, m_s\}$ is the set, of length s, of all possible masks;
- t is the generation;
- n is the number of members in the population;
- s is the number of individual masks (2^a) that exist within the state-space;

- a is the total number of attribute measurements within a given mask;
- m_{ti} is a given mask i of the population at generation t;
- w_1 is the weight given to the intercept value (1.0);
- w_2 is the weight given to the gradient value (0.1);
- w_3 is the weight given to the number of active attributes (0.1);
- $(w_1 + w_2 + w_3) > 0$;
- $f_1(mask)$ is a function that returns the Gamma statistic for a given mask *mask* such that $0 \leq f_1(mask) \leq 1$;
- $f_2(mask)$ is a function that returns an estimate of the model's complexity for a given mask *mask* such that $f_2(mask) \geq 0$;
- $f_3(mask)$ is a function that returns the number of active attributes within a given mask *mask* such that $f_3(mask) \geq 0$;
- $f(mask)$ is a function that returns the weighted objective score for a given mask *mask* such that $f(mask) \geq 0$;
- $V_{\text{ratio}}(mask)$ is a function that returns Gamma/Var(output), providing a standardised measure;
- *OutputRange* is the difference between the maximum and minimum values in the output column;
- *Active(mask)* is a function that counts the number of active attributes within a given mask m;
- *Length(mask)* is a function that returns the number of attributes within a given mask m;
- *Gradient(mask)* is a function that returns the slope of the regression line used to calculate the Gamma statistic.

7.4.2. The Object Relationship Fitness Function

The objective function used to evaluate masks examines the weighted relationship between relative measures of its quality. The general expression of the objective function is:

$$f(mask) = 1 - (w_1 f_1(mask) + w_2 f_2(mask) + w_3 f_3(mask)). \tag{6}$$

Where $f_1(mask)$ and w_1 represent, respectively, the number of conflicting objects and the weight of that particular measure, with a high value of $f(mask)$ indicating a good solution. These values are then scaled to a positive range.

The first term of the objective function, $f_1(mask)$, returns a measure of the quality of the intercept based upon the V_{ratio}, which is a standardised measure of the Gamma statistic that enables a judgement to be made, independently of the output range, as to how well the output can be modelled by a smooth function. Minimising the intercept by examining different mask combinations provides a means for eliminating noisy attributes, facilitating the extraction of a smooth model

from the input data, and is expressed as:

$$f_1(mask) = \left\{ \begin{array}{ll} 1 - \dfrac{1}{1 - 10V_{\text{ratio}}(mask)} & \text{if } V_{\text{ratio}}(mask) < 0 \\ 2 - \dfrac{2}{1 + V_{\text{ratio}}(mask)} & \text{otherwise} \end{array} \right\}. \tag{7}$$

The second term of the objective function, f_2, returns a measure of the complexity of any underlying smooth model based on the gradient, or slope parameter, A determined by the GT. Minimising this complexity is desirable, especially in cases where data points are relatively sparse, and is expressed as where $|\cdot|$ denotes the absolute value:

$$f_2(mask) = 1 - \frac{1}{1 + \left| \frac{\text{Gradient}(mask)}{\text{Output Range}} \right|}. \tag{8}$$

Lastly, the third term of the objective function, f_3, sums the number of active elements within the input vector and returns this as a percentage of the total number elements within the input vector. Minimising the number of active genes within our genome encourages the search procedure to find solutions that are less complex, resulting in input minimal input vectors.

$$f_3 = \frac{\text{Active}(mask)}{\text{Length}(mask)} \tag{9}$$

The next section deals with our GA implementation, with special consideration being given to the sub-ordinate heuristics used to direct the procedure through the search space.

8. ATTRIBUTE SELECTION AND DEPENDENCE MODELLING

Of the variations of GA available, the work presented in this paper utilised an approach similar to the Simple GA (SGA). Our SGA uses overlapping populations but with a pre-specified amount of overlap (expressed here as a percentage), these being the initial and next generation populations. The SGA first creates a population of individuals by cloning the initial genome. Then, at each generation during evolution, the SGA creates a temporary population of individuals, adds these to the previous population and then removes the worst individuals in order that the current population is returned to its original size. This strategy means that the newly generated offspring may or may not remain within the new population, dependant

upon how they measure up against the existing members of the population. The following sections examine each component of our SGA implementation.

8.1. Configuration, Search Space and Cost Function

Given a set of lagged observations (w) for a number of economic metrics (n), a configuration s corresponds to an individual mask of Boolean values associated with each attribute. The search space S is therefore composed of all such configurations. According to Eq. (6), for each solution $s \in S$, $f(s)$ corresponds to a combination of minimal intercept, gradient and active attribute values.

8.2. Population Size and Maximum Generations

The population size was set to 100 for all experiments, however, larger population sizes would facilitate the attribute selection procedure at the expense of longer run times. The maximum number of generations varied between experiments with the search procedure terminating upon population convergence (defined as when the average fitness score stabilised).

8.3. Crossover, Replacement and Mutation

The probability of crossover (selection for sexual reproduction) determines the proportion of parents within the population that will be selected for crossover at each generation. The single-point crossover strategy (illustrated in Fig. 4) was adopted for all experiments.

Each time crossover occurs, two offspring were created using material from each of its parents. The results for all experiments presented in this paper were generated using a crossover percentage of 50%, which is to say that at each generation 50% of the new population were generated by splicing two parts of each genomes' parents together to generate two new genomes. The position of the join is determined randomly for each pair of parents.

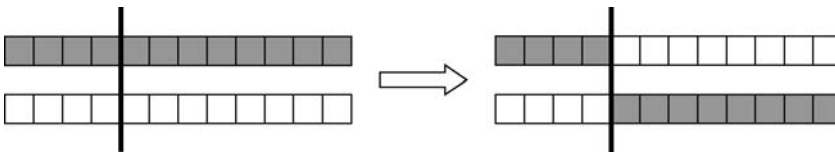


Fig. 4. Single Point Crossover.

Fig. 5. Single Bit Mutation.

Mutation is introduced to facilitate movement away from local optimality to a place closer to global optimality. However, mutation should not occur too often, as this would be detrimental to the search exercise. Consequently, the results presented here were generated using a 5% mutation probability, which was determined experimentally, utilising a single bit flip mutation operator (illustrated in Fig. 5).

9. ATTRIBUTE SELECTION COMPUTATIONAL EXPERIMENTS AND RESULTS

The experimental work that formed the basis for this paper utilised nine economic metrics dating from 1974 to 2001, which were pre-processed to provide measures of annual percentage movement for each of the series. The eight input metrics were converted into a set of vectors, of length six, which were concatenated together, along with a single corresponding output metric to provide 97 vectors. Experiments with the SGA produced the consolidated results presented below, where, for every member of the population, the number of times an attribute is active is totalled to provide an indication of its significance.

Totalling how often each lagged observation occurs within the population (presented graphically in Fig. 6) provides a useful means for pruning the input vector. In addition to this, totalling the number of times each lagged observation occurs within a each attribute provides a useful heuristic for determining the weight each attribute has on the outcome (illustrated in graphically in Fig. 7).

The algorithms and features referenced in this paper were implemented in VC++ 6.0 running under Windows NT on a Viglen P3 (800 MHz Pentium) with 128 Megabytes of RAM. Experimental data shows how certain lagged observations appear more often than others do. These results confirm a priori expectations to a significant degree. For example, it is noticeable that the recent behaviour of Average Earnings (see Fig. 6) is more important than in earlier periods as would be expected. The results suggest that the Bank Rate and Retail Price Index are consistently significant in house price formation. If this model provides a "useful" forecast of the house price index, it would confirm that their widespread inclusion in models as causal variables is rational.

Figure 7 ranks the variables by frequency of occurrence. The Bank Rate and Retail Price Index head the list as expected. Interestingly the Consumption of

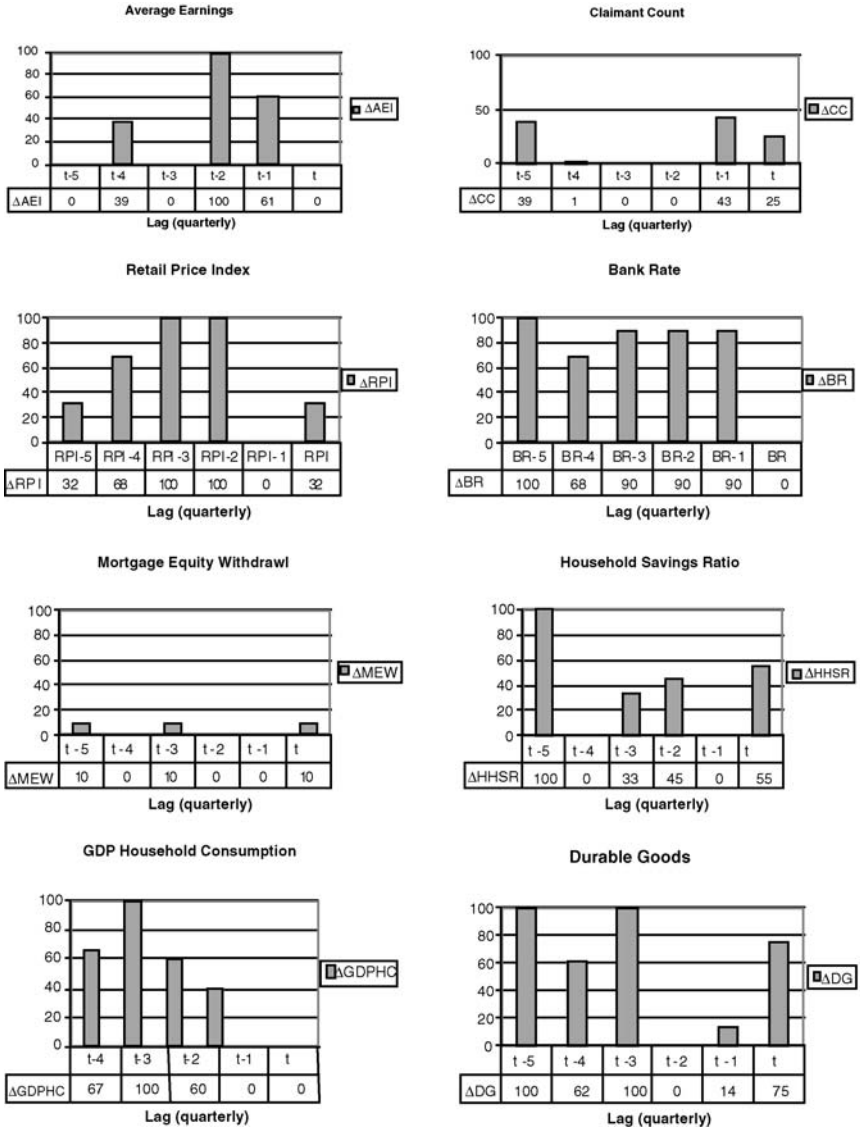


Fig. 6. Graphs Showing the Sum of How Often Each Lagged Observation Occurs in the Population.

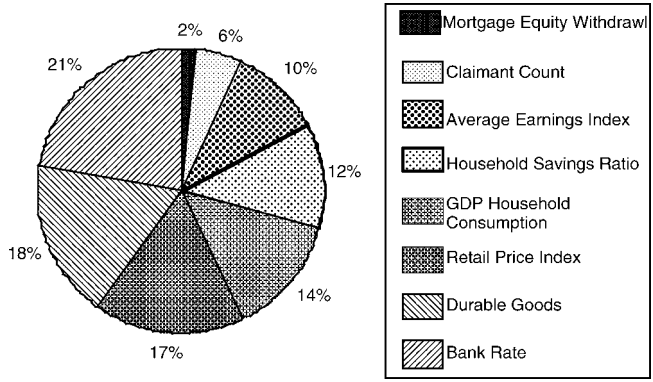


Fig. 7. Analysis of Relative Importance of Each Metric.

Durable Goods, not notable as a cause of house price changes per se, is also conspicuous. However, this is also expected. It is a feature of recent economic performance that the Consumption of Durable Goods and Average House Price are strongly correlated. At the other end of the scale it is observable that Mortgage Equity Withdrawal is a relatively weak input. While there are moments when this variable plays a role, across the whole time series it is expected that this would be of lesser significance.

Selecting those lagged observations that occurred in each member of the population (i.e. 100 times) significantly pruned the length of the original input vector (from 48 to 8). However, it was decided to heuristically reduce this to a single observation for each economic metric by again utilising the GT procedure. This, further, heuristic pruning of the suggested mask involved systematically eliminating each of the two ΔRPI and ΔDG metrics and measuring its affect on

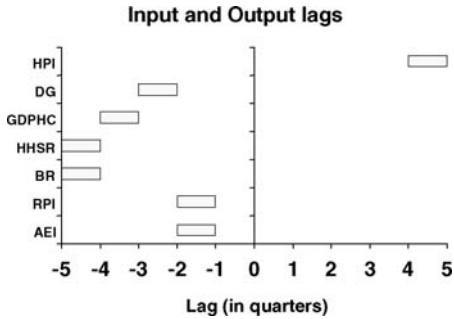


Fig. 8. Lagged Observations.

the Gamma Statistic and Gradient. Choosing the lagged value for the ΔRPI and ΔDG that least affected the Gamma and Gradient measures enabled a single most significant measure for each of these indices to be selected.

This procedure provided an input vector made up of varying lagged movements taken from each of the six predictively useful economic metrics (shown in Fig. 8) ready for the next, or predictive, step in the modelling exercise.

10. FORECASTING USING ARTIFICIAL NEURAL NETWORKS

Despite the many satisfactory characteristics of an ANN, building a neural network for a particular forecasting problem is a nontrivial task. Modelling issues that affect the performance of an ANN must be considered carefully. First, an appropriate architecture, that is, the number of layers, the number of nodes in each layer, and the number of arcs that interconnect with the nodes must be determined. Other network design decisions include the choice of activation function for the processing nodes, the training algorithm, data normalisation methods, training data, and performance measures (Zhang et al., 1998). In this section, an overview of the Back-Propagation ANN utilised to forecast a change in the HPI is provided.

10.1. The Network Architecture

Our ANN is composed of an input layer, which corresponds to the length of the input vector, an output layer, which provides the forecast values, and two layers of hidden nodes. Hornik (1991) has shown that a single hidden layer is sufficient for an ANN to approximate any complex non-linear function with any desired accuracy. However, recent findings have shown that two hidden layers can result in a more compact architecture that achieves a higher efficiency than single hidden layer networks (e.g. Chester, 1990; Srinivasan et al., 1994; Zhang, 1994).

10.1.1. The Number of Nodes in the Hidden Layers

It is important that the network has generalised across the time series and not simply fitted the inputs to their corresponding outputs. Therefore, the number of hidden nodes in each layer was determined by trial and error, with large numbers of nodes in the hidden layers being incrementally pruned to a minimum (of four nodes in each of the two hidden layers) whilst still producing relatively good forecasting capabilities.

10.1.2. The Number of Nodes in the Input Layer

The number of nodes in the input layer corresponds to the length of the *mask* generated during the attribute selection procedure described earlier. This is the most critical decision variable for a forecasting problem, since the vector contains important information about complex (linear and/or non-linear) structure in the data. Given that there is no widely accepted systematic way to determine the optimum length (or content) for the input vector the heuristically derived mask provides a significant step forward in this area of modelling.

10.1.3. The Number of Nodes in the Output Layer

For the time-series forecasting problem described in this paper, the single output node corresponds to the forecasting horizon. Here, a four-step-ahead (i.e. one-year into the future) was adopted.

10.2. Performance Measure

Although there can be many performance indicators associated with the construction of an ANN the decisive measure of performance is the prediction accuracy it can achieve beyond the training data. No one universally accepted measure of accuracy is available, with a number of different measures being frequently presented in literature (Makridakis et al., 1983). The performance measure adopted by the authors is the Mean Squared Error (MSE) function. The MSE provides an averaged measure of the difference between the actual (desired) and predicted value.

10.3. Updating the Weights

The new values for the network weights are calculated by multiplying the negative gradient with the learning rate parameter (set at 0.25) and adding the resultant vector to the vector of network weights attached to the current layer. In order to accelerate convergence a weighted momentum term (of 0.1) is added to the weight update.

10.4. Terminating the Training Procedure

As over-fitting is a widely accepted problem associated with modelling utilising ANNs, the GT's ability to accurately measure the "noise" within a data-set

and, consequently, the point at which training should stop provides a significant utility for practitioners. Over-fitting occurs because the ANN will attempt to fit all data encountered, including any noise present. Given that an ANN will tend to fit useful data before any noise (assuming that some underlying useful function exists within the data), providing a measure of any noise present in the data-set is of considerable utility as it allows training to end at a near optimal point. Therefore, the GT procedure (see Section 2, which calculates the variance of the noise) was applied to the input/output mask suggested by the heuristic procedure (see Section 8) to provide a MSE value at which training was stopped.

10.5. Partitioning of the Vector Set

Typically, training and test data sets are used during the ANN creation process. In this paper, the training set was used to construct the ANN's underlying model of the time series and the test set was used to measure the accuracy of this model. Next, the set of vectors was partitioned into a training set and a test set. The M-competition convention of retaining the last eight quarterly points for testing, mapped to input/output vectors, was adopted (Foster et al., 1992). The remaining vectors formed the training set. As the Gamma test statistic was used to stop the training procedure, there was no need to further partition the training set to provide a Validation Set (e.g. Wilson et al., 2002).

11. FORECASTING UTILISING AN ANN – EXPERIMENTAL RESULTS

In this section, the authors present the predicted and actual annual movements in the HPI in addition to the actual and predicted values for the HPI taking into account the predicted percentage change produced by ANN models that utilise the partial mask suggested by the method and the full mask for purposes of comparison. The salient decisions relating to GT, GA and ANN configuration and parameter settings utilised are summarised in Table 2.

11.1. Annual Percentage Movement Results

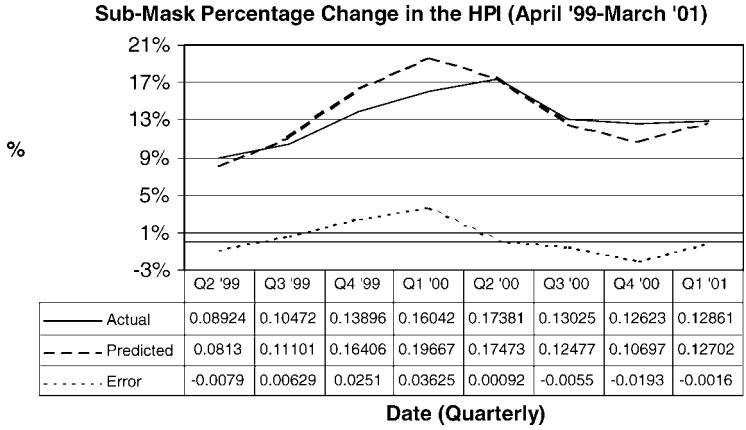
Experimental work utilising the training set of ninety, varying lagged, six indicator input-vectors to produce an ANN model that, when tested using the last eight

Table 2. Summary of Parameter Settings and Algorithm Configuration.

Gamma test objective function		
Intercept value (w_1)	1.0	
Gradient value (w_2)	0.1	
Number of active attributes (w_3)	0.1	
Number of near neighbours	10	
Genetic algorithm		
Type	Steady state	
Objective function	$f(mask) = 1-(w_1 f_1(mask) + w_2 f_2(mask) + w_3 f_3(mask))$	
Representation scheme	$mask$ = a binary string with length that corresponds with the total number of economic observations within each input vector	
Objective score scaling	Linear	
Genetic operators	Selection	Roulette wheel
	Crossover	Single point
	Mutation	Single bit inversion
	Replacement	Replace worse
Parameters	Crossover rate	50%
	Mutation rate	5%
Termination criteria	Population convergence – asymptotic average fitness	
Artificial neural network		
Type	Back-propagation	
Number of inputs	Determined by GA/GT	6
	All attribute observations	48 (8 windows of length 6)
Number of hidden layers	2	
Number of hidden nodes	4 per hidden layer	
Number of outputs	1	
Learning rate	0.25	
Momentum	0.10	
Performance measure	Mean squared error (MSE)	
Termination criteria	Determined by the Gamma Intercept ~ 0.004	

quarters of data, produced the results presented in Fig. 9. The model resulted in a MSE of 0.0030647 and a trend line that closely followed the actual in all but one of the test vectors, i.e. the direction of the change was detected in seven of the eight test quarters.

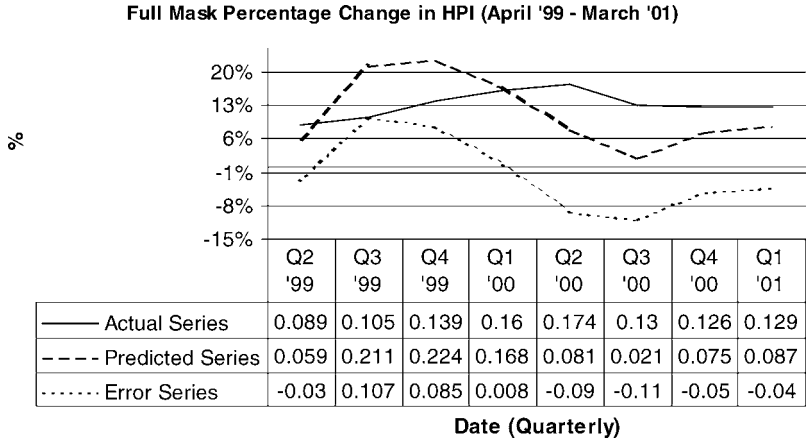
These results provide evidence that a predictively useful model has been extracted from the available feature data. However, the sparse nature of the data



	Q2 '99	Q3 '99	Q4 '99	Q1 '00	Q2 '00	Q3 '00	Q4 '00	Q1 '01
Actual	0.08924	0.10472	0.13896	0.16042	0.17381	0.13025	0.12623	0.12861
Predicted	0.0813	0.11101	0.16406	0.19667	0.17473	0.12477	0.10697	0.12702
Error	-0.0079	0.00629	0.0251	0.03625	0.00092	-0.0055	-0.0193	-0.0016

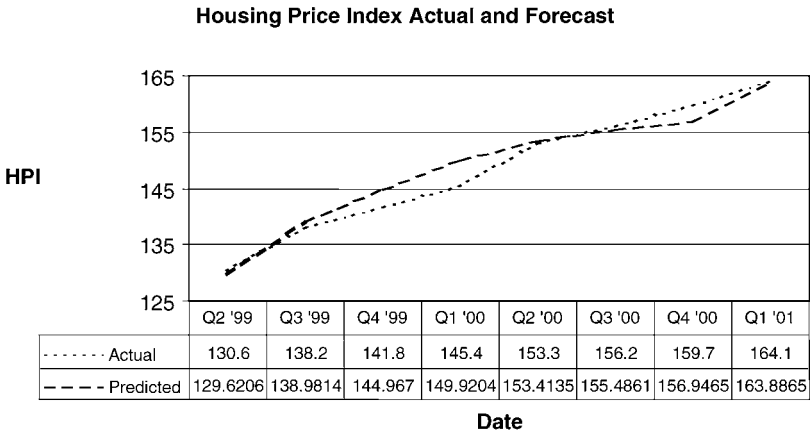
Fig. 9. Annual Percentage Movement Forecast Results using Predictively Useful Mask.

warrants further examination as more observations become available. Although direct comparison between ANN models constructed using different numbers of inputs and hidden nodes is problematic, the results generated using all the available input observations are illustrated in Fig. 10. This model resulted in a MSE of 0.0055438, but with considerable variance in the accuracy of the results, and a trend line that indicated the direction of the change in four of the eight test quarters.



	Q2 '99	Q3 '99	Q4 '99	Q1 '00	Q2 '00	Q3 '00	Q4 '00	Q1 '01
Actual Series	0.089	0.105	0.139	0.16	0.174	0.13	0.126	0.129
Predicted Series	0.059	0.211	0.224	0.168	0.081	0.021	0.075	0.087
Error Series	-0.03	0.107	0.085	0.008	-0.09	-0.11	-0.05	-0.04

Fig. 10. Annual Percentage Movement in Forecast Results Using Full Mask.



	Q2 '99	Q3 '99	Q4 '99	Q1 '00	Q2 '00	Q3 '00	Q4 '00	Q1 '01
Actual	130.6	138.2	141.8	145.4	153.3	156.2	159.7	164.1
Predicted	129.6206	138.9814	144.967	149.9204	153.4135	155.4861	156.9465	163.8865

Fig. 11. Housing Price Index Forecast.

11.2. Actual Movements in the House Price Index

Factoring the predicted percentage movement into the actual HPI resulted in forecasts within 3% of the actual and an average accuracy within 1.1% (shown in Fig. 11). The model built using the training data resulted in a standard deviation of the predictive error (0.066901), which is approximately 7% of the range.

12. CONCLUSION

This work has provided evidence that promising forecasting models can be produced using an ANN trained to a MSE suggested by the GT. In addition, it has been shown how predictively useful indicators can be heuristically selected from a database of economic metrics utilising a GT/GA procedure. The full-mask and partial-mask comparative results provided draw attention to the inherent problem of utilising ANNs to model functions with excessively large numbers of inputs and serve to highlight the importance of reducing the inputs (in terms of model efficacy) to a sub-set of salient features. As such, the quality (in terms of model accuracy) of the full-mask results would be expected to be poor given the data sparse (in terms of numbers of observations) problem domain. The quality of the model generated utilising the subset of inputs suggested by the methodology described in this paper supports the proposition that the GT has utility within the domain of attribute selection and dependence modelling.

13. FUTURE WORK

While care was taken in the selection of inputs for this pilot study, there may well be other useful indicators that could improve the performance of the present model. We have already identified a number of other time series that may further improve the usefulness of the forecasts beyond the national aggregate market to specific sub-markets and further work on these issues is planned. In addition, given the data sparse (in terms of numbers of observations), the robustness of the model will need to be tested as further data points become available.

Despite epistemological caveats, in general analysis such as this can be useful in refuting or confirming conventional wisdom regarding the relative importance of useful predictive variables. For example, there are competing theories regarding house-price formation that it might also be possible to test. One school of thought suggests that *land prices* are an important determinant of house price (at least in certain sub-markets). It will be relatively straightforward to run models with and without the variables suggested by these commentators. We also intend to test the models at regional and then urban aggregates where data is available.

NOTES

1. Description: TABLE 20.1: Bank of England Money Market Intervention Rates: Changes in Bank Rate, Minimum Lending Rate, Minimum Band 1 Dealing Rate and Repo Rate Source: Bank of England.

2. Description: CZBH: Percentage change over 12 months (headline rate), Annual 1949–1999, Monthly 1948 06 to 2000 (updated approximately monthly), Quarterly 1948 Q3 to 2000 (updated approximately quarterly). Source: Office for National Statistics.

3. Description: LNMU: Average earnings, Percentage change over 12 months, seasonally adjusted Monthly 1964 01 to 2000 (updated approximately monthly). Source: Office for National Statistics.

4. Description: BCJE: Claimant count, 1950–2000. Estimates of claimant count (the number of people claiming unemployment related benefits) in the U.K.; the level (thousands) and as a percentage rate. Source: Office for National Statistics.

REFERENCES

Adair, A. S., Berry, J. N., & McGreal, W. S. (1996). Hedonic modelling, housing sub-markets and residential valuation. *Journal of Property Research*, *13*, 67–83.

Antwi, A. (1995). Multiple regression in property analysis. *Estates Gazette Interactive*. Available: http://www.egi.co.uk/.

Aoki, K., Proudman, J., & Vlieghe, G. (2001). Why house prices matter. Bank of England Working Paper.

Ball, M., & Grilli, M. (1997). U.K. commercial property investment: Time-series characteristics and modelling strategies. *Journal of Property Research*, *14*(4), 279–296.

Barkham, R. J., & Geltner, D. M. (1996). Price discovery and efficiency in the U.K. housing market. *Journal of Housing Economics*, *5*(1), 41–63.

Chester, D. L. (1990). Why two hidden layers are better than one? In: *Proceedings International Joint Conference on Neural Networks* (Vol. 1, pp. 265–268). IJCNN-90-WASH-DC.

Durrant, P. J. (2001). *winGammaTM: A non-linear data analysis and modelling tool for the investigation of non-linear and chaotic systems with applied techniques for a flood prediction system*. Ph.D. dissertation, Dept. Comp. Science, Cardiff University, Wales, U.K.

Evans, D., & Jones, A. J. (2002). A proof of the Gamma test. *Proceedings Royal Society* (Series A, 458(2027), pp. 2759–2799). London.

Evans, D., Jones, A. J., & Schmidt, W. M. (2002). Asymptotic moments of near neighbour distance distributions. *Proceedings Royal Society* (Series A, 458(2028), pp. 2839–2849). London.

Foster, W. R., Collopy, F., & Ungar, L. H. (1992). Neural network forecasting of short, noisy time series. *Computers and Chemical Engineering*, *16*(4), 293–297.

Golberg, D. A. (1989). *Genetic algorithms in search, optimisation and machine learning*. Addison-Wesley.

Hornik, K. (1991). Approximation capabilities of multi-layer feed-forward networks. *Neural Networks*, *4*, 251–257.

Jenkins, D. H. (2002). Residential valuation theory and practice. *Estates Gazette*.

Jenkins, D. H., Lewis, O. M., Almond, N. I., Gronow, S. A., & Ware, J. A. (1998). Towards an intelligent residential appraisal model. *Journal of Property Research*, *16*(1), 67–90.

Jones, A. J., Evans, D., Margetts, S., & Durrant, P. J. (2002). *Heuristic and optimisation for knowledge discovery*. Idea Publishing, Hershey, PA (Chap. 9).

Lam, E. T.-K. (1996). Modern regression models and neural networks for residential property valuation. In: *Royal Institute of Chartered Surveyors*. The Cutting Edge.

Lewis, O. (1999). *The use of artificial intelligence techniques to assist in the valuation of residential properties*. Ph.D. dissertation, Dept. Math. and Comp., University of Glamorgan, Pontypridd, Wales, U.K.

Makridakis, S., Wheelwright, S. C., & McGee, V. E. (1983). *Forecasting: Methods and applications* (2nd ed.). New York: Wiley.

McCluskey, W., & Anand, S. (1999). The application of intelligent hybrid techniques for the mass appraisal of residential properties. *Journal of Property Investment and Finance*, *17*(3), 218–238.

McGreal, S., Adair, A., McBurney, D., & Patterson, D. (1998). Neural networks: The predication of residential values. *Journal of Property Valuation and Investment*, *16*(1), 57–70.

Meen, G., & Andrew, M. (1998). *Modelling regional house prices: A review of the literature*. University of Reading for the Department of the Environment, Transport and the Regions, ISBN 0 7049 1305 4.

Miles, D., & Andrew, S. (1997). *Merrill Lynch model of the U.K. housing market*. Financials Research, Merrill Lynch, London.

Ormerod, P. (1998). *Butterfly economics: A new general theory of social and economic behaviour*. Pantheon Books; ASIN: 0375407650.

Srinivasan, D., Liew, A. C., & Chang, C. S. (1994). A neural network short-term load forecaster. *Electric Power Systems Research*, *28*, 227–234.

Stefánsson, A., Koncar, N., & Jones, A. J. (1997). A note on the Gamma test. *Neural Computing and Applications*, *5*, 131–133.

Vemuri, V. R., & Rogers, R. D. (1994). *Artificial neural networks forecasting time series*. California: IEEE Computer Society Press.

Wang, S. (1999). An adaptive approach to market development forecasting. *Neural Computing and Applications*, *8*(1), 3–8.

Wilson, I. D., Paris, S. D., Ware, J. A., & Jenkins, D. H. (2002). Residential property price time series forecasting with neural networks. *Knowledge-Based Systems*, *15*(5–6), 335–341.

Worzala, E., Lenk, M., & Silva, A. (1995). An exploration of neural networks and its application to real estate valuation. *The Journal of Real Estate Research*, *10*(2), 185–201.

Zhang, G., Patuwo, B. E., & Hu, M. Y. (1998). Forecasting with artificial neural networks: The state of the art. *International Journal of Forecasting*, *14*(1), 35–62.

Zhang, X. (1994). Time series analysis and prediction by neural networks. *Optimization Methods and Software*, *4*, 151–170.

APPENDIX A: GRAPHICS OF THE SOURCE DATA

Figures A.1 and A.2

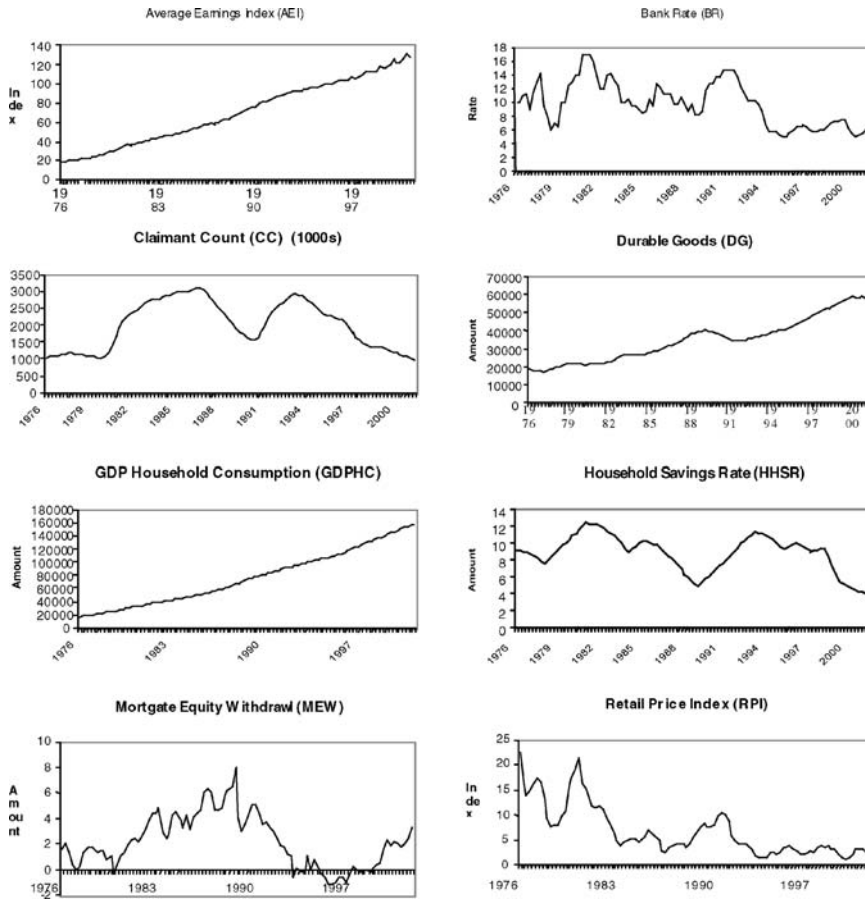


Fig. A.1. Graphs of the Real Movement within Each Economic Input Series.

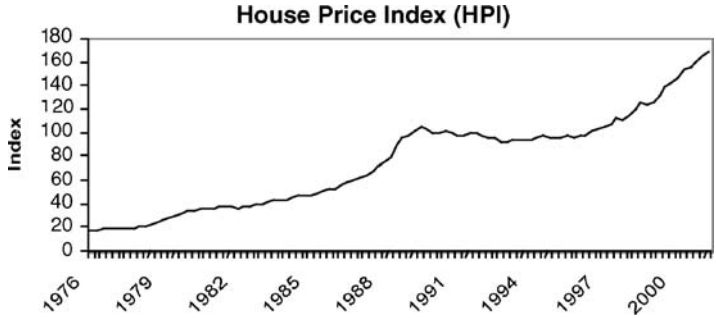


Fig. A.2. Graph Showing the Percentage Movement in the House Price Index.

APPENDIX B: GRAPHICS OF THE SOURCE DATA'S ANNUAL PERCENTAGE MOVEMENT

Figures B.1 and B.2

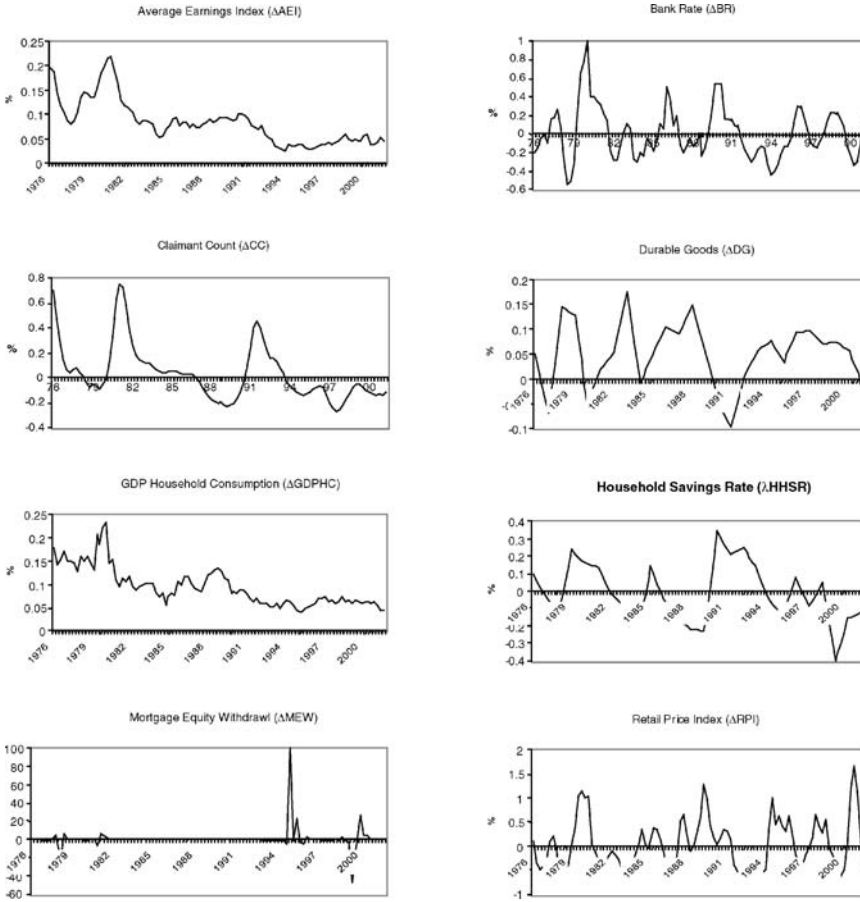


Fig. B.1. Graphs of the Annual Percentage Movement Within Each Economic Input Series.

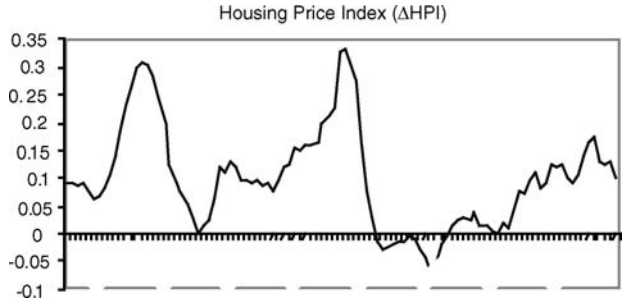


Fig. B.2. Graph Showing the Percentage Movement in the House Price Index.