ASTA
User's Reference Guide

Thomas Hellström

UMINF 00.16
ISSN-0348-0542

Department of Computing Science
Umeå University
S-901 87 Umeå, Sweden
email: thomash@cs.umu.se

August 28, 2000

Abstract

This document is an introduction and reference guide to **ASTA**, an Artificial Stock Trading Agent written in the Matlab language. The primary purpose of the **ASTA** system is to provide an easy-to-use environment for evaluation and development of multi-stock trading algorithms. All pre defined functions in the system are described with examples showing how to use them for creation of complex trading rules and fixed-horizon predictions. The **ASTA** system has been applied successfully to historical stock data, and results covering 11 years of the Swedish stock market are presented. A few applications with American stock data can also be found in the report.

Contents

1 Introduction

The idea of expressing stock prediction algorithms in the form of trading rules has gained considerable attention in academic research in the last years. The international conference NNCM-96 devoted a whole section in the proceedings to "Decision Technologies." Bengio [2] writes about the importance of training artificial neural networks with a financial criterion, rather than a prediction criterion. Moody and Wu [8] use reinforcement learning to train a trading system with objective functions, such as profit, economic utility, and Sharpe ratio. Atiya [1] describes a trading system based on time-variable, stop-losses, and profit objectives.

This report describes the system ASTA, which is an implementation of an Artificial Stock Trading Agent. Other information sources are: [5], [6], [7], and [4].

With ASTA, trading-rule-based prediction algorithms are easily evaluated, using historical data. The trading simulation extends the single-stock-prediction problem to a multi-stock problem. In this way, the prediction problem becomes focused on relative performance rather than the increase or decrease of individual stocks. Besides being an evaluation tool for existing algorithms, the system is a high-level development tool, where trading rules can be developed, combined, and tuned.

The program is written in the Matlab programming language, and is used either as an ordinary objective function called from a user's program, or as a Windows-based tool for making benchmarks, and developing trading algorithms. ASTA performs a simulation of multi-stock trading, where trading rules are executed for a large number of available stocks every day in the simulation period. The development of ASTA was instigated by a need for good working tools for the following research tasks:

1. A test bench for trading algorithms

   Many technical indicators for stock prediction are accepted and widely used, without ever having been subjected to an objective scientific analysis with historical stock data. It is true that many commercial software packages for technical analysis offer both a comprehensive programming language, and a simulation mode, where the performance can be computed. However, most available products do not take this task very seriously, and real trading simulation with a multi-stock portfolio is seldom possible. Furthermore, often the performance measures are not sufficient for a serious evaluation of the behavior of an algorithm for a longer period of time. Therefore, there is a need for a scientific test bench for the methods and algorithms already developed and in common use.

   The need for proper evaluation of new trading algorithms is of course the same as for existing ones. ASTA is developed in Matlab and therefore is suitable for tests of algorithms developed in the same language, but Matlab can also communicate with other languages.

2. An interactive development tool for trading rules

   There are reasons to believe that a successful trading system consists of many disjunct parts, where a buy signal can be for example, "screened" by looking at the traded volume. A buy signal issued with a low traded volume may be

then rejected. Other composite rules include looking at the general trend of the stock before accepting a signal from the system. ASTA provides the possibility to test such composite rules easily. The included function library and the possibility to define a trading strategy interactively, make implementation and evaluation of many trading strategies possible "without programming."

3. A non-interactive development tool for trading rules

   Furthermore, it is possible and maybe also fruitful, to automate the development of trading rules. Since ASTA defines the trading strategy as symbolic Buy and Sell rules given as arguments to the system, it would be perfectly possible for example, to construct Buy and Sell rules in a genetic framework.

   Even if the general look of the algorithm is fixed, there are often a lot of tunable parameters that affect the trading performance. Examples are filter coefficients, order of polynomials and levels above or below which an entity should pass to generate a trading signal. Since we believe that the actual behavior during a realistic trading situation is essential for proper selection and optimization of an algorithm, there is a need for an objective function that can be included in an optimization phase for parameter tuning.

4. A data-generating tool for post processing

   The comprehensive and user-friendly macro language in ASTA makes it a very suitable tool for extracting data for further analysis, such as classification with neural networks or fuzzy rule bases. A raw selection of trading situations is first set up with simple trading rules. Data ("features") for these situations is then automatically written to a file. A "target" value for each trade is also supplied. Thereafter it is a classification task to find out how to distinguish between a profitable trade and a non-profitable one, based on the given features.

2 Design Principles for ASTA

In this section we discuss the design and implementation of the ASTA system. The task of the Artificial Trader is to act on an artificial market with a large number of available stocks that vary in prices over time. The Trader has to execute the trading rule $T(t)$ at every time step, and decide whether to buy or sell stocks. The sole aim of the Trader is to produce as high a profit as possible. Various aspects of the calculation of the performance are discussed in the Section 2.1. The presentation and evaluation of the trading results are a major part of the ASTA system.

2.1 Performance Evaluation

The result of the artificial trader is presented as annual profits together with the increase in index. The mean difference between these two figures constitutes the net performance for the trader. The performance is displayed in both tabular and graphical formats as shown in the table part of Figure 3 and in Figure 4. The second last row of the table presents the annual profit above the index. The mean value of this entity is presented in the second rightmost column and represents a 1-figure

performance measure. However, the individual figures for each year should also be considered.

We now turn to the general architecture of the developed system.

2.2 Basic Architecture

The architecture of ASTA is based on an object-oriented approach with two major objects: the Market and the Trader. The two objects have a number of attributes and operations that can be applied on the objects. The basic components and their relations are presented in Figure 1.

2.2.1 The Market Object

The Market Object essentially consists of the total number of stocks that should participate in the trading simulation. A stock is defined by 4 time series; Close, High, Low, and Volume. Each of these time series has a numeric value (or NaN in the case of a unavailable value) for each date in the time period of interest. The basic operation on the Market Object is the simulation of changing prices as the date moves from start date to end date. The attribute T is updated by the Step-in-time operation applied to the Market Object.

2.2.2 The Trader Object

The Trader Object is more complex than the Market Object, as far as both attributes and allowed operations are concerned. The Portfolio attribute keeps track of the possession of stocks during the simulation. The Cash attribute is initialized to a certain value and thereafter is modified automatically as stocks are bought and sold. The Buy rule and Sell rule are the main attributes that affect the behavior of the Trader. They are expressed in a high-level language and may include calls to a large number of predefined Matlab functions that access the stock data in the Market Object. User-defined functions can also be called directly. The values of the Buy rule and Sell rule attributes are set interactively to enable fast experimentation when developing trading algorithms.

The basic operations on the Trader Object are Buy_recommendations that evaluates the Buy rule and Sell_recommendations that evaluates the Sell rule. The result is vectors with buy and sell recommendations for all relevant stocks. These recommendations are then carried out by the Buy and Sell operations. The behavior of the Trader is modified by a number of other parameters, e.g. minimum and maximum values for one individual trade.

2.2.3 Other Parts of the System

The Market and Trader Objects have to be controlled by a support system that takes care of the following "meta" operations:

- Simulation.

  The Step-in-time operation has to be applied to the Market Object in a loop for the selected time period. For each time step, the Trader Object should also be activated. The following pseudo code describes the full ASTA system:

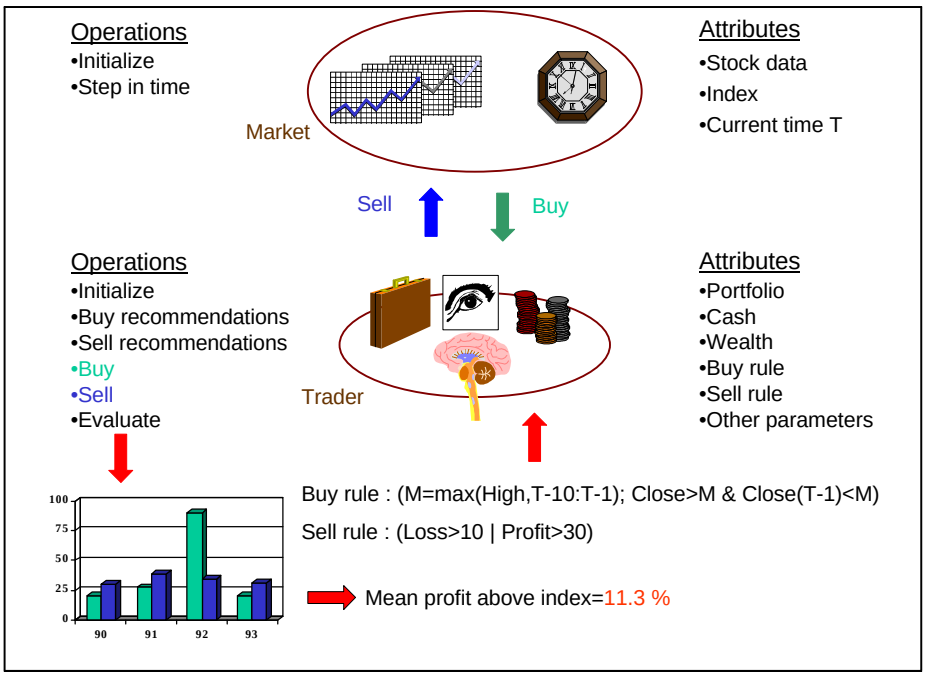


Figure 1: Basic Components of the ASTA System

Trader.Initialize
Market.Initialize
loop until **Market.T**≥EndDate
  s = Trader.Sell_Recommendations
  Trader.Sell(s) % Sell all stocks of type **s**
  s = Trader.Buy_Recommendations
  n = T.available_cash / length(s)
  Trader.Buy(s, n) % Buy **n** stocks of type **s**
 Market.Step in time
end loop
Trader.Evalute

- User interface.

  The end user assigns values to parameters such as the Buy rule and Sell rule of the Trader and the chosen time period for simulation. After the simulation the computed performance of suggested trades is presented and optionally printed to different media.

  The "silent mode" makes it possible to use the system as an objective function in a parameter optimization. The Artificial Trader is then called as a standard Matlab function from the optimization program code, returning a function value for each set of input parameters.

2.2.4 Predefined ASTA Functions

ASTA has a large number of predefined functions that make it possible to express compound trading rules interactively (as Buy rules and Sell rules). They also provide the developer of new algorithms with basic database access functions, as

well as some useful high-level functions. A complete description of the predefined functions in ASTA can be found in Section 5.

2.3 Other Design Issues

We conclude the general description of the design with some specific issues that must be considered in the design of an Artificial Trader.

2.3.1 When Is Data Available?

In a real trading situation on day T, the Close, High, and Low for day T are not available. However, it is common that prediction algorithms assume that this data is available, and can be included in the prediction of the following day's prices. Furthermore, it is often assumed that the Buy and Sell recommendations from the trading system can be executed using the close prices for day T. Of course, in reality this is hard to achieve, since the stock market is closed by the time the data for the current day becomes available, not to mention it having been transferred to a computerized trading system. ASTA can be configured to execute the simulated trades, using prices from either day T or day T+1 for buy and sell. The data for day T is assumed to be available for analysis on day T.

2.3.2 How Many Stocks to Buy and Sell?

The trading rule $T(s)$ does not contain any guide as to how many stocks or how big a portion of the cash should be invested in the particular stock that gets a buy signal. It is obvious that this fact affects the performance of a trading system, and it is also clear that a risk estimate coupled with the trading signal could be of help. The present version of ASTA does not offer any sophisticated procedures for the selection of the number of stocks to buy. The agent simply divides all available cash equally among the stocks that get a buy signal (taking into account the upper and lower limits for one individual trade.) When a stock gets a sell signal, all shares are sold.

2.3.3 Reinvesting Money

The Artificial Trader buys and sells stocks and hopefully increases the wealth during the simulated time period. In such a strategy there is a cumulative effect, since old profits are reinvested in future trades. The reinvesting is necessary if the trading results are to be comparable with the change in index, e.g. in equity diagrams as presented in Figure 4.

2.3.4 Interest on Cash

The money not invested in stocks is assumed to yield interest in a bank account. The level of interest varies over time, and in our test database follows the daily Swedish 3-month-bond interest rate. Refer to Section 4.17 for more information.

2.3.5 Unused Buy and Sell Signals

One problem of the basic approach of evaluation by simulated trading is that a large portion of the buy signals is neglected because there is a limited budget for the artificial trader. Even sell signals are neglected, simply because the stock that issues the signal is not in the trader's portfolio. Tests show that more than 90% of the signals often are ignored in this way. This seriously affects the statistical basis of a performance analysis for the trading algorithm. A possible way to attack the problem is a top level loop with multiple runs and randomization of entities like:

1. The starting date of trading
2. The acceptance of buy signals
3. The set of available stocks for trading

The present version of ASTA implements method 2 above. Since the running time of one simulation of a 10-year trading takes in the order of 1 minute on a 500 MHz Pentium PC the method is implemented as an optional feature. Refer to Section 4.9 for more information.

3 Using ASTA

3.1 Installation and Startup

The ASTA system has been developed in Matlab Version 5.3.0 but works fine also in Version 5.2. ASTA is installed by decompressing the installation file to a directory e.g. *c:\matlabx* (assuming that you are installing it on a PC). The decompression creates subdirectories *asta, asta\system, thutil and asta\work*. Help files for the first three of these directories are available by issuing the *Help* command in the Matlab command window. The supplied database (asta0.mat) with the Swedish stock data is placed on the *asta* directory.

When running ASTA, make Matlab's "current directory" *asta\work*.

It is convenient to create a separate "ASTA icon," which is a copy of the Matlab icon but with *asta* as "start in" catalog. In this way the *startup.m* in the *asta* directory is run automatically.

If you prefer to start it from your "ordinary" Matlab, take the following steps:

Enter `cd matlabx\asta`

Enter `startup`

startup.m adds paths and calls *wasta.m,* which is the Graphical User Interface for the system. If the paths are already set up, you can start the system directly by calling *wasta.m.*

ASTA is available in two versions:

- As a Windows application to be run under Matlab. The Buy rule and Sell rule and all other settings are controlled interactively and the results are presented on the screen.
- As a Matlab function, fasta.m, that takes a Buy rule and Sell rule as in parameters and computes the performance for a selected time period.

Both versions are described in separate sections below in this chapter.

3.2 Databases

The stock data is stored in a Matlab (.mat) file, which you load prior to the trading simulation. The following two sections describe how to load the stock data file. One database *asta0.mat* with data from the Swedish stock market is supplied with ASTA system and is installed to the *asta* directory during the installation process. American data from the Dow Jones index has also been converted to an ASTA database, but is unfortunately not freely available. However, examples from the American stock market can be found later on in this report. Other data sources can be adapted to the system as well. The relevant Matlab functions for this task can be found on the *asta/system* directory. However, the process of adapting new data to ASTA is not further described in this report.

3.3 FASTA

The purpose of the "batch" version of the program is to provide an objective function, *fasta.m*, that can be used for parameter tuning or automated generation of trading rules. This report does not describe *fasta* in more detail. The following example shows how it can be used:

Example 1 *:*

```
% load the workspace with stock data:
load('asta0')
% Simulate trading with Buy and Sell rules for 1990-1995.
% The performance is returned in the p variable.
p = fasta('Clos(T)>Clos(T-1)','Profit>20 | Loss>10',[],[90 95])
```

The following example shows how free parameters in the Buy and Sell expressions can be set by the third argument to the *FASTA* function. In this way repetitive simulations can easily be executed. One example of this usage can be found in [7] where *FASTA* is used as the objective function in a multi dimensional optimization of trading rules.

Example 2 *:*

```
load('asta0')
% The Buy and Sell rules contain the free parameter S.
% S is set in the third argument to FASTA.
p = fasta('Stoch(30,3,3,S,80)>0', 'Stoch(30,3,3,S,80)<0',...
                'S=30',[90 95])
```

3.4 WASTA

ASTA can be also run as an interactive application in the Matlab system. In this report many examples from this mode are demonstrated.

To start ASTA, run the *wasta.m* function or follow the procedure described in the beginning of this chapter. The screen layout is shown in Figure 2. The stock data is picked from a database file with raw data.

Enter the name of this file into the field to the right of the *Load* button, then click it to load the stock data into the Matlab work space. The data file freely

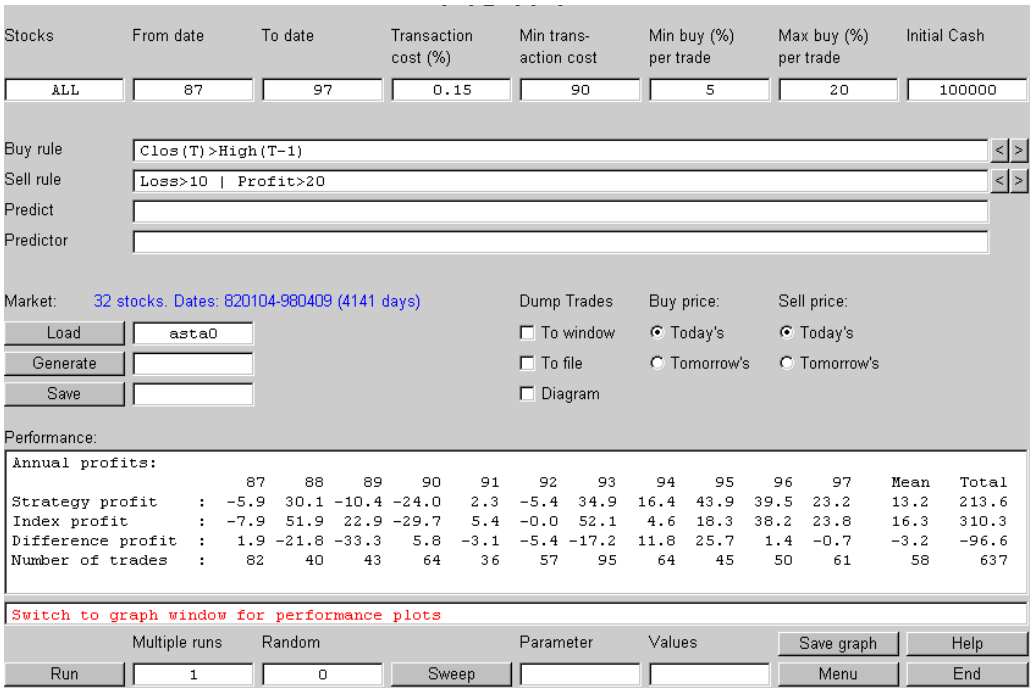


Figure 2: Screen layout for the Windows version of ASTA

distributed with ASTA is named *asta0.mat*. It contains data for 32 major stocks from the Swedish stock market between the years 1982 and 1998.

In the shown picture, you can set up parameters for a simulated trading. The most interesting items are the lines "Buy rule" and "Sell Rule". This is where the trading algorithm is decided. The rules follow Matlab syntax and can include any of the large number of predefined functions, or the your own functions with new algorithms. The simulation of trading starts by clicking the "Run" button. The results are presented in tabular form and in graphs as in Figure 4.

The command button "Sweep" initiates a whole series of simulations with different values on a symbolic parameter in the Buy and Sell rules. The name of the parameter is provided in the "Parameter" text box, and the values to be included in the multiple simulations in the "Values" text box. The results are presented in 6 graphs, which are also written as encapsulated postscript (eps) files. Examples can be found in Section 6.1.

3.5 Example of Simulated Trading

This section presents one example from the Windows version of ASTA.

32 major stocks with active trading from the Swedish stock market for the years 1987-1997 have been selected for analysis. The main ASTA screen is shown in Figure 3. The fields "Buy rule" and "Sell Rule" are where the trading algorithm is defined. The rules follow the Matlab syntax and can include the predefined functions or the users' own functions with new algorithms.

The example shows a test of the Stochastics indicator $Stoch[\ t,\ K,\ KS,\ D,$

Buylevel, *Sellevel*], common in technical trading, and here defined as:

$$\begin{aligned} Stoch(t) &= mav(100 * (Clos(t) - L) \ / \ (H - L),\ D) \quad \text{with} \\ L &= mav(\min(Low(T - K : T)),\ KS) \quad \text{and} \\ H &= mav(\max(High(T - K : T)),\ KS). \end{aligned} \tag{1}$$

The parameters K, KS, and D are the window lengths of the moving average function mav.

A Buy signal is issued when $Stoch(t) > Buylevel$ and a Sell signal when $Stoch(t) < Sellevel$. The parameters K, KS, D, $Sellevel$, and $Buylevel$ control the performance of a trading strategy based on the indicator and are subject to analysis in the next section.

In Figure 3, the "standard" values $30, 3, 3, 20, 80$ are used for the parameters. The results shown in Figure 3 and Figure 4 are quite stunning, with an average annual profit of 53.4% compared to the 16.3% achieved by the index. It is noteworthy that the only negative result is for the year 1997, where the strategy only made 1.7% whereas the index increased by 23.8%.

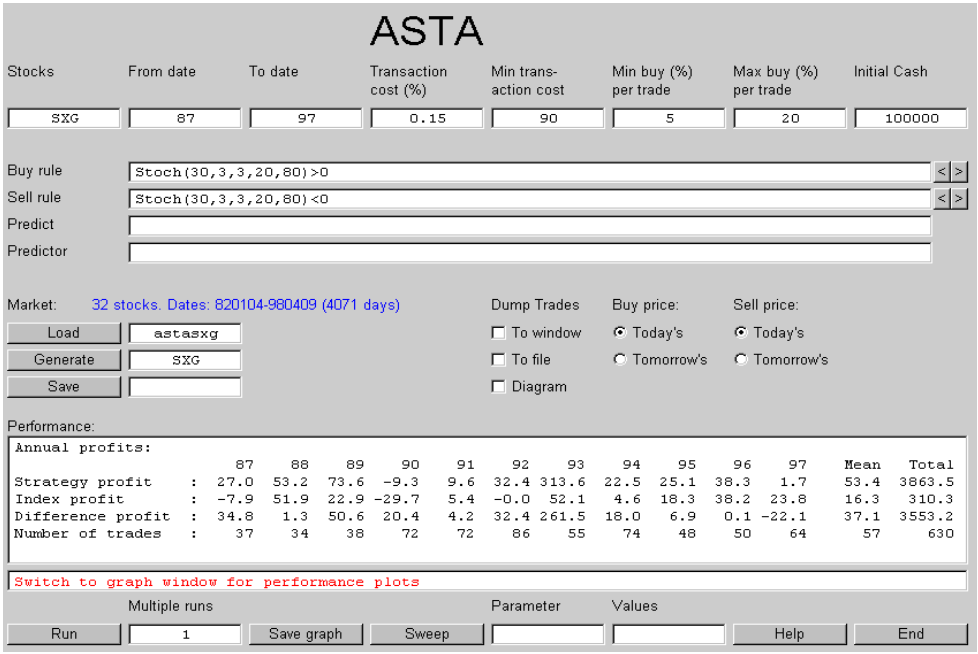


Figure 3: ASTA command window with Stochastics buy and sell rules

3.6 Fixed Horizon Predictions

ASTA has been primarily developed to evaluate Buy and Sell rules by simulated trading as described in the previous section. The other way to evaluate trading rules is a fixed prediction-horizon approach. It is implemented is ASTA in the following fashion. The entity to be predicted is input in the Predict field. The entity that should produce the predictions is input in the Predictor field. The Predictor expression is a function of stock data (close, high , low and volume) available at time

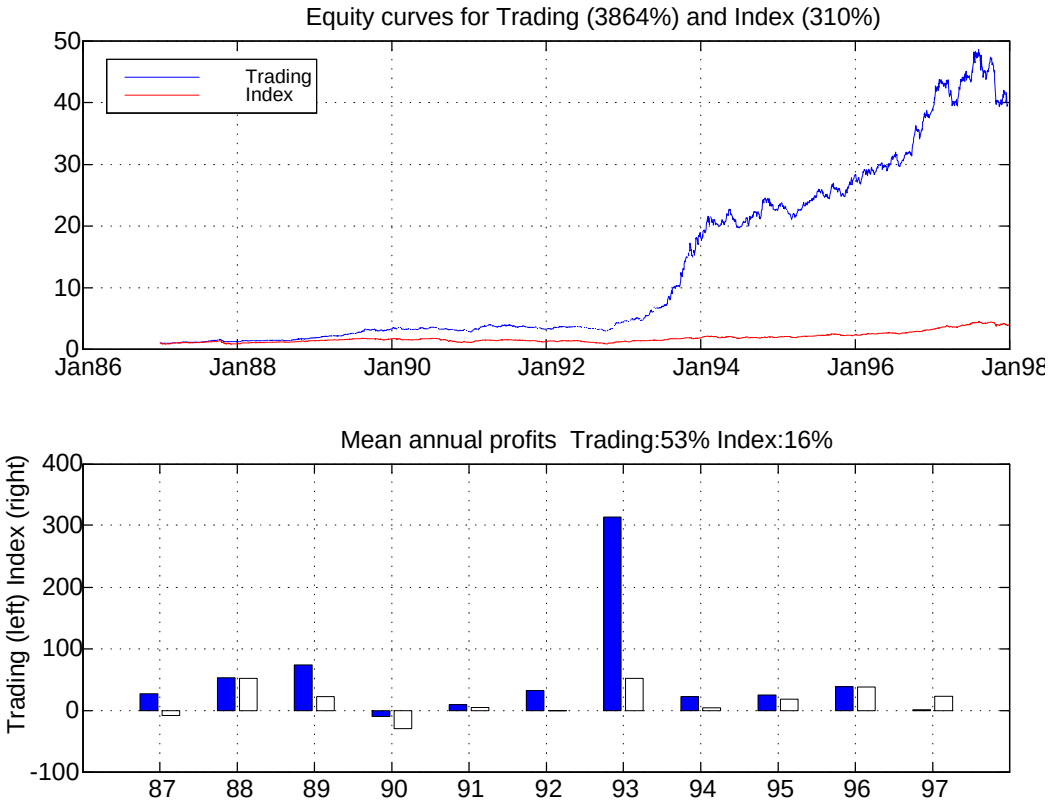


Figure 4: Performance of the Stochastics indicator.

T. The Predict expression is typically a function of data available at a time >T[1]. During execution the two expressions are computed for stocks and times, where the Buy rule evaluates to True, i.e. for the occasions when stocks should be bought according to the Buy rule. In this way a selection procedure can be combined with the prediction task. If the Buy rule is left empty, the Predict and Predictor expressions are computed for all steps during the simulation interval. Performance for the predictions is finally output with the following entities:

rmse	The square root of the mean squared prediction error over all predictions.
nrmse	rmse divided by the average value of the *Predict* entity for each stock.
npred	Number of predictions.
hitrate/pnts	Hit rate for the sign of all non zero predictions. The number of these predictions are *pnts*.
+hitrate/pnts	Hit rate for the sign, computed for all predictions>0. The number of these predictions are *pnts*.
-hitrate/pnts	Hit rate for the sign, computed for all predictions<0. The number of these predictions are *pnts*.

(2)

[1]Actually, the only important thing is to have correct time ordering of Predict and Predictor. As is shown in the following examples it is perfectly in order to express Predict at time T and Predictor at time T-1.

The performance measures are computed for each stock and is written in the Matlab command window one line for each stock. A summary for all the stocks is presented in the lower pane of the ASTA command window.

The functions entered in the Predict and Predictor fields can be composed from the large number of predefined functions described in chapter 5.

To illustrate the usage a simple example with correlations between international stock indexes is given. There is a common belief that the Dow Jones index has a great influence on other countries' stock markets. We would like to test and quantify that belief for the Swedish stock market measured by the official stock index Generalindex. The Predict and Predictor fields in ASTA are input as follows:

Predict:	Trendk(1,T)
Predictor:	Trendk(1,T-1,'DJ')
Stocks:	SXGEN

(3)

The Predict field is the 1-day trend for the investigated security computed at day T. The Predictor field is the 1-day trend for the Dow Jones computed at day T-1. The Stocks field defines the securities for which the Predict field is computed. The predefined symbol SXGEN denotes Generalindex and this security is the only one tested in the simulation. In Figure 5, the ASTA system for the task is shown for the time period 1989-1997. By leaving the fields Buy rule and Sell rule empty the performance for every day in the time period is computed when the Run button is clicked. The results after the simulation is completed are shown in the lower pane of the screen. The often most interesting performance measures are the total hit rate and the hit rate for the positive and negative predictions separately. The following are the results of the performance calculation:

hit rate	points	+hitrate	points	-hitrate	points
58.74	2002	61.17	1092	55.82	910

(4)

I.e.: an increase in New York is followed in 61.17% of the cases by an increase in Stockholm the following day. A decrease in New York is followed in 55.82% of the cases by a decrease in Stockholm the following day.

Next we want to examine if a screening of very small changes in the Dow Jones index improve the hitrate. This is achieved by adding a Buy rule to the system:

Buy rule:	abs(Trendk(1,T-1,'DJ'))>0.5

(5)

The addition of a Buy rule causes a selection of points to where the Buy rule evaluates to True. In this case we select days where the Dow Jones index has changed by at least 0.5% since the previous day. The presented results show that 906 days out of the total 2048 are selected. The hitrate for predicted increases in Generalindex is as high as 70.85% for these days.

4 The ASTA command window

The ASTA command window contains most of the information necessary to operate the system. Some additional commands are available through a popup menu,

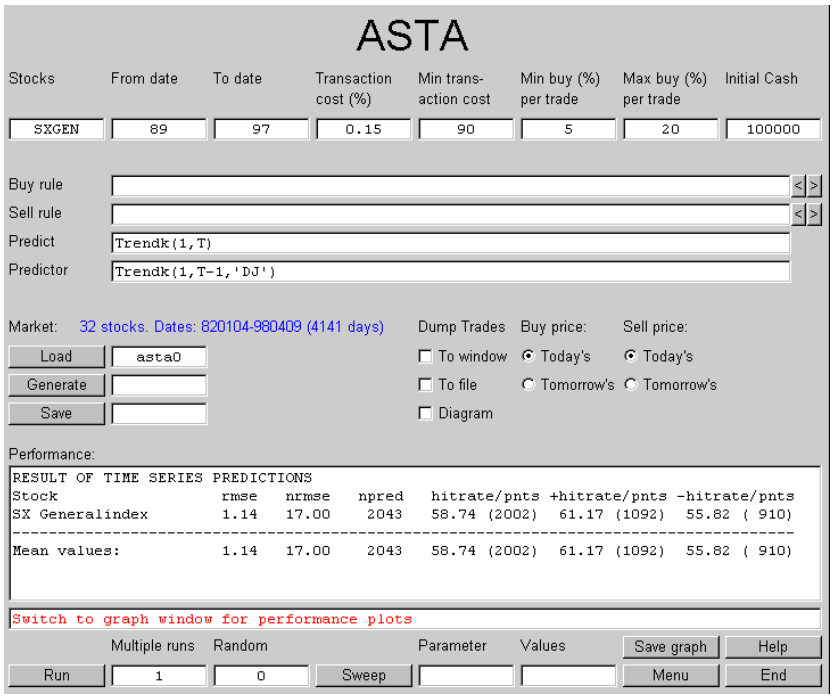


Figure 5: 1-day prediction for the Swedish Generalindex. The predictor is the sign of the previous day's return for Dow Jones.

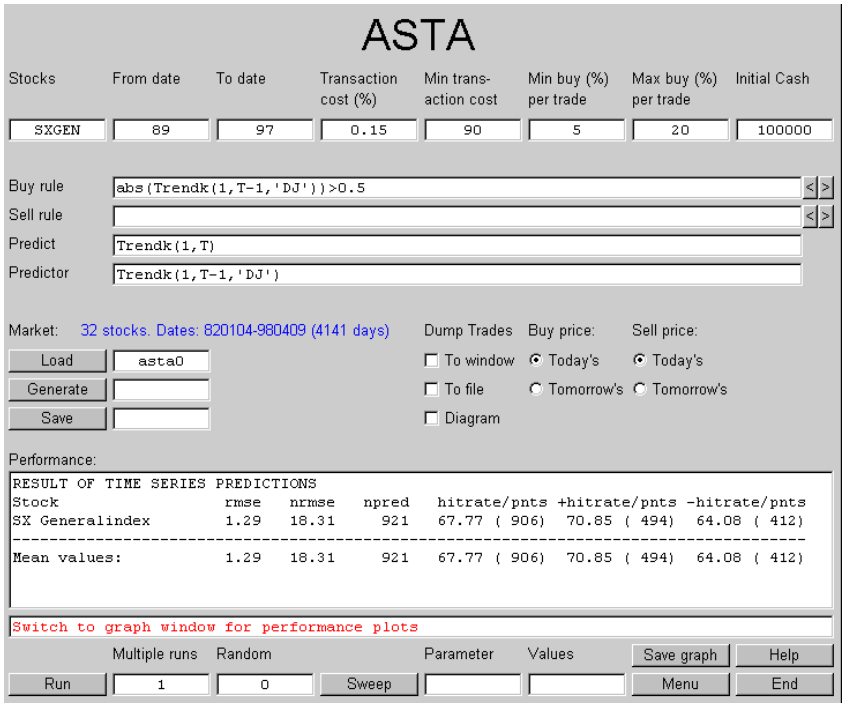


Figure 6: Same fixed-horizon prediction as in Figure 6. The predictions are only evaluated for those days where the Buy rule returns a signal. This occurs when the Dow Jones has changed at least 0.5% since the previous day.

activated by the Menu button. This section describes the available commands and setups by which the user can control the ASTA simulations.

4.1 Stocks

This field is meant to contain an expression for the particular stocks that should be included in the simulation. The predefined symbol **ALL** is a vector containing the numbers for all stocks available in the loaded database. **ALL** may be indexed to access individual stocks. The indexes for all stocks are listed by the Menu command "Statistics" (see Figure 8). Other symbols such as stock indexes are often also predefined in the database. These symbols are also listed by the "Statistics" command.

Example 3 *Stocks* : $ALL(1:5)$
The simulation includes the first 5 stocks as listed in the "Statistics" command.

Example 4 *Stocks* : $SXGEN$
The simulation includes only the Swedish Generalindex, which is predefined as the symbol $SXGEN$*.*

4.2 From Date / To Date

The simulation runs between these two dates, which can be given in format $YYMMDD$ or YY.

Example 5 *From date* : 87 *To date* : 88
The simulation runs between January 1st 1987 and December 31st 1988.

4.3 Transaction cost% / Min transaction cost

The transaction costs can be defined by two fields; *Transaction cost%* and *Min transaction cost.* The following amount is subtracted from the trader's wealth when stocks for a total amount S have been bought or sold:

$$\max(S * Transaction\ cost\%,\ Min\ transaction\ cost) \tag{6}$$

By setting the *Transaction cost%* to 0, a fixed non-proportional transaction cost is used.

4.4 Min / Max Buy per Trade

To avoid too small or too large individual trades, a lower and upper bound for each trade can be set in the fields *Min buy (%) per trade* and *Max buy (%) per trade.* The values are expressed as fractions of the trader's current wealth.

4.5 Initial Cash

The Initial Cash is the amount of initial wealth for the Trader. Since all performance calculations are relative measures, the exact amount does not normally affect the performance for a particular trading strategy

4.6 Buy rule / Sell rule

The *Buy rule* and *Sell rule* are used to generate trades in the trading simulation. They evaluate to TSM's and a signal is issued for a stock if the corresponding column in the TSM is non-zero. The expressions in these fields must not reference data into the future, i.e. times>T where T is the global variable that represents the present time in the simulation. If times>T are referenced, an error message is issued. The scroll buttons to the right of the fields can be used to select from a number of predefined expressions. The rules may contain several Matlab expressions separated by semicolons. The value of the rule is in such a case the last expression in the compound expression.

Example 6 $Use('SXGEN');\ x = Gvol10 > 1; Use('');\ x\ \&\ Trendk(10) > 2$
The compound expression uses an intermediate variable x *and has the value* $x\ \&\ Trendk(10) > 2$ *when used as a Buy rule or Sell rule.*

More complex *Buy rules* and *Sell rules* are preferably written as separate Matlab functions.

4.7 Predict

This field may be used if a fixed-horizon prediction should be simulated instead of a Trading situation. The Predict field should contain the entity to be predicted. Often it is convenient to predict the sign instead of the absolute values of future returns. E.g.: sign(Trendk(1,T+1)). The safeguard against looking into the future (as described in the previous section) is NOT active for this field.

4.8 Predictor

This field may be used if a fixed-horizon prediction should be simulated instead of a Trading situation. The Predictor field should contain the expression that predicts the entity in the Predict field. E.g.: sign(Trendk(1,T)). The safe guard against looking into the future is NOT active for this field since the important thing is that the data used to compute the entity in the Predictor field must not be available at times referenced in the Predict field. This can not automatically be checked in a simple way and is therefore left as the user's responsibility.

4.9 Multiple Runs / Random

As has been previously mentioned, one problem with the basic approach of evaluation by simulated trading is that a large portion of the buy signals are neglected due to lack of money available for the artificial trader. Even sell signals are neglected, simply because the stock that issues the signal is not in the trader's portfolio. Tests show that more than 90% of the signals often are ignored in this way. This seriously affects the statistical basis of a performance analysis for the trading algorithm. Therefore, a system with multiple runs and randomization of the acceptance of buy signals has been implemented:

Example 7 *By setting Random to 50, 50% of the buy signals are randomly ignored. By setting Multiple runs to 10, 10 separate simulations are executed. In each run different buy signals are ignored and the finally presented performance is the mean value for these 10 separate simulations.*

The value 0 for the field *Random* causes no randomization to take place. This is the default value.

4.10 Dump Trades

The individual trades may be inspected in detail by the following options:

- **To window**. The trades are printed in the Matlab window. Example: In Figure 14, the box "Dump trades to window" has been checked. All generated trades are printed in the Matlab command window as shown in Figure 7.
- **To file**. The trades are written to a data file for external analysis.
- **Diagram**. The trades are marked in the diagram. Refer to Figure 15.

4.11 Buy Price / Sell Price

In a real trading situation on day T, the Close, High, and Low for day T are not available. However, it is common that prediction algorithms assume that this data is available, and can be included in the prediction of the following day's prices. Furthermore, it is often assumed that the Buy and Sell recommendations from the trading system can be executed using the close prices for day T. Of course, in reality this is hard to achieve, since the stock market is closed by the time the data for the current day becomes available, not to mention it having been transferred to a computerized trading system. The radio buttons *Today's* and *Tomorrow's* can be set to execute the simulated trades, using prices from either day T or day T+1 for buy and sell.

4.12 Sweep

The buy and sell rules may include a variable, which is assigned values by this command option. The name of the parameter should be input in the *Parameter* field and the range should be input in the *Values* field. Clicking *Sweep*, causes a sequence of trading simulations to take place. Before each simulation, the variable is assigned a value from the *Values* field. The result is presented in a number of graphs, which are also saved as .eps files.

Example 8

Buy Rule	*Rank5<x*
Sell Rule	*Stoploss(15,-5,5) | Profit>40*
Parameter	*x*
Values	*-0.48:0.01:-0.40*

The x parameter in the buy rule is set to the values -0.48,-0.47,-0.46,-0.45,-0.44,-0.43,-0.42,-0.41,-0.40 in 9 consequtive trading simulations. The performance is presented in a number of graphs.

Trade	Date	No.	Stock	Price	Total Profit (%)		Wealth	Cash
1	930105	77	21 Skandia	85.27	6566.05			93343.95
2	930105	79	27 Sydkraft C	82.60	6525.22			86728.73
3	930105	1535	31 Allgon B	4.28	6574.72		99730.00	80064.01
4	930107	375	20 SHB A	26.34	9875.96			70119.50
5	930107	329	32 Nokia A	30.00	9870.00		99585.62	60159.50
6	930108	349	10 Hennes & Mauritz	28.20	9841.80			50260.08
7	930108	323	25 SSAB A	30.50	9851.50		99860.44	40318.58
8	930115	431	7 Avesta Sheffield	22.68	9776.63			30528.47
9	930115	351	20 SHB A	27.84	9771.77		100374.14	20666.71
10	930121	258	2 ABB A	38.40	9907.20			10702.32
11	930121	64	14 Kinnevik B	81.00	5184.00		NaN	5428.32
12	930127	54	21 Skandia	97.46	5262.68		NaN	84.48
13	930302	-64	14 Kinnevik B	79.00	-5056.00	-2.47	128010.13	5051.23
14	930322	-131	21 Skandia	93.71	-12275.62	3.78	131216.32	17260.75
15	930323	-431	7 Avesta Sheffield	23.16	-9980.37	2.08	NaN	27165.09
16	930325	-323	25 SSAB A	33.75	-10901.25	10.66	132663.45	37991.32
17	930329	4897	31 Allgon B	5.45	26675.18		132626.85	11247.39
18	930406	2031	31 Allgon B	5.51	11186.34		132246.33	1.77
19	930519	-258	2 ABB A	43.70	-11274.60	13.80	164709.29	11186.39
20	930525	113	22 Skanska B	97.50	11017.50		NaN	94.54
21	930608	-329	32 Nokia A	46.25	-15216.25	54.17	187009.69	15221.10
22	930610	154	27 Sydkraft C	97.95	15083.58		184210.74	54.44
23	930621	-349	10 Hennes & Mauritz	38.00	-13262.00	34.75		13226.55
24	930621	132	22 Skanska B	99.00	13068.00		185878.28	68.55
25	930716	-233	27 Sydkraft C	93.56	-21799.15	0.88	197758.60	21778.13
26	930719	308	7 Avesta Sheffield	34.97	10770.39		197540.39	10922.70
27	930720	235	2 ABB A	46.00	10810.00		199141.88	30.11
28	930906	-235	2 ABB A	47.20	-11092.00	2.61	273799.16	11032.41
29	930916	-308	7 Avesta Sheffield	36.86	-11352.70	5.41	272928.45	22322.39
30	930920	227	9 Ericsson B	97.83	22207.73		272118.39	33.78
31	930921	-245	22 Skanska B	138.00	-33810.00	40.37	270682.53	33753.80
32	930922	-726	20 SHB A	95.00	-68970.00	251.03	269365.98	102627.47
33	930923	672	32 Nokia A	80.00	53760.00		270719.82	48798.81
34	930927	562	10 Hennes & Mauritz	43.20	24278.40		282121.66	24450.66
35	931004	495	2 ABB A	49.20	24354.00		309655.25	42.43
36	931108	-227	9 Ericsson B	102.87	-23350.81	5.15	NaN	23303.54
37	931111	136	21 Skandia	170.55	23194.39		429195.43	42.48
38	931126	-672	32 Nokia A	100.50	-67536.00	25.62	399726.79	67477.30
39	931203	764	30 Volvo B	88.20	67384.19		387109.87	85.89
40	931215	-136	21 Skandia	159.30	-21664.94	-6.59	434649.03	21661.02
41	931221	127	22 Skanska B	169.00	21463.00		443866.23	132.74

Figure 7: Dump of generated trades from Stochastic trading rules for the year 1993.

The *Sweep* function is often combined with the *Multiple runs / Random* function described in Section 4.9 above. For an example refer to Section 6.1 and Figure 13

4.13 Load

The name of an ASTA database should be input in the field to the right of the Load button. By clicking the button the stock data is loaded into the work space. The default value when starting ASTA is *asta0,* which is a database with 32 stocks from the Swedish stock market between 1982 and 1987.

4.14 Generate

This option is not available in the free version of ASTA. It is used when generating ASTA databases from external data sources.

4.15 Save

Some functions cause data to be computed and temporarily stored to speed up further simulations. These functions are: *Trend1, Trend2, Trend5, Trend20, Rank1, Rank2, Rank5, Rank20, Gvol2, Gvol5, Gvol10, Gvol20, Volat2, Volat5, Volat10* and *Volat20.* The menu command *Compute Stock potentials* does also compute values in the same way. The computed data is available during the current session but has to be recomputed the next time the database is loaded. By using the *Save* option these pre-computed entities can be saved on a disk and reloaded later on by the *Load* command. Fill in a file name (e.g.: *asta1*) in the field to the right of the *Save* button and click on the button.

4.16 Save Graph

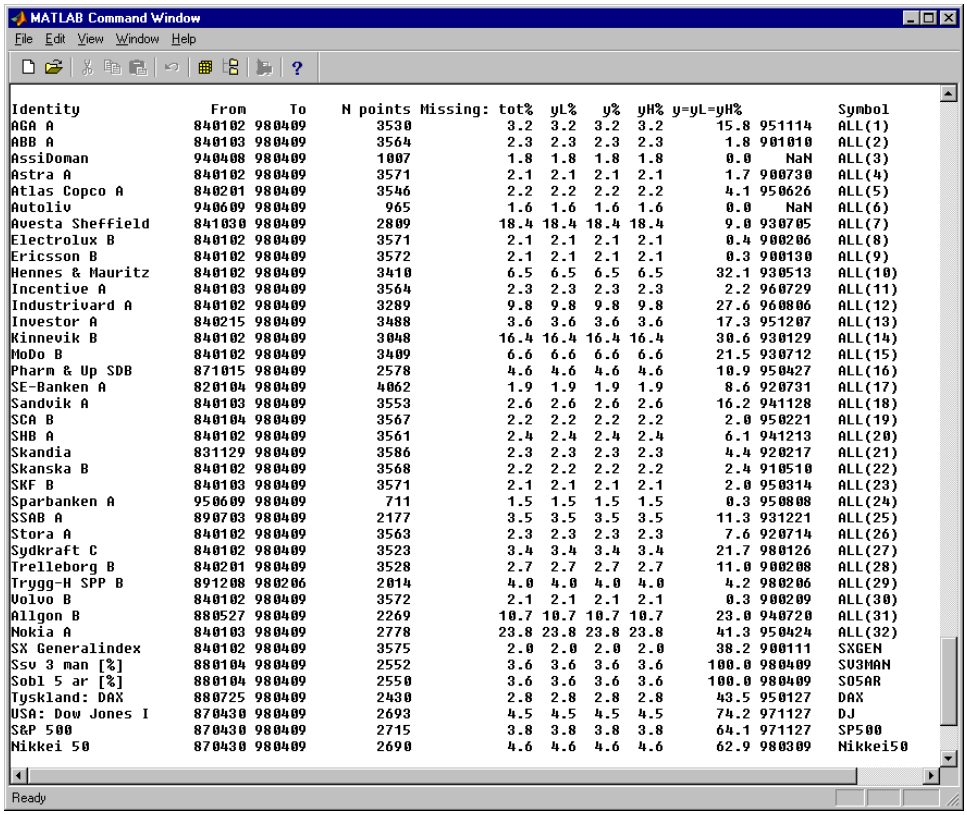
The performance for the trading simulation is presented in a graph such as Figure 4. By clicking the *Save Graph* button, this graph is saved as an .eps file, *astapic.eps*

The corresponding ASTA command window is saved as file *astacmd.eps.* These files can be easily included in various kinds of documentation.

4.17 Menu

The Menu button displays a popup menu with the following additional commands:

- **Statistics for securities.** Prints summary statistics for the stocks in the loaded database. Information regarding missing data for Close, Low, and High (y, yL, yH) is also presented. Refer to Figure 8.
- **Choose Index.** Any one of the securities in the loaded database may be used as basis for the computation of what is regarded as Index. This security is used as benchmark in all presented performance measures. Refer to Figure 10.
- **Choose Interest rate.** Any one of the securities in the loaded database may be used as as Interest Rate. Refer to Figure 9.
- **Compute Stock potentials.** Computes the Stock Potential with an expression provided by the user. Refer to Figure 11.and to Section 5.7.3 for details.
- **Set debug level.** Controls debug printouts from the system. Refer to Figure 12.



Identity	From	To	N points	Missing: tot%	yL%	y%	yH%	y=yL=yH%		Symbol
AGA A	840102	980409	3530	3.2	3.2	3.2	3.2	15.8	951114	ALL(1)
ABB A	840103	980409	3564	2.3	2.3	2.3	2.3	1.8	901010	ALL(2)
AssiDoman	940408	980409	1007	1.8	1.8	1.8	1.8	0.0	NaN	ALL(3)
Astra A	840102	980409	3571	2.1	2.1	2.1	2.1	1.7	980730	ALL(4)
Atlas Copco A	840201	980409	3546	2.2	2.2	2.2	2.2	4.1	950626	ALL(5)
Autoliv	940609	980409	965	1.6	1.6	1.6	1.6	0.0	NaN	ALL(6)
Avesta Sheffield	841030	980409	2809	18.4	18.4	18.4	18.4	9.0	930705	ALL(7)
Electrolux B	840102	980409	3571	2.1	2.1	2.1	2.1	0.4	980206	ALL(8)
Ericsson B	840102	980409	3572	2.1	2.1	2.1	2.1	0.3	980130	ALL(9)
Hennes & Mauritz	840102	980409	3410	6.5	6.5	6.5	6.5	32.1	930513	ALL(10)
Incentive A	840103	980409	3564	2.3	2.3	2.3	2.3	2.2	960729	ALL(11)
Industrivard A	840102	980409	3289	9.8	9.8	9.8	9.8	27.6	960806	ALL(12)
Investor A	840215	980409	3488	3.6	3.6	3.6	3.6	17.3	951207	ALL(13)
Kinnevik B	840102	980409	3048	16.4	16.4	16.4	16.4	30.6	930129	ALL(14)
MoDo B	840102	980409	3409	6.6	6.6	6.6	6.6	21.5	930712	ALL(15)
Pharm & Up SDB	871015	980409	2578	4.6	4.6	4.6	4.6	10.9	950427	ALL(16)
SE-Banken A	820104	980409	4062	1.9	1.9	1.9	1.9	8.6	920731	ALL(17)
Sandvik A	840103	980409	3553	2.6	2.6	2.6	2.6	16.2	941128	ALL(18)
SCA B	840104	980409	3567	2.2	2.2	2.2	2.2	2.0	950221	ALL(19)
SHB A	840102	980409	3561	2.4	2.4	2.4	2.4	6.1	941213	ALL(20)
Skandia	831129	980409	3586	2.3	2.3	2.3	2.3	4.4	920217	ALL(21)
Skanska B	840102	980409	3568	2.2	2.2	2.2	2.2	2.4	910510	ALL(22)
SKF B	840103	980409	3571	2.1	2.1	2.1	2.1	2.0	950314	ALL(23)
Sparbanken A	950609	980409	711	1.5	1.5	1.5	1.5	0.3	950808	ALL(24)
SSAB A	890703	980409	2177	3.5	3.5	3.5	3.5	11.3	931221	ALL(25)
Stora A	840102	980409	3563	2.3	2.3	2.3	2.3	7.6	920714	ALL(26)
Sydkraft C	840102	980409	3523	3.4	3.4	3.4	3.4	21.7	980126	ALL(27)
Trelleborg B	840201	980409	3528	2.7	2.7	2.7	2.7	11.0	980208	ALL(28)
Trygg-H SPP B	891208	980206	2014	4.0	4.0	4.0	4.0	4.2	980206	ALL(29)
Volvo B	840102	980409	3572	2.1	2.1	2.1	2.1	0.3	980209	ALL(30)
Allgon B	880527	980409	2269	10.7	10.7	10.7	10.7	23.0	940720	ALL(31)
Nokia A	840103	980409	2778	23.8	23.8	23.8	23.8	41.3	950424	ALL(32)
SX Generalindex	840102	980409	3575	2.0	2.0	2.0	2.0	38.2	980111	SXGEN
Ssv 3 man [%]	880104	980409	2552	3.6	3.6	3.6	3.6	100.0	980409	SV3MAN
Sobl 5 ar [%]	880104	980409	2550	3.6	3.6	3.6	3.6	100.0	980409	SO5AR
Tyskland: DAX	880725	980409	2430	2.8	2.8	2.8	2.8	43.5	950127	DAX
USA: Dow Jones I	870430	980409	2693	4.5	4.5	4.5	4.5	74.2	971127	DJ
S&P 500	870430	980409	2715	3.8	3.8	3.8	3.8	64.1	971127	SP500
Nikkei 50	870430	980409	2690	4.6	4.6	4.6	4.6	62.9	980309	Nikkei50

Figure 8: Statistics summary for the loaded stocks. The summary is useful to identify time periods where most of the stocks have data available. The defined symbols are also shown to the right in the printout.

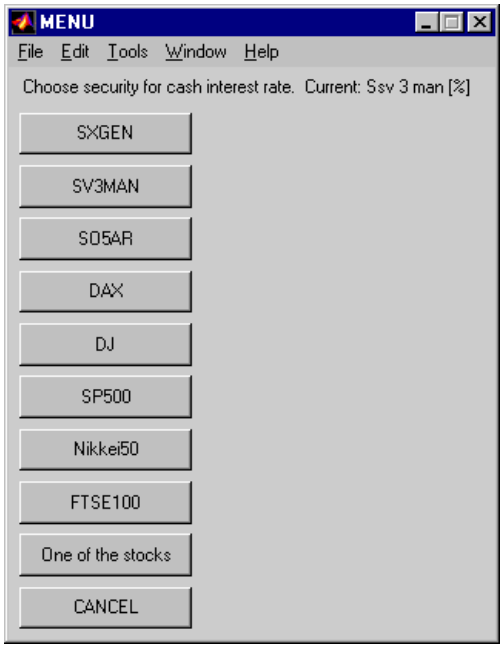


Figure 9: Setup of security to be used as interest rate for non-invested capital in the trading simulations. In most cases, the short or long rate is used.

Figure 10: Setup of security to be used as as comparative index in the trading simulations. In most cases, the national index correspoding to the stocks is used.

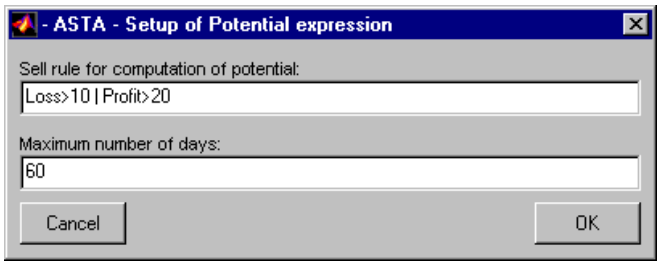


Figure 11: Definition of the expression used by the Potential function.

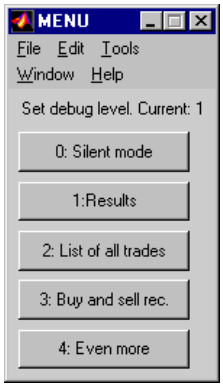


Figure 12: Adjustment of print level for printouts. The default is "1:Results"

5 Developing Trading Algorithms with ASTA

One of the key goals of using ASTA has been to provide a solid and easy-to-use basis for development of trading algorithms. The algorithm should be formulated in a day-by-day structure, where buy and sell recommendations are produced daily by the algorithm. The decisions should be based on past stock data only. Fundamental data about the companies cannot be used in the present version of the system. It should be also emphasized that the "Trading Rule" approach implies a decision-making system, and not a modeling system. However, implemented algorithms may use modeling "internally" and base the buy and sell recommendations upon it.

The following goals and guidelines have been kept in mind while designing the user interface in ASTA:

- Foolproofness

  An "off-by-one" error seldom results in more drastic consequences than when developing prediction algorithms. Looking into the future, which is normally pretty hard to do, is in the case of prediction algorithms surprisingly easy to do, even unintentionally. In the case of indexing time series vectors however, it should not come as a surprise that finding an error in code segments, like the following one, is indeed difficult.

  $i = k - h + 1$

  $x(m) = y(k + 2 * (j - j) - 1)...$

  Therefore, the access functions implemented in ASTA have been designed with a check for arguments that peep into the future. The global variable T is automatically updated by the simulation mechanism to point to the "current day". Attempts to address the stock time series after this point, automatically issue an error message. Example:

  $y = Clos(T - 4 : T)$ assigns the close prices for the current day and the 4 previous days to the variable y.

  $y = Clos(T : T + 1)$ issues an error message when executed during the simulation.

  The same check routine has been implemented in the higher-level routines as well, thereby issuing clearer error messages pointing to the first instance, where an illegal time reference is found.

- Ease of use

  Existing ASTA functions can be used interactively to create Buy and Sell rules. When the user has implemented a new trading algorithm, the Matlab function can be used also in the Buy and Sell rules. It can be easily tried out interactively with different settings on parameters etc.

- High level functionality

  Predefined ASTA functions can perform such operations as moving averages and detecting crossings between time series. Other predefined functions compute derived entities from the stock data time series: gaussian volume, volatility, trend, and ranks. These functions can be included in the user-defined algorithms, thus simplifying the coding considerably.

ASTA is equipped with a large number of predefined functions to be used interactively in Buy and Sell rules. They may be used also in user-defined functions. Before going into details of the available functions, some basic concepts are described.

5.1 Global Variables

The variables listed in Table 1 are dynamically set by the system and normally should not be changed by any user routines.

Name	Type	Size	Decription
T	numeric	1	Row pointer to "current" day
Stocks	numeric	$1xN$	Column pointers to allowed stocks. **Stocks** is set to all stocks in market when evaluating a **Buy rule** **Stocks** is set to all stocks in portfolio when evaluating a **Sell rule**
Market	struct	1	The **Market** object
Trader	struct	1	The **Trader** object
ST	struct	1	Parameters controlling the Trader's behavior
ALL	numeric	$1xK$	Vector with numbers for the K stocks. The numbers are listed by the menu command "Statistics". **ALL** may be indexed to access individual stocks. The command is primarily used in the *Stocks* field to select stocks to use in the simulation.

Table 1: Global variables in ASTA

The variable T is central both when defining Buy and Sell rules and when writing new functions for prediction. The idea is that the user should never have to worry about the time aspect, since it is automatically taken care of by the simulation framework. By using T as index, the "current" day is always referenced. By using T-1, the previous day is referenced etc. Attempts to use T+1 issue an error message, thus prohibiting the program from peeping into the future.

It is seldom necessary to use the variable Stocks when writing user-defined functions. However, it should be made clear that the variable is set to all stocks in the market when the system looks for stocks to buy (i.e. when the Buy rule is evaluated) and to all stocks in the Trader's portfolio when the system tries to find stocks to sell (i.e. when the Sell rule is evaluated). The function call Clos(T) for example, returns a row vector with the close values for all stocks in the Stocks variable for the day T (the "current" day in the simulation).

5.2 Time Series Matrices

Most time series data within the ASTA system are enclosed in objects denoted Time Series Matrices, and are abbreviated TSM in this documentation. They are ordinary 2-dimensional Matlab matrices, where each row represents one date and each column represents one stock. The actual dates and actual stocks in a particular

matrix vary, but the last row most often corresponds to the "current" day T. The columns normally correspond to the stocks in global *stocks* variable.

The available data is the stock time series enclosed in the Market Object: Close, High, Low, and Volume. These are the only features available when creating Buy and Sell rules, and also when creating new prediction algorithms. Clos, High, Low, and Volume are the names of the access functions used throughout the system.

Example 9 *: The call* ***Clos(T-5:T)*** *returns a 6 row matrix, where each column is assigned the close value for a stock in the global* ***Stocks*** *variable. Row 1 corresponds to day* ***T-5****, row 2 to day* ***T-4,*** *down to row 6, which corresponds to day* ***T****.*

A whole range of functions that accept TSM's as arguments and/or generate it as output, are included in the system. This design principle turns out to be convenient, since the column oriented matrix operations within Matlab can be used, and makes the code very readable and effective. The Matlab function OHIGH in the following example detects *Clos* values that cross a k day maximum of *High* from below.

Example 10 *:*

```
function b = OHIGH(k)
  C = Clos(T);
  Cm1 = Clos(T-1);
  M = max( High( T-k:T-1 )); % The call to High returns a k
                             % rows long Time Series Matrix
  b = C>M & Cm1<=M;
return
```

The function returns a binary row vector with a "1" in those columns where the corresponding stock fulfilled the condition. It is an example of the data type *Indicator Vector*.

5.3 Indicator Vectors

An Indicator Vector is a single-row TSM. Sell and Buy rules should evaluate to binary Indicator Vectors used directly as sell and buy recommendations. Other Indicator Vectors, such as the return values from the function calls *Profit* and *Upfract*, contain real numbers as values.

In the previous section the function OHIGH returned an Indicator Vector with binary values, "0" or "1". A number of predefined functions return Indicator Vectors, and user-defined trading functions normally should return an Indicator Vector. This enables such compositions of Buy rules as:

$$\text{OHIGH(10) \& (Trendk(1)>1 | Trendk(5)>0.5)} \tag{7}$$

The Buy rule evaluates to a binary Indicator Vector with "1" for stocks where OHIGH(10) returned "1" and, where either the short trend is greater than 1%/day or the long trend is greater than 0.5%/day. Functions *Trendk* is described in section 5.6.

5.4 Predefined ASTA Functions

The following sections provide documentation of all predefined functions that have been included to enable expressing compound trading rules interactively (as Buy rules and Sell rules) and also to provide the developer of new algorithms with basic access functions, as well as some useful high-level functions.

The functions are divided into 4 categories:

1. Feature Functions

   Feature Functions are primarily access functions to stock prices (Close, High, Low) and traded Volume. Numerous functions for commonly-used derived entities, such as trend, gaussian volume, and volatility, are provided as well. A Feature Function always returns a TSM.

2. Indicator Functions

   In general an indicator function returns a TSM. Many indicator functions return a 1-row TSM (i.e. an Indicator Vector) and can therefore be used as is in the definitions of Buy rules and Sell rules. Examples are standard technical indicators, such as MACD and Stochastics. Other Indicator Functions implement a dynamic stop-loss handling and achieved profit or loss since a stock has been bought.

3. Operator Functions

   Operator functions perform a computation on one or several TSMs, and return another one. Examples are computation of moving averages, minimum values in time windows, etc.

4. Utility Functions

   Some functions are support functions that may be of use primarily when writing and debugging new algorithms.

Note:

All functions of types 1, 2, and 3 return a TSM. Each column represents one stock in the global variable Stocks. Stocks is automatically set to all stocks in the market when the system looks for stocks to buy (i.e. when the Buy rule is evaluated) and it is set to all stocks in the Trader's portfolio when the system tries to find stocks to sell (i.e. when the Sell rule is evaluated). It is seldom necessary to use the Stocks variable explicitly in Buy rules or Sell rules or when writing user-defined functions.

5.5 General Parameters

This section describes some parameters, which are common to many of the predefined functions.

5.5.1 Days

This parameter tells for which day or days the function should return values. In the case of days being a vector, the result is a TSM with the same number of rows as

elements in the *Days* parameter. Each row contains values as if computed on the corresponding day. The default value for the Days parameter is the global T, i.e. the current day.

Example 11 *: **Clos(T)** returns a single-row TSM with close prices for all the stocks in the global variable **Stocks**.*

Example 12 *: **Trend5(T-9:T)** returns a 10-row TSM with 5-day trends for all the stocks in the global variable **Stocks**.*

To simplify the description of functions below, the *days* parameter is not covered. The Description field in the tables refers to each row of the returned TSM. The default value for the T parameter is for most functions the current day in the ongoing simulation.

5.5.2 Plot

Many functions have the optional *plot* argument. By setting *plot* to the global constant PLOT, the function is plotted versus time for all the stocks selected in a row. The exact contents of the plot depends on the function. User-defined functions can easily incorporate the plotting functionality, which is extremely useful during development and debugging.

Only one function at a time can have the *plot* parameter set to a non-zero value (the global constant PLOT equals to 1). The default value for the *plot* parameter is 0.

Example 13 *By adding a plot parameter in the call to many of the predefined ASTA functions, illuminating graphs are generated. The call to the previously used Stoch function for example, may look like:*

$$Stoch(30,3,3,20,80,PLOT)>0. \tag{8}$$

In Figure 15 the components of the Stochastics indicator are displayed. The two horizontal lines mark the buy level (80%) and the sell level (20%). When the "Oscillator %K" value passes these lines, a buy or sell signal is issued.

5.6 Feature Functions

Feature Functions are primarily access functions for obtaining stock prices (Close, High, Low) and traded Volume. Derived entities such as ranks and trends are also provided.

All the Feature Functions return a TSM with one row for each day in the *days* parameter and one column for each stock in the global *Stocks* variable. Table 2 lists all predefined Feature functions in ASTA. The *plot* argument is described in Section 5.5.2. Note that the default value for the *Days* parameter is the current trading day T. The returned TSM contains in this case one row with one column for each stock in the global variable *Stock.*

Function	Parameters	Description
Clos	**days, plot**	The close value
High	**days, plot**	The highest traded price
Low	**days, plot**	The lowest traded price
Volume	**days, plot**	The traded volume value
Trend1	**days, plot**	The 1-day trend (%/day)
Trend2	**days, plot**	The 2-day trend (%/day)
Trend5	**days, plot**	The 5-day trend (%/day)
Trend20	**days, plot**	The 20-day trend (%/day)
Rank1	**days, plot**	The 1-day rank value
Rank2	**days, plot**	The 2-day rank value
Rank5	**days, plot**	The 5-day rank value
Rank20	**days, plot**	The 20-day rank value
Trendk	**k, days, use, plot**	The k-day trend (%/day)
Gvol2	**days, plot**	The 2-day gaussian volume
Gvol5	**days, plot**	The 5-day gaussian volume
Gvol10	**days, plot**	The 10-day gaussian volume
Gvol20	**days, plot**	The 20-day gaussian volume
Volat2	**days, plot**	The 2-day relative volatility
Volat5	**days, plot**	The 5-day relative volatility
Volat10	**days, plot**	The 10-day relative volatility
Volat20	**days, plot**	The 20-day relative volatility

Table 2: Predefined feature functions

5.6.1 Clos(days, plot)

The Clos function returns a TSM, in which each column represents the close price for each stock in the global variable Stock. Missing data for individual stocks is returned as NaN (not a number).

5.6.2 High(days, plot)

The High function returns a TSM, in which each column represents the highest traded price for each stock in the global variable Stock. Missing data for individual stocks is returned as NaN (not a number).

5.6.3 Low(days, plot)

The Low function returns a TSM, in which each column represents the lowest traded price for each stock in the global variable Stock. Missing data for individual stocks is returned as NaN (not a number).

5.6.4 Volume(days, plot)

The Volume function returns a TSM, in which each column represents the traded volume (number of stocks) for each stock in the global variable Stock. Missing data for individual stocks is returned as NaN (not a number).

5.6.5 Trend1(days, plot) (also Trend2, Trend5 and Trend20)

Functions that return values identical to the Trendk function with the corresponding value of the k parameter.

5.6.6 Trendk(k, days, use, plot)

The k-step return $R_k^m(t)$ of a stock m with a price time series $Clos^m(t)$ is defined as:

$$R_k^m(t) = 100 \cdot \frac{Clos^m(t) - Clos^m(t-k)}{Clos^m(t-k)}. \tag{9}$$

By setting k at different numbers you obtain measures indicating how much the stock has increased since its value k days ago. The Trend functions can be used to compute return. The k-step trend $T_k(t)$ of a stock m is defined as:

$$T_k^m(t) = \frac{1}{k} \cdot R_k^m(t). \tag{10}$$

By setting k at different numbers you obtain measures indicating how much the stock has increased per day since its value k days ago.

The four functions Trend1, Trend2, Trend5, and Trend20 can be used to access 4 return measures for $k = 1, 2, 5$, and 20. They are computed and stored the first time any of the functions are called. The function Trendk accepts any value for the parameter k and does not cause any data to be precomputed and stored. Since the speed loss compared to the fixed functions (Trend1, Trend2, etc.) is insignificant in most cases, the Trendk function should be used in most cases.

The Trend functions all return a TSM, in which each column represents one particular stock in the global variable *Stocks*. The parameter *use* can be used to force the TSM to contain the trend for one specific stock in all columns. This can for example be useful in situations where one part of a Buy or Sell rule is the trend for a market index. Refer to Figures 5 , 6 and 26 for examples of this usage.

5.6.7 Rank1(days, plot) (also Rank2, Rank5, Rank20)

The k-step Ranks A_k^m for stocks $\{s_1, ..., s_N\}$ are computed by ranking the N stocks by the k-step returns R_k^m. The ranking orders are then normalized, so the stock with the lowest R_k gets rank -0.5 and the stock with the highest R_k gets rank 0.5. Therefore, the definition of the k-step Rank A_k^m for a stock s_m, belonging to a set of stocks $\{s_1, ..., s_N\}$, can be written as:

$$A_k^m(t) = \frac{order(\ R_k^m(t),\ \ \{R_k^1(t), ..., R_k^N(t)\}\ ) - 1}{N-1} - 0.5 \tag{11}$$

where the *order* function returns the ranking order of the first argument in the second argument, which is an ordered list. R_k^m is the k-step return (definition 9) computed for stock m.

The Rank for a stock is a measure seldom or never seen as input to trading systems. Since the ultimate goal is to "beat the market", i.e. to do better than

the average stock, it seems reasonable to have them available at least as potential inputs.

The four functions Rank1, Rank2, Rank5, and Rank20 can be used to access four rank measures for $k = 1, 2, 5$, and 20. They are computed and stored during market initialization to speed up the use during the simulation phase. The set $\{s_1, ..., s_N\}$ is all the stocks available for trading. Refer to Section 6.2 for examples with the rank functions.

5.6.8 Gvol2(days, plot) (also Gvol5, Gvol10 and Gvol20)

The gaussian volume $V_n(t)$ is defined as:

$$V_n(t) = (V(t) - m_V(t))/\sigma_V(t) \tag{12}$$

where $m_V(t)$ and $\sigma_V(t)$ are computed in a running window of length n:

$$m_V(t) = \frac{1}{n}\sum_{i=1}^{n} V(t-i) \text{ and} \tag{13}$$

$$\sigma_V = \sqrt{\frac{1}{n-1}\sum_{i=1}^{n}(V(t-i) - m_V(t))^2}. \tag{14}$$

V_n expresses the number of standard deviations by which the volume differs from its running mean. Therefore, it can be used as an indication of "abnormal" values, caused by a sudden change in market interest for a particular stock. That is the reason that the sums in Definition (13) and (14) do not include the current day's volume, i.e. $V(t)$. The four functions Gvol2, Gvol5, Gvol10, and Gvol20 can be used to access four Gaussian volumes V_n for $n = 2, 5, 10$, and 20. They are computed and stored during market initialization to speed up the use during the simulation phase.

5.6.9 Volat2(days, plot) (also Volat5, Volat10 and Volat20)

The Relative volatility $W_n(t)$ is here defined as:

$$W_n(t) = 100 \cdot \frac{\sigma_C(t)}{m_C(t)} \tag{15}$$

where $m_C(t)$ and $\sigma_C(t)$ are computed on the time series *Clos* in a running window of length n. I.e.:

$$m_C(t) = \frac{1}{n}\sum_{i=0}^{n-1} Clos(t-i) \tag{16}$$

and

$$\sigma_C(t) = \sqrt{\frac{1}{n-1}\sum_{i=0}^{n-1}(Clos(t-i) - m_C(t))^2}. \tag{17}$$

Definition (15) is recognized as the "coefficient of variation" or "relative standard deviation" in statistics. Relative volatility W_n is preferred to ordinary volatility estimation by the standard deviation, because it provides a measure that can be compared over longer time periods and also between stocks with totally different price levels.

The four functions Volat2, Volat5, Volat10, and Volat20 can be used to access four volatilities W_n for $n = 2, 5, 10$, and 20. They are computed and stored during market initialization to speed up the use during the simulation phase.

5.7 Indicator Functions

Indicator functions return a TSM with one column for each stock in the global variable *Stocks*. The functions can be used to compose complex trading rules either as user-defined functions or by manual input in the WASTA command window. Table 3 lists all predefined Indicator Functions. The plot argument is described in Section 5.5.2.

5.7.1 Profit and Loss

The system automatically keeps track of the change in price of each stock since it has been bought. The Profit is defined as:

$$\text{Profit}(t) = (\text{Clos(t)} \ / \ \text{BuyPrice} - 1) * 100. \tag{18}$$

The Loss is sometimes a more convenient measure and is defined as:

$$\text{Loss}(t) = (1 - \text{Clos(t)} \ / \ \text{BuyPrice}) * 100. \tag{19}$$

BuyPrice is the average buy price of each stock in a portfolio. Profit and Loss are not defined for stocks outside the portfolio.

The functions are of great use when defining trading rules. For example, a simple Sell rule may be defined as:

$$\text{Loss}{>}10 \ |\text{Profit}{>}20 \tag{20}$$

The Sell rule evaluates to an Indicator Vector with "1" for stocks where either Loss returned a value less than 10 or Profit returned a value greater than 20. A more complex function based on the profit and loss concept is Stoploss.

5.7.2 Stoploss(S1, S2, P1)

An often mentioned "trading rule to live by" is that of never allowing a profitable trade to turn into a loss. This idea is implemented in the function Stoploss. Each stock gets a stop-loss parameter of its own. This parameter is initialized to the value S1 when a stock is bought. It is changed to S2 when the Profit function registers more than P1% profit. Typical values for the parameters S1, S2 and P1 are 20%, -5%, and 10%. This results in the following behavior during trading:

1. The stock is sold immediately if the price drops by more than 20% below purchase price, thus cutting the maximum loss.

2. If the price ever rises by 10% above the purchase price, a consecutive drop in price to 5% above purchase price issues a sell signal. In this way, a profitable trade never turns into a loss.

5.7.3 Potential

This function computes the profit that a purchase of a stock would result in given a certain sell rule. The potential values are precomputed by the menu command "Compute Stock Potentials" in ASTA. The Sell Rule is given by the user and the potential profit for each day in the database is computed for all the stocks. These values are made available through the Potential function. The function value at time t is a measure of the potential profit of any Buy Rule that is executed at the same time t. Potential is therefore an alternative to the usual fixed horizon approach when evaluating Buy rules. It can be argued that this way of evaluating Buy rules is far more realistic than assuming that a stock is sold a fixed number of days after the purchase.

The Potential function at time t is defined as:

$$\text{Potential} = (\text{Clos}(t_1) \ / \ \text{Clos}(t) - 1) * 100 \tag{21}$$

where t_1 is the time that the Sell rule defined through the "Compute Stock Potentials" suggests as selling time.

Example 14 *Choose "Compute Stock Potentials" from the ASTA menu. In the dialog box enter:*

$$\textit{Sell rule for computation of potential:} \quad \textit{Profit>20} \mid \textit{Loss>10} \tag{22}$$

The database is now updated with values that can be accessed through the Potential function. The values may be saved in the database (Market.potential and Market.potdays) by clicking the Save button in WASTA and will in that case be immediately available next time ASTA is started with that database.

The computation can also be done in non interactive mode by calling the function *makepot.* E.g:

$$\text{makepot('Profit>10 |Loss>20 | Daysin>30 |Dnfract(1)>70',100)} \tag{23}$$

5.7.4 Active(v, minv, plot)

The function Action returns a binary value 0/1 depending on the value of the volatility for the stock.The parameters *volattyp* and *minvolat* control the computation:

volattyp	2, 5, 10 or 20. Causes the volatility functions Volat2, Volat5, Volat10 or Volat20 to be used
minvolat	Volatilities less than this value are assigned Active=0.

5.7.5 Upfract(k, days, plot)

The function Upfract computes the fraction of stocks in global *Stocks* that have Trendk(k)>0. The same value is returned for ALL stocks.

Example 15 *Upfract(1)*
Assume that global Stocks contains 8 stocks. The function call Upfract(1) returns an 8-element vector: [37.5 37.5 37.5 37.5 37.5 37.5 37.5 37.5] when computed at time T. This means that 37.5% of the 8 stocks have Trendk(1,T)>0.

5.7.6 Dnfract(k, days, plot)

The function Dnfract computes the fraction of stocks in global Stocks that have Trendk(k)<0. The same value is returned for ALL stocks.

Example 16 *Dnfract(10)*
Assume that global Stocks contains 8 stocks. The function call Dnfract(10) returns an 8-element vector: [37.5 37.5 37.5 37.5 37.5 37.5 37.5 37.5] when computed at time T. This means that 37.5% of the 8 stocks have Trendk(10,T)<0.

5.7.7 Daysin(plot)

The function *Daysin* computes the number of days since each stock was bought. This is normally computed inside a Sell rule and as usual, a TSM with one column for each stock in global *Stocks* is returned. If *Daysin* is computed inside a Buy rule, *Stocks* might contain stocks that are not in the trader's portfolio. In these cases, NaN is returned in these columns in the returned TSM.

5.7.8 Nstocks(plot)

The function Nstocks computes the number of stocks in the portfolio for each stock. It is normally computed inside a Sell rule and, as usual, a TSM with one column for each stock in global *Stocks* is returned

5.7.9 Closepos(k, days, plot)

Returns a TSM with close price position within the High-Low range, for all stocks in global Stocks. Filtering of High and Low is done when parameter k>0.

Algorithm:

$$\begin{aligned} &ClosePos = (Close - mavL) \;/\; m(mavH - mavL) \\ &mavL = Mav('Low', k, days, 0) \\ &mavH = Mav('High', k, days, 0) \end{aligned} \tag{24}$$

Each row in the returned TSM corresponds to one day. The day numbers should be given in parameter days. If k>1, moving averages with length k are computed for High and Low, k=0 (default) gives no filtering.

Example 17 $ClosePos(5, T)$
Returns a row vector with today's values for the stocks in global Stocks.

Example 18 $ClosePos(5, T-9:T)$
Returns a 10-row matrix with the 10 latest values for the stocks in global Stocks.

5.7.10 CloseRuns(days, plot)

Returns run length for the one-day returns. I.e.: Number of consecutive days with same sign for the 1-day return. CloseRuns returns negative values for consecutive negative returns and positive values for consecutive positive returns.

Example 19 *Let x=CloseRun(T). x is now a one-row TSM where each column gives the run length for the corresponding stock in the global variable Stocks.* $x(k) = -3$ *means that* $R_k^1(T) < 0$, $R_k^1(T-1) < 0$ *and* $R_k^1(T-2) < 0$. $x(k) = 2$ *means that* $R_k^1(T) > 0$ *and* $R_k^1(T-1) > 0$.

5.7.11 Underlow(L)

Returns a binary one-row TSM. The function looks at the minimum of the Low price in a window L days back. Underlow returns a logical 1 if the close price crosses this computed minimum value in a certain fashion. The algorithm has two variants depending on the sign of L:

```
C = Clos(T)
Lm1 = Low(T-1)
M = min(Low(T-abs(L):T-1))
if L>0 then
   Underlow = C<M & Clos(T-1)>M & Lm1>M
else
   Underlow = C<M & Lm1~=M
end
```
(25)

5.7.12 Overhigh(L, plot)

Returns a binary one-row TSM. The function looks at the maximum of the High price in a window L days back. Overhigh returns a logical 1 if the close price crosses this computed maximum value in a certain fashion. The algorithm has two variants depending on the sign of L:

```
C = Clos(T)
Hm1 = High(T-1)
M = max(High(T-abs(L):T-1))
if L>0 then
   OverHigh = C>M & Clos(T-1)<M & Hm1<M
else
   OverHigh = C>M & Hm1<M
end
```
(26)

5.7.13 Days(dys)

Returns a binary one-row TSM with one column for each stock in global *Stocks*. *Days* return 1 in all columns iff the current day number is found in the vector *dys*. Day numbers are coded as follows: 1:Monday 2:Tuesday...

Example 20 *Days([1 2]) returns 1 in all columns iff the current day T is Monday or Tuesday.*

5.7.14 Stoch(K, Ks, S, plot)

The standard technical indicator Stochastics. The function returns -1, 0 or +1 to facilitate usage in both Buy rules and Sell rules. For more information refer to the program code in the file *stoch.m*.

5.7.15 Macd(K, D, S, plot)

The standard technical indicator MACD. The function returns -1, 0 or +1 to facilitate usage in both Buy rules and Sell rules. For more information refer to the program code in the file *macd.m*

5.7.16 Hurst(x, k, days, plot)

This function computes the Hurst coefficient for the TSM generated by the function x. The function does the computation in a k days long window backwards. Default value for k is 2000. For more information refer to the program code in the file *hurst.m*.

Example 21 *Hurst('Trend1',2000,T,PLOT)*

5.7.17 OBV(n, plot)

The technical indicator *On Balance Volume*. Adds or subtracts traded volume depending on the sign of the 1-day return for Close. Performed in an n day long window up to time T. OBV returns Returns a one-row TSM with the accumulated volume figures for each stock. Algorithm:

$$\begin{aligned} c &= sign(Clos(T-n+1:T) - Clos(T-n:T-1)) \\ s &= Volume(T-n+1:T) \\ OBV &= sum(c.*s) \end{aligned} \tag{27}$$

5.7.18 Rsi(k, s, b, plot)

The standard technical indicator RSI. The function returns -1, 0 or +1 to facilitate usage in both Buy rules and Sell rules. For more information refer to the program code in the file *rsi.m*.

5.7.19 Keyrev(n, plot)

The technical indicator Key Reversal. Returns a binary one-row TSM. Algorithm:

```
Keyrev = Clos(T-1)>Mav('Clos',n,T-1) and
         High(T)>=Maxx('High',n,T-1) and
         Low(T)<Low(T-1) and
         Clos(T)<Clos(T-1)
```
(28)

5.8 Operator Functions

Operator functions perform an operation on one or several TSMs and return another TSM with transformed values. All Operator Functions return a TSM with one row for each day in the *days* parameter, and one column for each stock in the global *Stocks* variable. The parameters x, x1, and x2 can be either a TSM or a string with the name of a function returning a TSM.
The parameter days:

- If x is a TSM, the *days* parameter should be row numbers in the matrix. The default value for days is the last row in the matrix.
- If x is a string, the *days* parameter should be a vector with day numbers, such as T, T-1, etc. The default value for days is T.
  Note:

The Operator Function performs its operation on each of the rows in the x matrix and returns a TSM with the result for each row. In the following description of Operator Functions, the d variable is implicitly assumed to run through the full range defined by the *days* parameter. Each value on d results in one row in the output TSM. Table 4 lists all predefined Operator Functions in ASTA.

5.8.1 Minn(x, L1, days, plot)

Computes the minimum value in an L1 days long window backwards.
If x is a string, $Minn$ is defined as:
$Minn \ = \ min(x[d - L1 + 1 : d])$
where the square brackets denote function invocation of x.
$Minn('Clos', 4, T)$.
Generates a 1-row TSM: $min(Clos(T - 3 : T))$.

Example 22 $Minn('Clos', 4, T - 1 : T)$.
Generates a 2-row TSM: $\begin{pmatrix} min(Clos(T - 4 : T - 1)) \\ min(Clos(T - 3 : T)) \end{pmatrix}$.

If x is a not a string, it should be a TSM. $Minn$ is in this case defined as:
$Minn \ = \ min(x(d - L1 + 1 : d, :))$.

Example 23 $Minn(Z, 4, 10)$ *where Z is a 10-row-long TSM.*
Generates a 1-row TSM: $min(Z(7 : 10\ , :))$.

Example 24 $Minn(Z, 4, 9 : 10)$ *where Z is a 10-row-long TSM.*
Generates a 2-row TSM: $\begin{pmatrix} min(Z(6 : 9\ , :)) \\ min(Z(7 : 10\ , :)) \end{pmatrix}$.

5.8.2 Maxx(x, L1, days, plot)

Computes the maximum value in an L1-day-long window backwards. If x is a string, $Maxx$ is defined as:
$Maxx = max(x[d - L1 + 1 : d])$
where the square brackets denote function invocation of x.
If x is a not a string, it should be a TSM. $Maxx$ is in this case defined as:
$Maxx = max(x(d - L1 + 1 : d, :))$.
Refer to $Minn$ for examples.

5.8.3 Mav(x, L1, days, plot)

Computes the moving average value in an L1-days-long window backwards. If x is a string, Mav is defined as:
$Mav = mean(x[d - L1 + 1 : d])$
where the square brackets denote function invocation of x.
If x is a not a string, it should be a TSM. Mav is in this case defined as:
$Mav = mean(x(d - L1 + 1 : d, :))$.
Refer to $Minn$ for examples.

5.8.4 Stdd(x, L1, days, plot)

Computes the standard deviation in an L1-days-long window backwards. If x is a string, $Stdd$ is defined as:
$Stdd = std(x[d - L1 + 1 : d])$
where the square brackets denote function invocation of x.
If x is a not a string, it should be a TSM. $Stdd$ is in this case defined as:
$Stdd = std(d - L1 + 1 : d, :))$.
Refer to $Minn$ for examples.

5.8.5 Mavx(x, L1, x2, L2, A, days, plot)

Crossings between moving averages of various signals are often used in the definition of traditional technical indicators. The Mavx function supplies a convenient way to implement the detection of such crossings. The optional argument A makes it possible to specify the minimum angle for the crossing. Crossings at angles lower than the specified one do not generate a logical "1" at that point. Mavx can be used to detect both crossings from below and above, by changing the order of arguments L1 and L2.

Example 25 $Mavx('Clos', 2, 'Clos', 20, 45)$.
Returns a binary Indicator Vector with "1" iff a 2-day moving average of ***Clos*** *crosses a 20-day moving average of* ***Clos*** *from below in an intersection angle no less than 45 degrees. The computation is performed "today" since the default value for the* ***days*** *parameter is* ***T****.*

5.8.6 Repeats(x, n, plot)

Returns the number of times x was evaluated to "1" during the last n days. x should be a string that evaluates to a 1-row TSM.

Example 26 *Repeats*$('Clos(T) > Clos(T-1)', 5, PLOT) > 2$
Returns "1" iff the Close price has increased since the previous day more than 2 times during the period $T, T-1, T-2, T-3, T-4$.

Example 27 *Repeats*$('Trendk(5) > Trendk(10)', 10) > 3$
Returns "1" iff the 5-day trend has been greater than the 10-day trend more than 3 times during the period $T, T-1, ..., T-9$.

5.9 Utility Functions

Utility functions are support functions that may be of use primarily when writing new indicators and algorithms. They are normally not used as such in Buy and Sell rules, because they do not return Time Series Matrices. Table 5 lists all predefined Indicator Functions in ASTA.

5.9.1 Use(s)

The function $Use(s)$ replaces the global *Stocks* with a vector of the same size but with every element=s. In this way all subsequent functions that use the global *Stocks* operate instead on the single stock s. This can be utilized to construct expressions that depend on one single security, such as a stock index. The Use command is in effect until reset by another call with an empty s argument: $Use('')$. The Use function is also automatically reset at the next time step in the simulation. The parameter s can be either a string (e.g.'SXGEN') or a predefined symbol (e.g: SXGEN). Recursive calls to Use are not allowed.

Example 28 $Use('SXGEN');\ x = Gvol10 > 1; Use(''); x\ \&\ Trendk(10) > 2$
The expression used in a Buy rule gives a buy signal for stock k if the 10-day Gaussian Volume for Generalindex (SXGEN) is greater than 1 at the same time the 10-trend for stock k is greater than 2% per day.

Note: function $Trendk$ has a built-in option of the same kind as function Use .

6 More Examples

This section provides some mixed examples of how ASTA can be used for both trading simulations and fixed horizon predictions.

6.1 Viewing the Trader as an Objective Function

The obvious wish to maximize the profit of a trading system can be tackled in two ways:

1. Parameterizing the Buy rule and Sell rule, i.e. introducing parameters within the rules. Example:

   $Buy\ rule\ = 'Clos(T) > Maxx('High', Nhigh, T-1)'$

   $Sell\ rule\ = 'Loss > L \mid (Profit > P\ \&\ Clos(T) < Clos(T-1))'$

The function Maxx returns the maximum time series value in the interval $[T - Nhigh, T - 1]$. It is now possible to optimize the profit P with respect to the parameters $Nhigh, L$ and P.

2. Viewing the Buy rule and Sell rule as symbolic expressions. The optimization then turns into a search problem, most naturally implemented in a genetic framework or using Inductive Logic Programming.

An optimization of one of the parameters in the Stochastics indicator serves as an example for approach 1. We set up an optimization with buy and sell rules according to:

Buy rule:	$Stoch(30, 3, 3, Sellevel, 80)$
Sell rule:	$Stoch(30, 3, 3, Sellevel, 80)$

(29)

The excess profit P can be optimized now with respect to $Sellevel$. The graphs in Figure 13 are automatically generated by the "Sweep" function in the ASTA system (refer to Section 4.12 for details). The name of the parameter is given in the "Parameter" text box and the range in the "Values" text box.

It must be emphasized that the optimization of the performance is a multi-dimensional parameter estimation problem. The graphs presented show the profit P as a function of *one* of the involved parameters, whereas the rest of the parameters are fixed. The main purpose is to illustrate the possibilities and problems involved, even in a 1-dimensional optimization. For a multi-dimensional analysis refer to [7].

From the summary graph (top left in Figure 13) over the entire training period, we can deduce that the highest profit is achieved for a $Sellevel$ somewhere around 35. However, a look at the data at a higher resolution reveals a more complicated situation. In the bottom left diagram the same relation is plotted with one curve for each year in the training dataset. It is clear that the mean profit is totally dominated by the results from one of the years (1993).

From these curves we can learn at least two important points:
1. The spread between individual years is very high.
2. The location of the maximum is not obvious.
This behavior of data is typical for most variables. The profit cannot be easily described as a function of measurable variables without introducing a dominant noise term in the function. Let us therefore view the annual excess profit as a stochastic variable $P(\theta)$, where θ stands for one particular setting of the parameters that affect the profit. In the shown example, θ is the $Sellevel$ parameter. $P(\theta)$ has been sampled once per year during the 11 years in the dataset:

$$\{P_1(\theta), P_2(\theta), P_3(\theta), P_4(\theta), P_5(\theta), P_6(\theta), P_7(\theta), P_8(\theta), P_9(\theta), P_{10}(\theta), P_{11}(\theta)\}. \quad (30)$$

Viewed this way, the task of tuning the parameter θ to find the "maximum" profit P is not well defined. P is a stochastic function and consequently has no "maximum". It has a probability distribution with an expectation value E and a variance V . It is important to realize that tuning θ in order to maximize $E[P(\theta)]$ is just one of the available options. Maximizing $E[P(\theta)]$ provides the highest mean performance. Another possibility is to maximize the lower limit of a confidence

interval. Since the risk factor is always a major concern in investments, and since the spread between individual years obviously can be very high, this sounds like a promising idea. A lower limit P_{low} for a confidence interval could be defined as:

$$P_{low} = E[P(\theta)] - \sqrt{V[P(\theta)]}, \tag{31}$$

where $V[P(\theta)]$ is the variance of the stochastic variable $P(\theta)$. Yet another possibility is to use the Sharpe Ratio SR, which expresses the excess return in units of its standard deviation as:

$$SR = \frac{E[P(\theta)]}{\sqrt{V[P(\theta)]}}. \tag{32}$$

The Sharpe Ratio is normally used to evaluate the performance of a trading strategy (Sharpe [9, 10]). However, Choey and Weigend [3] propose to use the Sharpe Ratio as an objective function in portfolio optimization and derive a learning algorithm for artificial neural networks. The Sharpe Ratio should be as high as possible for an optimal trading algorithm.

Due to the high noise level in the data, we have added a modified Sharpe Ratio, where the outliers have been removed. The largest and smallest $P_i(\theta)$ in Expression (30) have been removed before taking the expected value and standard deviation for each θ. The unmodified Sharpe Ratio is shown in the top-right diagram, and the one with outliers removed in the mid-right diagram. As we can see, the one with the outliers removed clearly reveals a maximum at around 20-25, followed by a clear decline. The unmodified Sharpe Ratio shows no such pattern.

6.2 Trading with Ranks

We continue with another example where the trading rules are based on a rank concept for the stocks. The Rank concept was briefly introduced in section 33. The k-step rank A_k^m for a stock s_m in the set $\{s_1, ..., s_N\}$ is computed by ranking the N stocks in the order of the k-step returns R_k. The ranking orders are then normalized so the stock with the lowest R_k gets rank -0.5 and the stock with the highest R_k gets rank 0.5. The definition of the k-step rank A_k^m for a stock m belonging to a set of stocks $\{s_1, ..., s_N\}$, can thus be written as

$$A_k^m(t) = \frac{order(R_k^m(t), \{R_k^1(t), ..., R_k^N(t)\}) - 1}{N - 1} - 0.5 \tag{33}$$

where the *order* function returns the ranking order of the first argument in the second argument, which is an ordered list. Thus, *order* returns an integer between 1 and N. R_k^m stands for the k-step returns computed for stock m. The scaling between -0.5 and $+0.5$ assigns the stock that has the median value on R_k the rank 0. A positive rank A_k^m means that stock m performs better than this median stock, and a negative rank means that it performs worse. This new measure gives an indication of how each individual stock has developed relatively to the other stocks, viewed on a time scale set by the value of k. The scaling around *zero* is also convenient when defining a prediction task for the rank. It is clear that an ability to identify,

at time t, a stock m, for which $A_k^m(t+h) > 0$ means an opportunity to make profit in the same way as identifying a stock, for which $R_k(t+h) > 0$. And even better, since such a method is guaranteed to do better than the average stock. The hit rate for the predictions can be defined as the fraction of times, for which the sign of the predicted rank $A_k^m(t+h)$ is correct. A value greater than 50% means that such a trading strategy would do better than the average stock. The statistical properties of the rank measure have been investigated empirically in more detail in Hellström [4]. In the following we use the observation that the ranks for a set of stocks exhibit a negative first lag in the auto-correlation function, signifying a mean reverting behavior for the rank measure. The database contains 207 stocks from the Swedish stock market between the beginning of 1987 and the end of 1997. We set up the following buy and sell rules:

Buy rule:	Rank5(T)<-0.48
Sell rule:	Rank5(T)>0.48

(34)

where the ASTA function *Rank*5 for a stock m is computed as $A_5^m(T)$ in definition 33. The buy rule suggests buying if the stock is among the worst performing 2% viewed over the last 5 days. The sell rule suggests selling if the stock is among the best performing 2%. Quite contrary to what is often considered correct, we buy the losers and sell them when they are among the winners. So how does such a strategy perform? In Figures 16 and 17 we see the results of 10 trading simulations with random acceptance of the buy signals. The strategy makes 45% annual profit while the index in average increases by 16%. We also observe that the trading strategy is better than the index 8 out of 11 years. The reverse strategy, buying the winners and selling the losers, has also been tested. The results are presented in Figures 18 and 19 which show that the annual profit is only 5% and is better than the index only 1 out of 11 years.. This trading strategy actually produces a 14% loss while the index goes up by 310% during the same time period.

In Figures 20 and 21 the selling rule from Figure 16 is replaced by the *Profit* and *Stoploss* functions (described in section 5.7.2). In this way the effect of using ranks as a pure buying rule can be analyzed. The buy and sell rules are:

Buy rule:	$Rank5(T) < -0.48$
Sell rule:	$Stoploss(15, -5, 5) \mid Profit > 40$

(35)

As we can see in the Figures 20 and 21, the trading results are pretty good. The strategy does better than index 8 out of 11 years as before and the mean annual profit has increased to 54.8%.

6.3 Looking at the Relation between Today's High and Close

A few technical indicators like *Stochastics* look at the High and Low prices as well the the Close price for the day. The following example implements a simple heuristic buy rule where the Close price should cross the highest High for the last 10 days at the same time as the highest High was not reached the previous day. This is

implemented in the following buy rule:

$$\begin{aligned} L &= 10; M = max(High(T-L:T-1)); \\ Clos(T) &> M \ \& \ Clos(T-1) < M \ \& \ High(T-1) < M. \end{aligned}$$

The expression is written in free matlab syntax and input in the Buy rule field in the ASTA command window as shown in Figure 22.

6.4 Fixed Horizon Predictions of Individual Stocks

In Section 3.6 the possibilites to perform fixed horizon predictions with ASTA were illustrated with examples predicting the Swedish Generalindex. In this section some examples of predictions of individual stock time series are presented. As opposed to most examples given earlier in this report, the data is taken from the American stock market between 1987 and 1997. 30 stocks from the Dow Jones index are included in the database.

The first example in Figure 25 shows a basic 1-day prediction of returns for 30 stocks. The entity to be predicted is input in the *Predict* field: $sign(Trendk(1, T+1))$. The predictor is input in the *Predictor* field: $sign(Trendk(1, T))$. When the simulation is started the Predict and Predictor fields are evaluated each time the Buy rule evaluates to *True* or, if no Buy rule is given, at ALL time steps. The latter is the case in the shown example. The summarized results are shown in the lower pane. The complete results stock by stock are written to the Matlab command window and can be inspected there. However, the summary is most often enough for evaluation of the prediction results. A description of the presented measures can be found in Section 3.6.

The *Predict* field typically has the same layout as above: $sign(Trendk(1, T+1))$. By careful design of the *Predictor* field, the ability to predict both rises and declines in the stock prices can be analyzed in one single run. The *+hitrate* gives the probability that a predicted increase in a stock's price really is followed by an increase. Similarly, the *-hitrate* gives the probability that a predicted decrease in a stock's price really is followed by a decrease.
Two lines with figures are presented in the summary: *Mean Values* and *Total Means.* The first line gives the average time-averaged performance for the included stocks. The second line gives mean values computed over all predictions at once. The first line is hence susceptible to cases where some stocks occur less often in the generated trades (either due to missing data or due to differences in stocks inclination to generate Buy signals). However, most often the two lines show similar performance values.

In the shown example in Figure 25 the performance rather confirms the random-walk nature of the stock prices. The overall hitrate is 50.26% computed over 70203 predictions. This can be compared to the performace of the *naive-increase* predictor (set the Predictor field to : 1). This predictor has a hit rate of 50.58% where the excess 0.58% is due to the general positive trend in the stock market.

The next exampe, shown in Figure 26, extends the predictor with a term

$$Trendk(1, T, UNIFORM).$$

UNIFORM is a predefined symbol (each stock database has a different set of predefined symbols) corresponding to an equally weighted index. By adding $UNIFORM$

as a third parameter to the $Trendk$ function, $Trendk$ does not look at each individual stock but instead returns the 1-day trend for this uniform index.The complete Predict expression $Trendk(1,T) > 1$ & $Trendk(1,T,UNIFORM) > 1$ predicts a rise if a stock's 1-day return is at least 1% and at the same time the uniform index also has gone up by at least 1% since the previous day. Note that the Predict expression only evaluates to 0 or 1 as opposed to the previous example where -1 was also returned by the *sign* function. The presented performance therefore contains no data for $-hitrate$. i.e for predicted declines. As we can see, the hit rate for predicted rises, $+hitrate$, goes up to 52.32% in this case. Also note that the number of produced predictions has gone down to 5793, which affects both the statistical significance of the conclusions and often also the practical usability for the prediction method. The results stock by stock are as usual written to the Matlab command window as shown in Figure 27.

7 Acknowledgements

Converting raw stock data for serious data analysis is neither trivial nor simple. The Swedish database supplied with the ASTA system has been converted, adapted, and cleaned up by Kenneth Holmström, from Mälardalens Högskola, Västerås, Sweden. The development of ASTA has also benefited in general from the many fruitful discussions I have had over the years with Kenneth Holmström on the subject.

References

[1] A. Atiya. Design of time-variable stop losses and profit objectives using neural networks. In A. S. Weigend, Y. S. Abu-Mostafa, and A.-P. N. Refenes, editors, *Decision Technologies for Financial Engineering (Proceedings of the Fourth International Conference on Neural Networks in the Capital Markets, NNCM-96)*, pages 76–83, Singapore, 1997. World Scientific.

[2] Y. Bengio. Training a neural network with a financial criterion rather than a prediction criterion. In A. S. Weigend, Y. S. Abu-Mostafa, and A.-P. N. Refenes, editors, *Decision Technologies for Financial Engineering (Proceedings of the Fourth International Conference on Neural Networks in the Capital Markets, NNCM-96)*, pages 36–48, Singapore, 1997. World Scientific.

[3] M. Choey and A. S. Weigend. Nonlinear trading models through Sharpe Ratio maximization. In A. S. Weigend, Y. S. Abu-Mostafa, and A.-P. N. Refenes, editors, *Decision Technologies for Financial Engineering (Proceedings of the Fourth International Conference on Neural Networks in the Capital Markets, NNCM-96)*, pages 3–22, Singapore, 1997. World Scientific.

[4] T. Hellström. *A Random Walk through the Stock Market.* Licentiate thesis, Umeå University, Umeå Sweden, 1998.

[5] T. Hellström. ASTA - a Tool for Development of Stock Prediction Algorithms. *Theory of Stochastic Processes*, 5(21)(1-2):22–31, 1999.

[6] T. Hellström. Data Snooping in the Sstock Market. *Theory of Stochastic Processes*, 5(21)(1-2):32–50, 1999.

[7] T. Hellström and K. Holmström. Parameter Tuning in Trading Algorithms using ASTA. In Y. S. Abu-Mostafa, B. LeBaron, A. W. Lo, and A. S. Weigend, editors, *Computational Finance – Proceedings of the Sixth International Conference, Leonard N. Stern School of Business, January 1999*, Cambridge, MA, 1999. MIT Press.

[8] J. E. Moody and L. Z. Wu. Optimization of trading systems and portfolios. In A. S. Weigend, Y. S. Abu-Mostafa, and A.-P. N. Refenes, editors, *Decision Technologies for Financial Engineering (Proceedings of the Fourth International Conference on Neural Networks in the Capital Markets, NNCM-96)*, pages 23–35, Singapore, 1997. World Scientific.

[9] W. F. Sharpe. Mutual fund performance. *Journal of Buisness*, pages 119–138, January 1966.

[10] W. F. Sharpe. The Sharpe Ratio. *Journal of Portfolio Management*, 21:49–58, 1994.

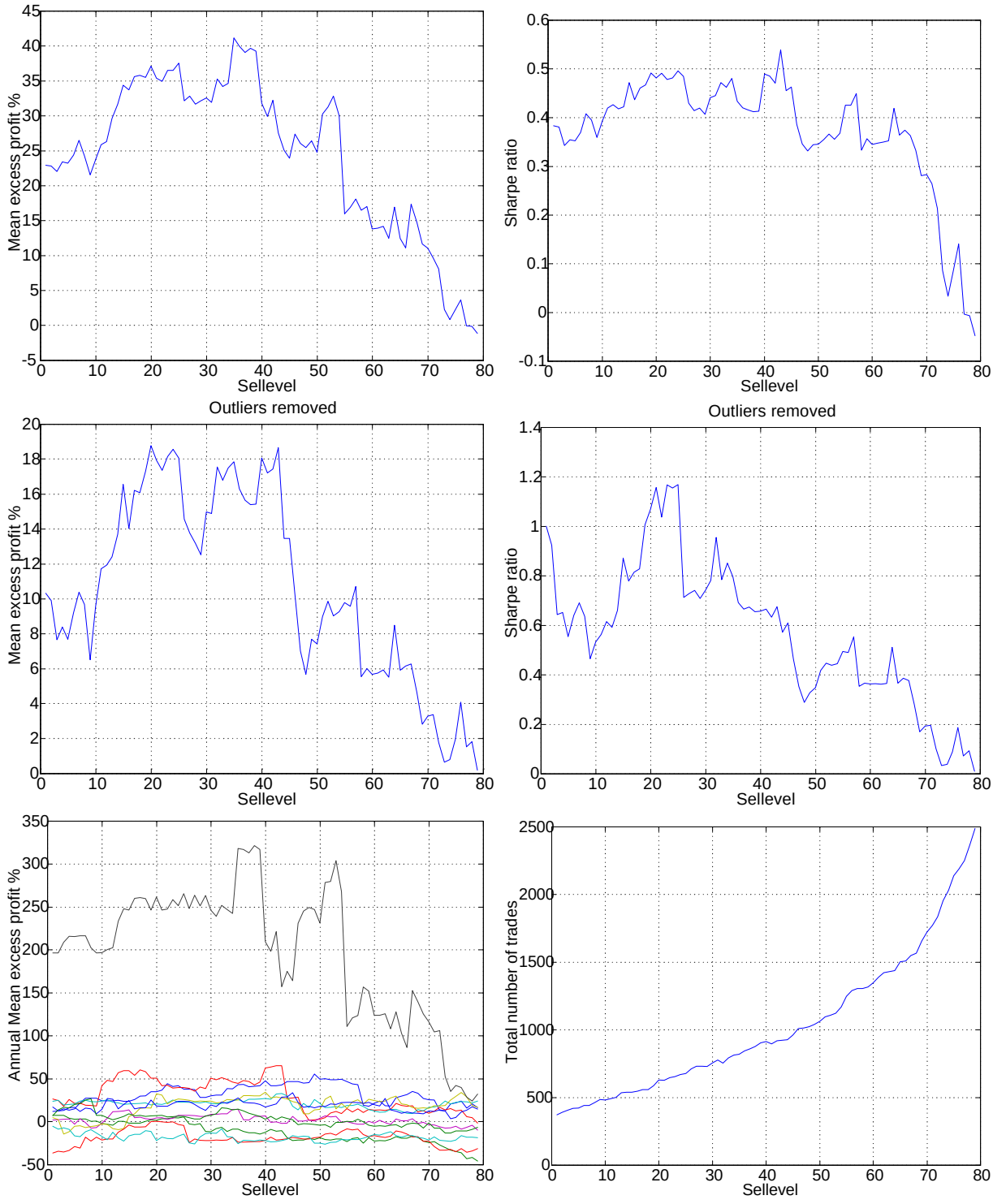


Figure 13: Trading results as a function of the Sellevel parameter. Stocks: SXG. Years:87-97. Buy rule: Stoch(30,3,3,Sellevel,80)>0. Sell rule: Stoch(30,3,3,Sellevel,80)<0

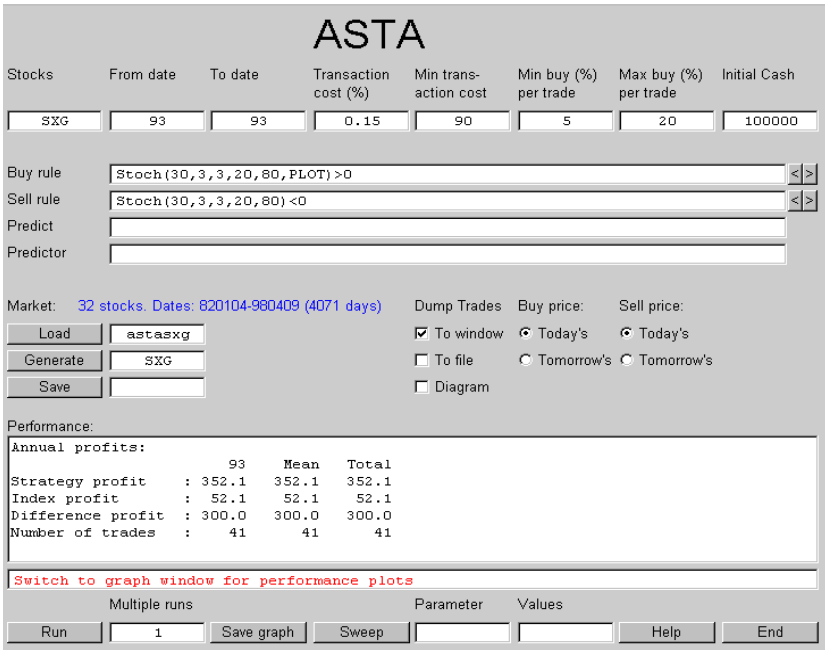


Figure 14: ASTA command window with Stochastic buy and sell rules. The PLOT option in the Buy rule generates separate plots with signals for each stock. The check box "Dump trades to window" generates a list of all trades in the Matlab command window.

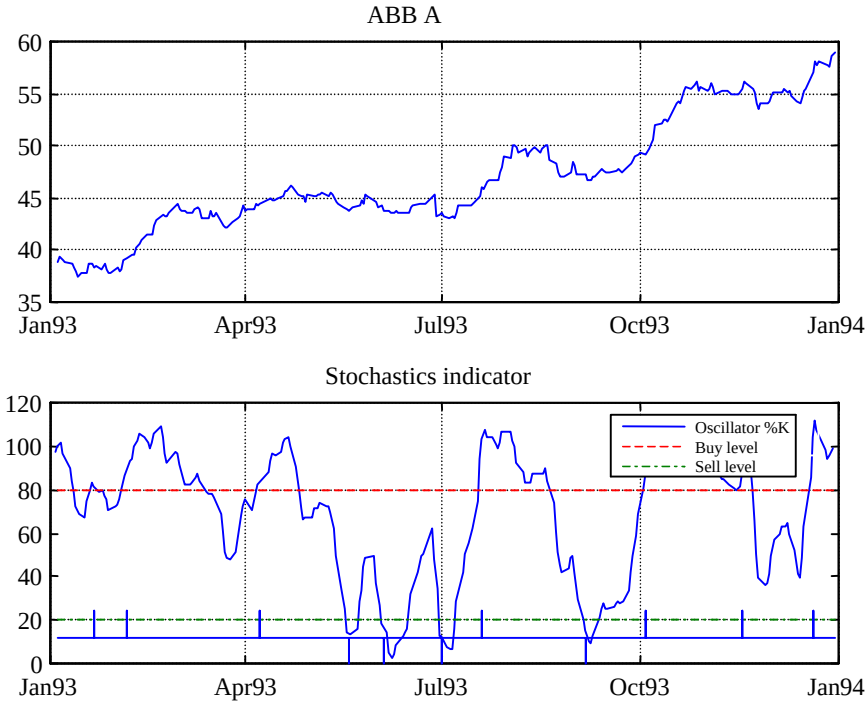


Figure 15: Stochastics buy and sell signals for the ABB stock. Upgoing bar denotes a buy signal and downgoing bar denotes a sell signal.

Function	Parameters	Description
Profit	-	% increase in **Close** since buy of stock
Loss	-	% decrease in **Close** since buy of stock
Potential	**plot**	Computed potential for stock
Active	**v, minv, plot**	Activity measure based on volatility
Upfract	**k, days, plot**	Fraction (%) of stocks with **Trendk(k)**>0
Dnfract	**k, days, plot**	Fraction (%) of stocks with **Trendk(k)**<0
Daysin	**plot**	Number of trading days since last buy
Nstocks	**plot**	Number of stocks in portfolio (each stock separately)
ClosePos	**k, days, plot**	Close price position within the High-Low range
CloseRuns	**days, plot**	Number of consequtive days with same sign for the 1-day return
Stoploss	**S1, S2, P1**	Implements a dynamic stop-loss function: Stop-loss is initially set to **S1** If **Profit**>**P1**, change the stop-loss to **S2**, **Stoploss** returns "1" iff the **Loss** is less than the dynamically set stop-loss. This function keeps track of each stock separately.
Firsttime		1 iff current date equals the first trading day in the current simulation
Underlow	**L**	"1" iff **Close** crosses a **L** day min of **Low** from above
Overhigh	**L,plot**	"1" iff **Close** crosses a **L** day max of **High** from below and **High(T-1)** is less than the same max
Days	**dys**	"1" iff day number is in vector **dys** (1:Monday...7:Sunday)
Stoch	**K,Ks,S,plot**	The technical indicator Stochastics.
Macd	**K,D,S,plot**	The technical indicator MACD
Hurst	**x,k,days,plot**	Hurst parameter for **x**
OBV	**n, plot**	The technical indicator OBV.
Rsi	**k, s, b, plot**	The technical indicator RSI.
Keyrev	**n, plot**	Bearish Key reversal indicator.

Table 3: Predefined indicator functions

Function	Parameters	Description
Minn	**x, L1, days, plot**	**L1**-day-min value in the rows of **x**
Maxx	**x, L1, days, plot**	**L1**-day-max value in the rows of **x**
Mav	**x, L1, days, plot**	**L1**-day-moving av. for columns in **x**
Stdd	**x, L1, days, plot**	**L1**-day-st.deviation for columns in **x**
Mavx	**x1,L1,x2,L2,A, days,plot**	"1" iff **Mav(x1,L1)** crosses **Mav(x2,L2)** from below. The **A** argument (optional) is the min required angle for the crossing.
Repeats	**x, n, plot**	Number of times x was evaluated to "1" during the last n days

Table 4: Predefined operator functions

Firsttime		Returns 1 iff current date equals the first trading day in the simulation. Otherwise 0.
Use	**s**	Causes the stock **s** to be used instead of the global variable *Stocks*
Secname	**s**	String with the name of stock **s** which for example could be the numbers in the global variable *Stocks*

Table 5: Predefined utility functions

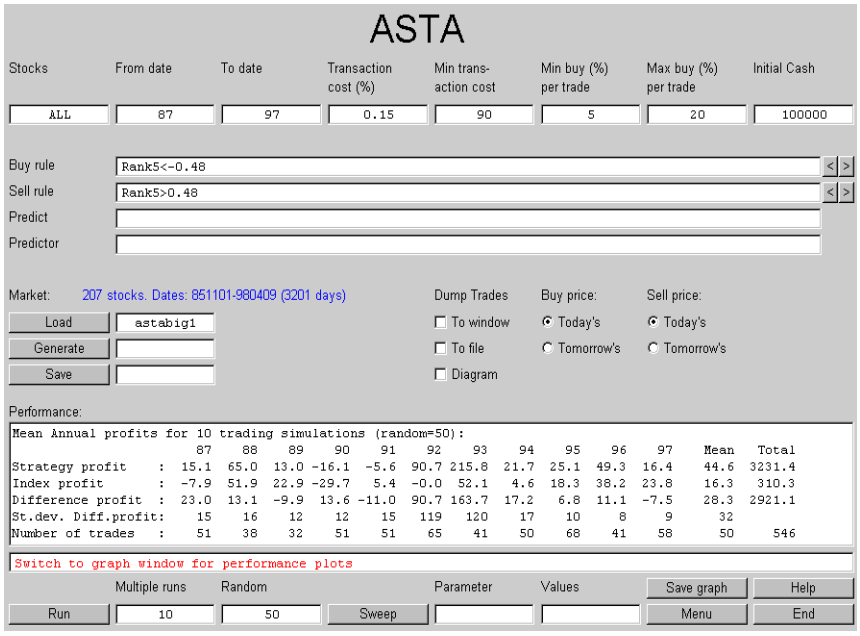


Figure 16: ASTA command window with the Rank function as Buy and Sell rules: Buy the loosers and Sell the winners.

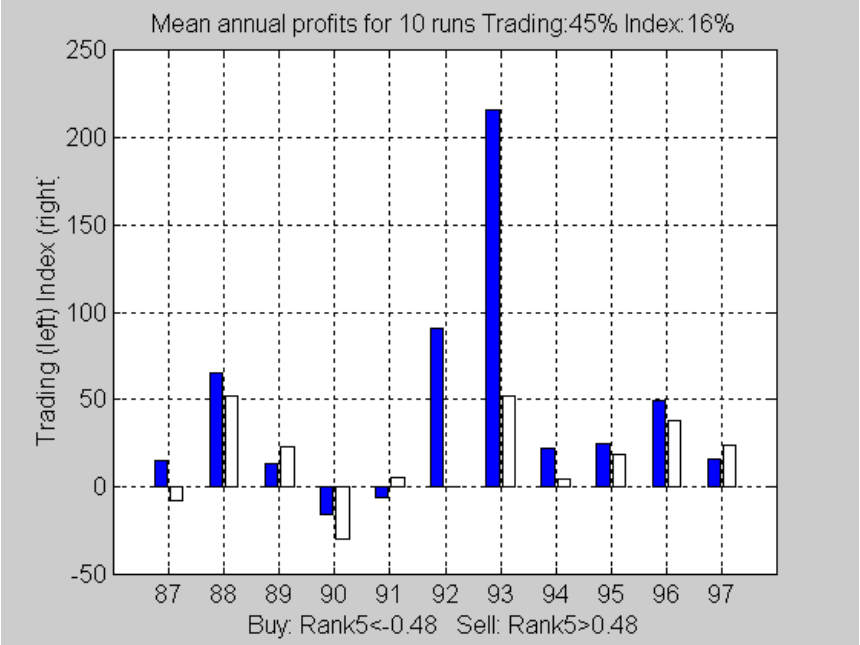


Figure 17: Trading results from the Rank-based trading rules in Figure 16

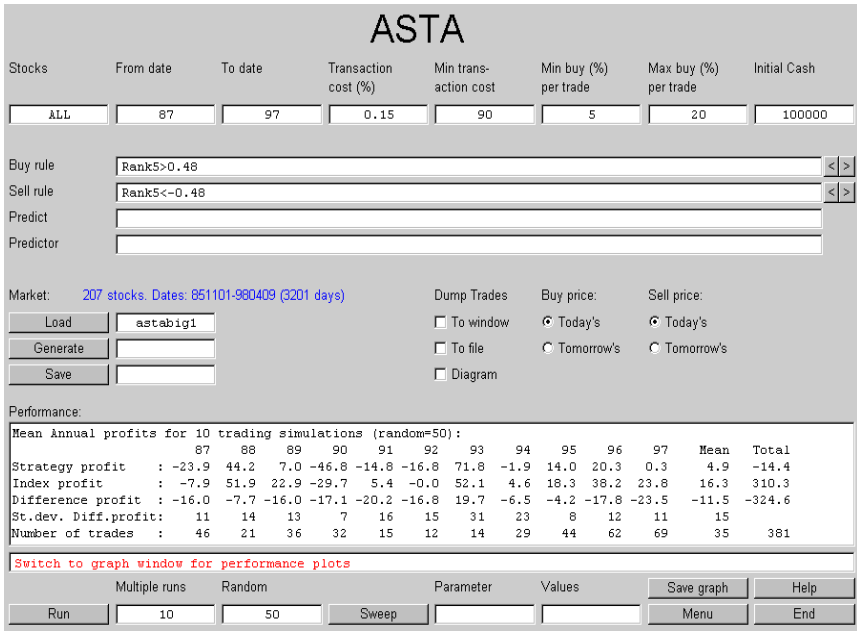


Figure 18: ASTA command window with reversed Rank trading rules: Buy the winners and Sell the losers.

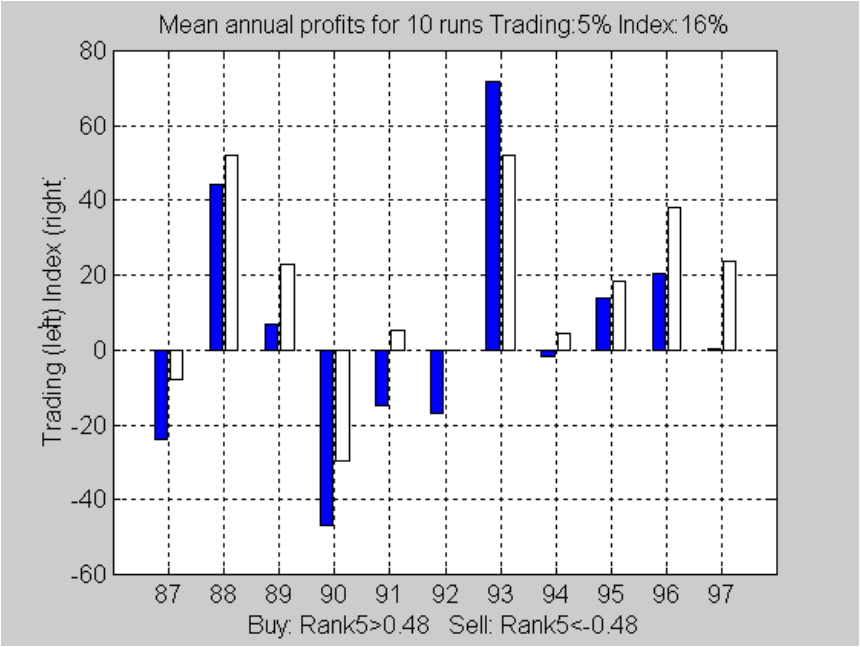


Figure 19: Trading results from the Rank-based trading rules in Figure 18

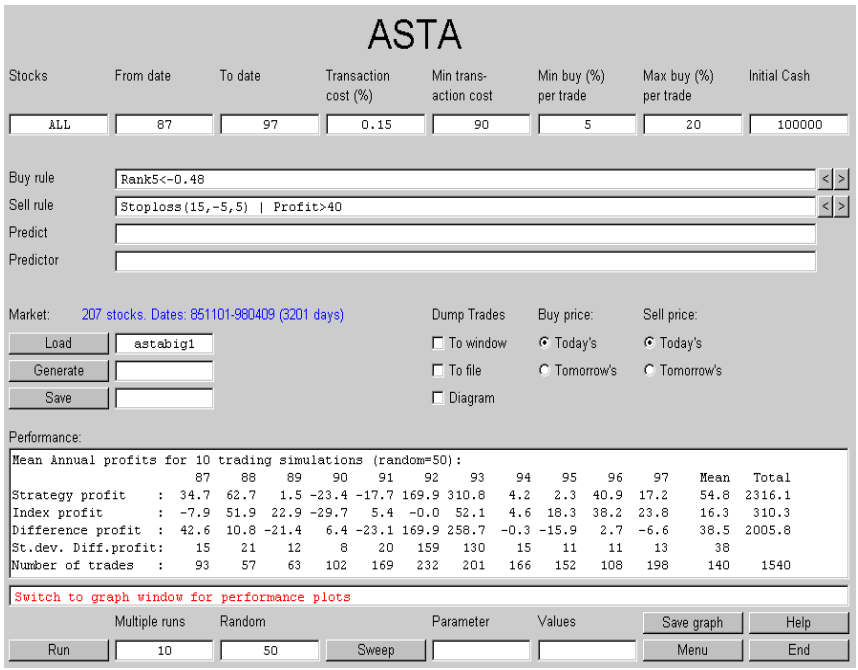


Figure 20: Another Rank-based trading. The Sell rule is based on the Stoploss function, which adapts the level for stop loss as the price goes up for a stock. A Sell is also issued if the Profit is 40% or more.

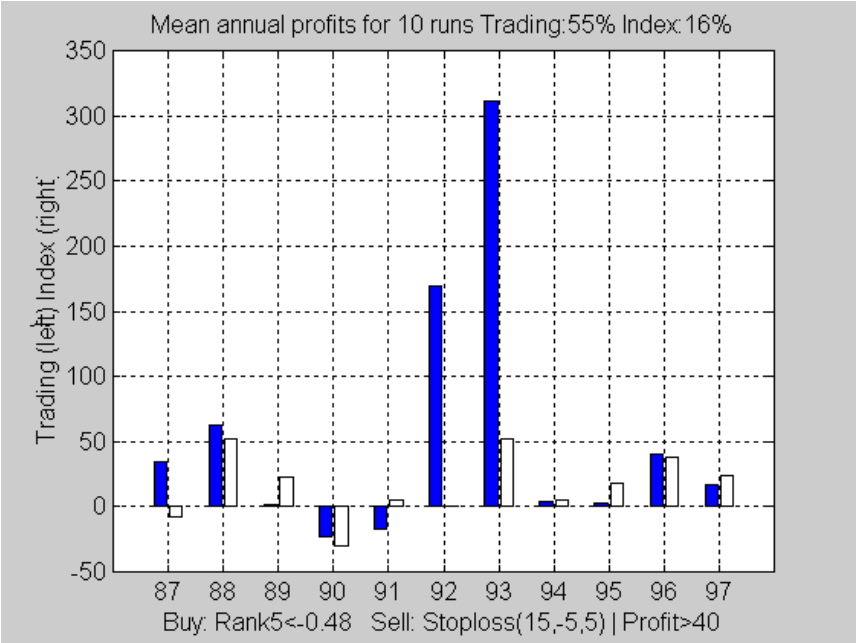


Figure 21: Trading result for the trading defined in Figure 20

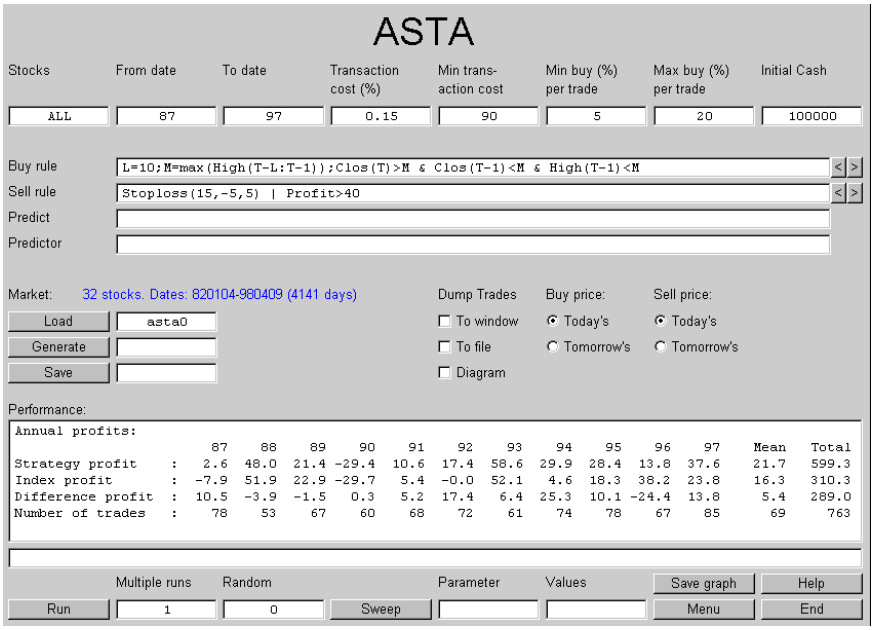


Figure 22: Trading simulation with a complex expression written in Matlab syntax in the Buy rule field.

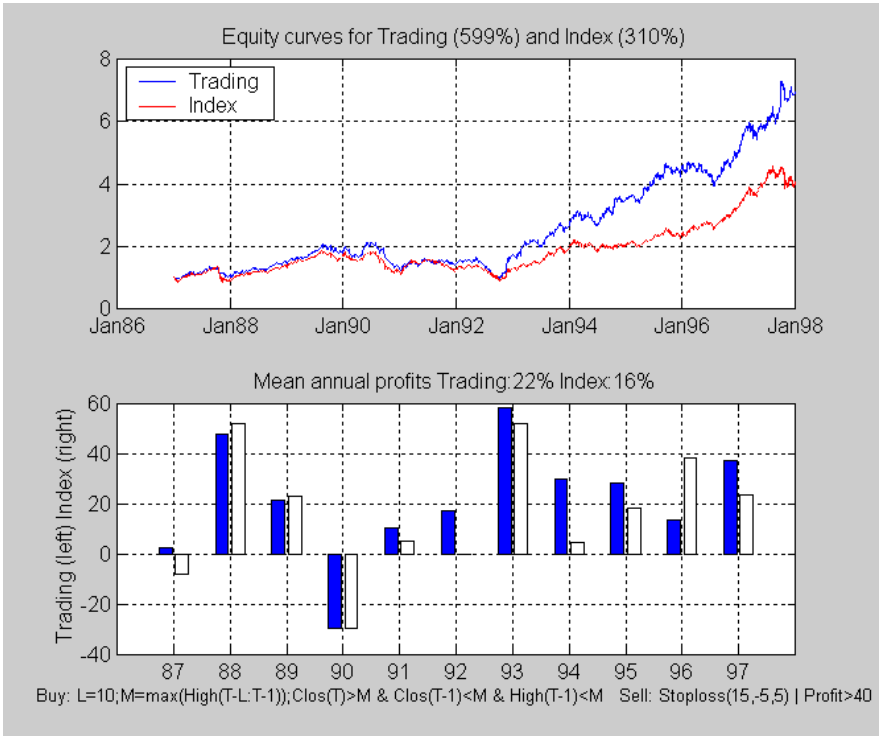


Figure 23: Performance for 1 simulation with the Buy and Sell rules defined in Figure 22. The performance for the trading rule is better than index 8 out of 11 years.

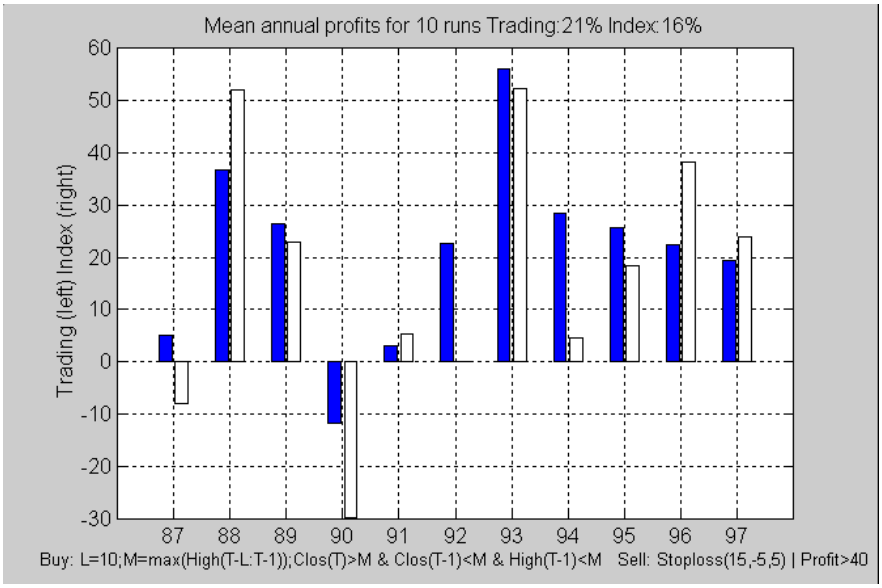


Figure 24: Performance for 10 simulations with the same Buy and Sell rules as in Figures 22 and 23. By multiple simulations, the random acceptance of buy signals gives more stable performance measures.

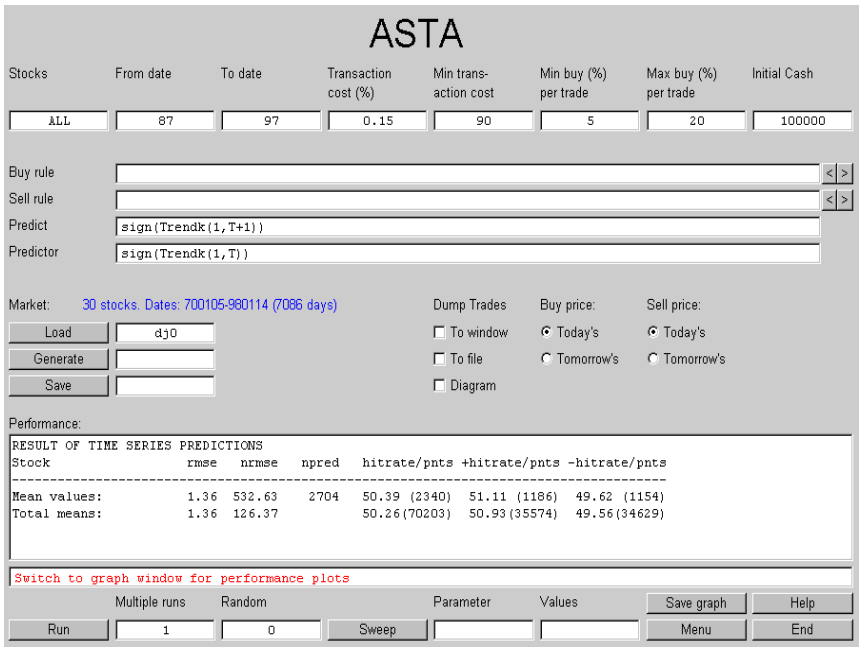


Figure 25: Fixed horizon prediction of 30 stocks from Dow Jones Index. The predictor is the naive prediction that an increase follows an increase and vice versa.

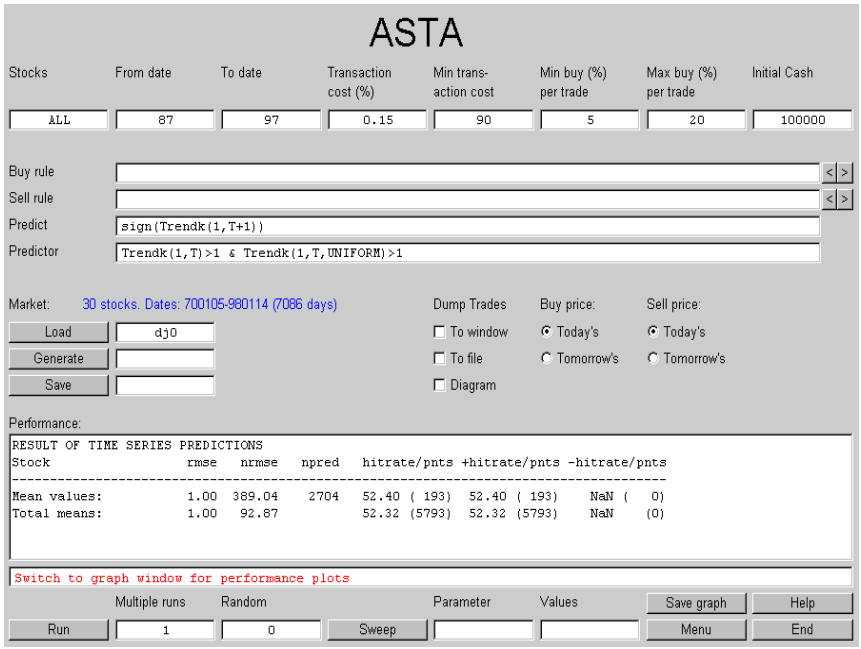


Figure 26: Another fixed horizon prediction of Dow Jones stocks. The Predictor predicts that an increase follows an increase if the increase is at least 1% and a uniform index of all stocks also shows the same 1% increase.

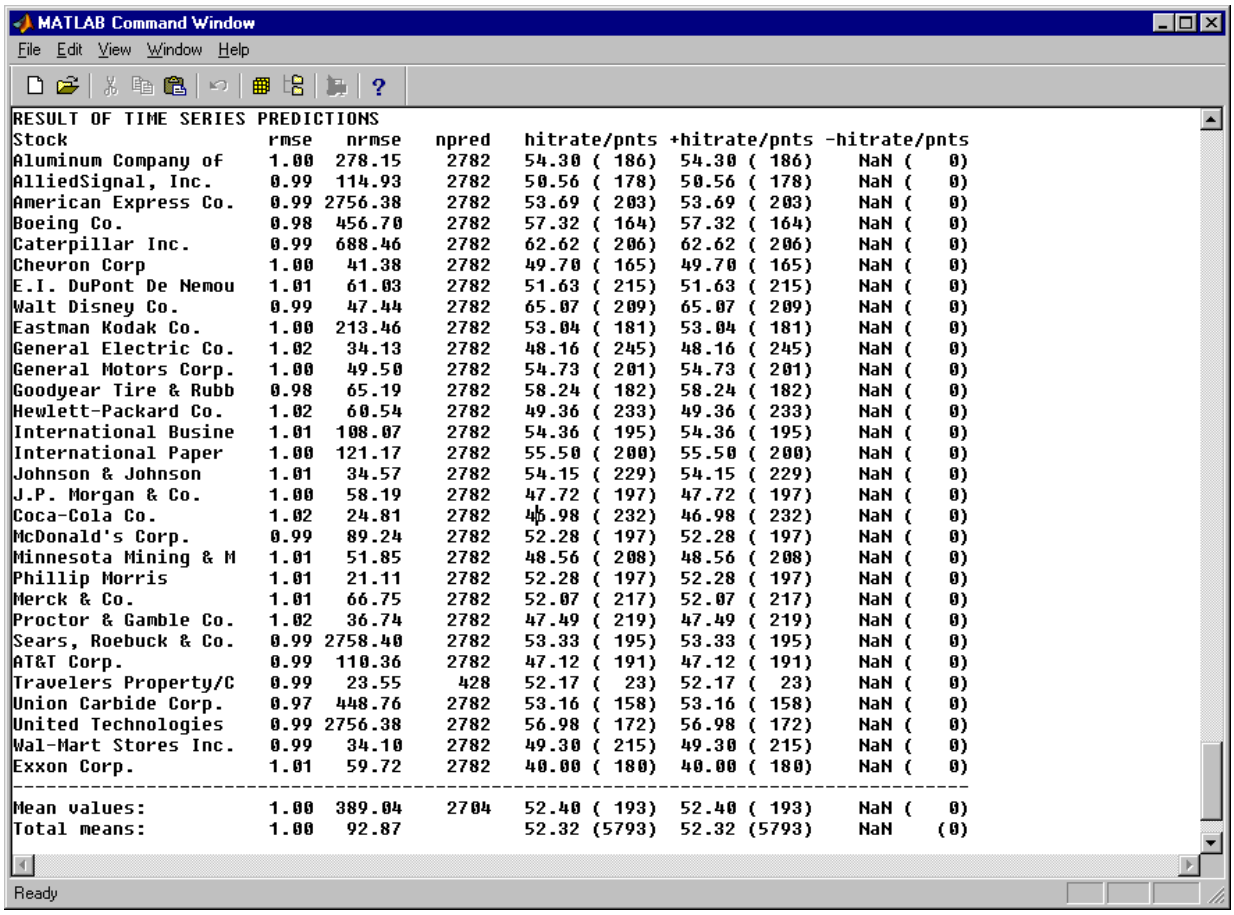


Figure 27: Result from the fixed horizon prediction in Figure 26. The performance is presented for each stock separately in the Matlab command window.