Neural Network Pre-Processing and Analysis Routines
for
Financial Forecasting

Revised 25-Sep-92 to include Non-Linear Dynamical Systems Routines

These routines are translations and extensions of several routines in the Wing-Z script published in Advanced Technology for Developers, Volume 1, August 1992 issue, as well as Wing-Z scripts for Non-Linear Systems Analysis published in the September 1992 issue. These routines are for the sole use of ATD disk subscribers and may not be copied, re-distributed, sold, posted to a bulletin board or in any other way provided to third parties or embedded in other systems without the express written permission of the publisher of Advanced Technology for Developers.

These programs were developed without any government or public sector funding. All of these programs were written or translated from Wing-Z scripts by Casimir C. Klimasauskas.

DISCLAIMER

These programs are provided on an "as is" basis. The publisher of Advanced Technology for Developers makes no warranties as to fitness, operation, or results which may be derived from the use of these programs. The publisher of Advanced Technology for Developers makes no claim or warranty that these programs do not infringe any domestic or foreign patents or copyrights. The limit of liabilities of the publisher of Advanced Technology for Developers is the price paid for the diskette containing these routines.

Contents

System Requirements

All of the programs provided with this document were compiled and linked using the Zortech "C/C++" compiler and developers routines. The large memory model was used. A floating point co-processor is not required. In some circumstances, it may be advisable

to re-compile the programs using in-line floating point, and 386 protected mode. This will result in substantially faster processing and the ability to deal with larger data-bases. The basic requirements are:

IBM PC or compatible with 286, 386, or 486 processor
500K application memory
math co-processor (for 286 or 386) **required**

NOTE: due to the computationally intensive nature of the non-linear dynamical system routines, the code has been compiled with in-line floating point instructions. As such, a math co-processor is REQUIRED. The name of the analysis program has been changed to "FINPRE**2**.EXE". The linear regression model in FINRTN.C has also been changed to improve stability of the numeric algorithm.

With 500K of application memory, it is possible to generate a total of 100 fields, each covering 750 trading days. This means that the program (FINPRE2.EXE) can load a data-base of 20 fields over 750 trading days, and produce 80 transformations of that data.

If you decide to change the source code in any way, the following compiler will be required or its equivalent:

Zortech C/C++ Compiler (Symantec)

These programs have been written with the intent of portability across multiple computer systems. They should port quickly to most work-stations. If there are difficulties in this area, High-Tech Communications may be able to provide assistance on a fee basis.

Creating a Financial Data-Base

FINBLD.EXE is a DOS program designed to process a Compu-Serve session file and produce a data-base as an output. This disk contains a file "finsess.csv" which contains a sample Compu-Serve session. This session is used to illustrate the various steps in the process. The session collects 750 trading days of information for the commodities TBOND and TNOTE. The beginning and the end of the file were edited to remove log-on information, however, this is not necessary. The data was collected by using a standard communications program to connect to Compu-Serve. The "log session" option of the communication program was used to record each character received by the program. Communication was in full-duplex mode with remote echo. Other methods may be used to collect the data. However the final format must be as follows:

```
ticker: symbol
date    label   label   label   label
-----   -----   -----   -----   ------
....
```

All data must be in fixed width columns. The width of the columns is determined by the position of the dashes under the labels. The line immediately above the dashes is used for labels. Any number of lines can come between the line which contains "ticker: symbol" and the line with dashes. The first column MUST always be the date. The dates must be in ascending order.

As shown with "finsess.csv", multiple securities can be in each session. A data-base is created by:

```
C>del fin.db

C>FINBLD fin.db finsess1.csv finsess2.csv ...
```

First, make sure that any existing data-base of the name has been removed. A warning message will be given that the program cannot find the data-base "fin.db", then one which indicates that is creating it. Any number of session files (up to the maximum line size allowed by DOS) can be processed. Each session file may contain multiple securities. Each security is merged into the data-base column-wise in alphabetical order. Within a particular security, the columns are arranged in alphabetical order by label. When a session is processed with contains a column with a label not already in the data-base, this is automatically added. Date synchronization is automatically handled across multiple securities.

The translation process automatically handles conversion of dates in any of the following forms. Dates are always stored in the data-base in the "preferred format". This is compatible with Excel and several other spread-sheets.

```
mm-dd-yy
mm/dd/yy      (preferred format)
dd-MMM-yy
```

It also can handle price data in any of the following formats:

```
dd.ddd        - decimal
dd dd/ddd     - fractional
```

All data is converted to decimal format for internal processing and external storage. Special tags to indicate the reason for missing data (hol, na, nd, n/a) are all translated to the value zero. A maximum of 512 fields, including the date field, can be merged. This is limited by the define "MAXFIELD". The format of the data-base as created or updated is designed for easy access by Excel or other spread-sheets which use tab-separated data-fields. The format of the generated data-base is:

```
\t            ticker\t      ticker\t      ticker...
date\t        label\t       label\t       label...
```

01/01/88 80.55 81.22 ...

The program is smart enough to be able to identify when one block of data ends and another begins. It is also able to recognize and ignore Compu-Serve session data. It is possible to log the results of an interactive session to a file and then process the resulting log file directly.

To create a data-base named "fin.db" from the session "finsess.csv":

```
C>del fin.db          - make sure the data-base does not exist
C>FINBLD fin.db finsess.csv
```

The resulting data-base will contain 750 trading days of the two securities TBOND and TNOTE. The file "finsess.csv" is provided for illustrative purposes.

NOTE: it is possible to process more than one session at a time. The data-base is updated by:

```
C>FINBLD fin.db finupd.csv
```

Where "finupd.csv" is the session to use to update the data-base. An update can add additional columns or extend the number of trading days in the system.

Creating Network Ready Data

FINPRE2 - Financial Pre-processing Program

The financial transformation program, "FINPRE2" takes the data-base created and updated by "FINBLD" and provides the ability to generate a variety of transformations using the "C" equivalents of the routines (finrtn.c) described in the accompanying article. To make it easier to work with the program, a command line parser (fincmd.c) is used to handle a very simple command language. For the command portion, it is only necessary to type the first few letters of the command which are required to uniquely identify it. Other features of the command line parser:

```
??          - prints out the entire command set syntax
?lo         - prints out the value of all variables which begin with lo
lo?         - prints out the command syntax for commands that being
                with lo.
```

In addition to providing the capability of generating network data, the FINPRE2 program is also capable of analyzing the output of a neural network and integrating it with a

trading strategy. This is done with the commands described under the "analysis" section.

Command Set:

load	data_base_name [,start [,end]]	- loads the master data-base
saveall	data_base_name [,start [,end]]	- creates a new data-base including transform data
interpolate		- interpolate "zero" values
scale	min,max	- scale transformed data
transform	data_file [,start_date [,end_date]]	- creates a transform file
log	filename	- log all commands
dictionary		- print out current dictionary
quit		- terminates the program

Transformations commands:

movingaverage	field,days	- moving average
rsi	field,days	- relative strength index
stochastic	field,days[,smooth]	- smoothed stochastic
volatility	field,days	- volatility
highlow	h-field,l-field,days	- high/low volatility
slope	[x-field],y-field,days	- regression slope
intercept	[x-field],y-field,days	- regression intercept
correlation	x-field,y-field,days	- rolling correlation
raw	field	- copy raw data (with lag/lead)

field: ticker.field[offset]

example:	tbond.high	- current "t-bond" high value
	tbond.high[0]	- ...ditto...
	tbond.settle[-2]	- t-bond settle value 2-days ago

Analysis:

network	file,start_date	- load ".nnr" file at start_date
report	{all,summary,transactions}	- specify what to report
long	lowlimit,highlimit	- long range
short	lowlimit,highlimit	- short range - out of market is anything between short high & long low.
capital	startingcapital	- starting capital
price	field	- field which contains closing price
analyze	startdate[,field]	- analyze results from startdate - put analysis in "log" file

- if more than one field, specify #

NOTE: start = start date; end = end date. This allows for loading of a portion of the data-base or saving of a portion of it.

A sample session to create a test set and training set follows:

```
C>FINPRE2.EXE
Command> log fin.log                                  ! log commands to file
Command> load fin.db                                  ! load the data-base
Command> interpolate                                  ! interpolate holes
Command> raw tnote.settle[-5]                         ! five days prior
Command> raw tnote.settle[-4]                         ! four days prior
Command> raw tnote.settle[-3]                         ! three days prior
Command> raw tnote.settle[-2]                         ! two days prior
Command> raw tnote.settle[-1]                         ! yesterday
Command> raw tnote.settle                             ! today
Command> rsi   tnote.settle[-5],10                    ! five day shifted rsi
                                                      !   over 10 trading days
Command> raw tnote.settle[1]                          ! predict tomorrow's price
Command> scale -1,1                          ! scale all data
Command> trans wrktrain.nna,0/0/0,12/31/87            ! generate training file
Command> trans wrktest.nna,1/1/88,12/31/99            ! generate test file
Command> quit                                         ! terminate the session
```

Non-Linear Dynamic Systems Analysis

There are three sub-sections which implement dynamical system analysis. Each of these has its own set of parameters and commands which are independent from the rest of the program. These routines were described in the September 1992 issue of Advanced Technology for Developers. The WingZ scripts were translated to "C". In some instances, the algorithms were re-written to improve the performance of the resulting system. The three commands are:

hurst	- compute the Hurst dimension of a series
correlationdim	- compute the correlation dimension of a series
lyapunov	- compute the largest lyapunov exponent

Each of these commands activates a "subsection" which is described below. For more information about how to set the parameters, see the article in the September 1992 Advanced Technology for Developers or "Chaos and Order in the Capital Markets" by Edgar E. Peters. Each of these analysis routines has been coded and to the best knowledge of the translator, Casimir C. Klimasauskas, they accurately reflect the correct algorithms. However, they have not been tested against bench-mark data.

Hurst Analysis:

series	ticker.field,start_date,end_date	- select the series to analyze
startwindowsize	size	- starting window size (# of items)
deltawindowsize	size	- amount to increment window by each iteration.
iterations	niter	- set the number of iterations. 100 is a good number.
derivative	no	- if yes, use the derivative of the series
printfile	name,type	- re-direct printed output to file "name"
type	name	- type this file to the console
go		- perform the Hurst analysis.

Correlation Dimension:

series	ticker.field,start_date,end_date	- select the series to analyze
embedding	dim	- nominal embedding dimension
tau	steps	- number of time steps between dimensions for re-constructing the phase space
iterations	niter	- set the number of iterations. 100 is a good number.
printfile	name,type	- re-direct printed output to file "name"
type	name	- type this file to the console
go	s,e	- perform the correlation dimension analysis. If "s" and "e" are present, perform the analysis setting the embedding dimension to all values from "s" through "e". Otherwise, just use the embedding dimension.

Lyapunov Exponent

series	ticker.field,start_date,end_date	- select the series to analyze
embedding	dim	- nominal embedding dimension
tau	steps	- number of time steps between dimensions for re-constructing the phase space
minscale	min	- minimum radius for computations
maxscale	max	- maximum radius for computations
evolve	iter	- number of time steps in each evolution
lagminimum	steps	- minimum lag for "near" points (typically

		10-12).
printfile	name,type	- re-direct printed output to file "name"
type	name	- type this file to the console
go	s,e	- perform the Lyapunov Exponent analysis. If "s" and "e" are present, perform the analysis setting the embedding dimension to all values from "s" through "e". Otherwise, just use the embedding dimension.

For printed output, the types of output file are: formatted (formatted ascii text), lotus (comma separated values, quoted literals), excel (tab separated values). These formats make it easy to import the results into a spreadsheet or word-processing program.

A sample session to analyze the time series "tnote.settle" is shown below. Prior to this step, it is necessary to construct a data-base as described previously (FINBLD above).

```
C>FINPRE2.EXE
Command> log fin.log                    ! log commands to file
Command> load fin.db                    ! load the data-base
Command> interpolate                    ! interpolate holes ***
Command> hurst                          ! hurst analysis
Hurst> series tnote.settle              ! series to analyze
Hurst> printfile hurst.csv,lotus        ! write results in lotus-format
Hurst> derivative = yes                 ! compute derivative
Hurst> iterations = 50          ! 50 iterations
Hurst> startwindow = 5                  ! start window size (1 week)
Hurst> deltawindow = 5                  ! increment by one week
Hurst> go                               ! execute hurst calculations
Hurst> type                             ! type the print file
Hurst> quit                             ! exit from hurst analysis

Command> correlationdim                 ! enter corr dim section
CorrDim> series tnote.settle    ! series to analyze
CorrDim> print corrdim.tab,excel        ! tab separated values
CorrDim> tau = 1                        ! one day per phase space
CorrDim> iterations=100                 ! 100 iterations
CorrDim> go 2,6                         ! run for embed=2 to 6
CorrDim> quit                           ! return to main menu

Command> lyapunov               ! enter lyapunov analysis
Lyapunov> series tnote.settle   ! series to analyze
Lyapunov> print lyapun.txt,formatted    ! formatted text
Lyapunov>  go 2,6                       ! run for embed=2 to 6
```

```
Lyapunov> quit                          ! exit from lyapunov analysis
Command> quit                           ! terminate finpre2.
```

*** It is very important to interpolate the holes in the data-base. Failure to do so will result in incorrect processing by the analysis routines.

Analyzing the Output of a Neural Network

A sample session to analyze the output of the network follows:

```
C>FINPRE2.EXE
Command> log fina.log                       ! log results of analysis
Command> load fin.db                        ! load the data-base
Command> network wrktest.nnr,1/1/88         ! load the network output
Command> price tnote.settle             ! price
Command> report all                     ! summary & transactions
Command> capital 50000                      ! starting capital
Command> analyze                            ! do analysis
Command> quit                               ! exit the program
```

Appendix A - Technical Notes

This section contains some brief notes on internal data-structures. It is not designed as an indepth description of how the system works.

Program files:

fin.h	- defines and proto-types
fincmd.h	- defines and proto-types for command line processor
finbld.c	- routines to generate the FINBLD.exe (main program)
fincmd.c	- command line parser
findic.c	- dictionary management
finevl.c	- neural network evaluation routines translated from WingZ
finpre2.c	- pre-processing driver (main program)
finrtn.c	- financial routines translated from WingZ
finnlds.c	- non-linear dynamical system analysis routines from WingZ
makefile	- program to build the executable programs

Internal Structures:

Global Data:

FirstDate	- starting date of the master data-base
LastDate	- ending date of the master data-base
VectorI	- # of items in each vector

Field Array (maximum of 1024 fields):

FTickCP	- pointer to ticker symbol (dictionary)
FSymCP	- field within ticker
FDataFP	- data vector for this field
FFlagsI	- flags: data-base, transform, interpolated

for generated fields:

FTickCP is "transXXX" indicating the transform number
FSymCP is the transform type itself

for network output fields:

FTickCP is "network" indicating from network
FSymCP is "fieldXXX" to indicate the specific field

When a field name is parsed, it fills in the following structure:

FTickCP	- ticker symbol
FSymCP	- field within ticker
FOffsetI	- offset +/- for lead/lag generation