Series on Advanced Economic Issues
Faculty of Economics, VŠB-TU Ostrava

Aleš Kresta

FINANCIAL ENGINEERING IN MATLAB: SELECTED APPROACHES AND ALGORITHMS

Ostrava, 2015

Aleš Kresta
Department of Finance
Faculty of Economics
VŠB-Technical University Ostrava
Sokolská 33
701 21 Ostrava, CZ
ales.kresta@vsb.cz

Review

Miloš Kopa, Charles University in Prague
Sergio Ortobelli Lozza, University of Bergamo

The research was supported by the European Social Fund under the Opportunity for young researchers project (CZ.1.07/2.3.00/30.0016) as well as by GA ČR (Czech Science Foundation – Grantová Agentura České Republiky) under the project no. 13-18300P.

The text should be cited as follows: Kresta, A. (2015). *Financial Engineering in Matlab: Selected Approaches and Algorithms*, SAEI, vol. 33. Ostrava: VSB-TU Ostrava.

Printed in KLEINWÄCHTER s.r.o.
Cover design by MD communications, s.r.o.

ISBN 978-80-248-3702-4

Preface

In recent decades we have been witnessing the dynamic evolution in the financial engineering[1], which is a multidisciplinary field involving applied mathematics, computer science, statistics and economic theory. The attention was mostly given to the areas of corporate finance, pricing of derivatives and structured products, financial regulation, portfolio and risk management and algorithmic trading.

The practice of financial engineering has also been subject to a criticism in recent years. Probably, the most know is Taleb (2008), which is generally perceived as the criticism of a current state of quantitative financial models. However, Taleb does not state the pointlessness of the quantitative models. Rather, he is pointing out that: i) for contemporary models the probabilities of extreme events, which he calls black swans, are generally underestimated; ii) estimation of these probabilities from historical observations is nearly impossible; iii) we, as a mankind, are prone to find rules and causes in what was purely random (he says that we are fooled by the narration fallacy), for these phenomena see also Taleb (2013). To conclude, Taleb is pointing out that the contemporary quantitative models are not panacea and should be taken with moderation. We can, however, add that imprecise models are better than no models at all. On the other hand, the advantage of the quantitative models is that they can be statistically tested.[2]

The book is focused on the essential part of financial engineering, which is financial time series modelling and its application in portfolio management and risk management. Contemporary state of the art of financial time series modelling is connected to the Efficient Market Hypothesis according to which *prices fully reflect all available information* and hence are unforecastable, see Samuelson (1965) and Fama (1965a, 1965b, 1970). A strong form of this hypothesis presumes asset returns to be even independent and identically distributed, see e.g. Fama (1970). However, note that works, such as Lo and MacKinlay (2011) and Lo et al. (2000), provided compelling evidence that markets are not efficient, i.e. price data do possess statistical properties that noticeably deviate from the models of random price evolution.

In the book we explain principles and models of financial time series modelling, as well as provide reader with immediate practical applications of these principles in programming language. The example codes are written in Matlab environment (for brief description of the environment and the programming language see Appendix A). We are aware that there are some books already written on the topic of financial engineering providing applications in Matlab, see for

[1] Sometimes also addressed as a quantitative or mathematical finance.

[2] Good example is a technical analysis. If the technical analysis is performed subjectively, the accuracy can be hardly tested. However, if the technical rules are clearly defined, the accuracy can be easily back-tested on the historical data. However, even in this case the results should be taken carefully and one should be aware of possible data-snooping bias. A well written reference in this field is Aronson (2006).

instance Chan (2008, 2013), Huynh et al. (2008) or Kienitz and Wetterau (2012). However, in these books the discussed topics and codes are focused rather specifically. In this book we want to provide the reader with both the knowledge of selected approaches in financial time series modelling and fully working programs[3], which can be directly utilized without any knowledge about programming in Matlab. Thus, the book can be utilized by both undergraduate and postgraduate students in finance and computer science as the learning reference of Matlab programming language. As the presented models and algorithms are rather simple, we also include the ideas for further research and development at the ends of chapters where it is possible. We hope that the book can provide some interesting ideas for diploma theses elaboration.

The book is structured into five chapters. In order to provide fully working algorithms we had to start at the beginning with the data acquisition, which is the content of the first chapter. In the book we utilize freely available data sources such as Prague Stock Exchange (PSE), Czech National Bank (CNB), Yahoo Finance and TrueFX website. When we download the particular time series it is necessary to combine them together, so that we create an matrix in which one dimension of the matrix represents the particular symbols and second dimension of the matrix represents time (see subchapter 1.3). For some data sources, mainly for tick-by-tick data, these data have to be first resampled in order to be easily manipulated. Thus, we briefly discuss also the data resampling issue (subchapter 1.2).

In the second chapter we describe the approaches and models applicable for modelling of returns. However, the returns must be computed first. Speaking about returns, we can distinguish discrete and continuously compounded returns. The difference is not only in their definition, and thus in computation, but also in the way they can be aggregated over time and the way the portfolio return can be computed (see subchapter 2.1.4). The historical returns calculation and estimation of future returns are described in subchapter 2.1. The returns modelling itself is described through subchapters 2.2–2.6.

Generally, when modelling financial time series, we have to deal with empirical evidences.[4] Firstly, the empirically observed returns of financial time series are characterized by fatter tails compared to the Gaussian (normal) distribution. Thus, it can be concluded, in line with Mandelbrot (1963b), that Gaussian distribution is not appropriate for modelling of financial returns. We address this issue by introducing Student distribution (subchapter 2.2.2) and Lévy family of models (subchapter 2.3). Secondly, empirical volatility of returns is not constant over time, but is rather clustered. Thus, for the same asset, the periods with high volatility (high profits/losses) can be seen as well as the periods in which volatility is low (the profits/losses are close to zero). This issue can be tackled by volatility modelling. We address this feature in subchapter 2.4. Lastly, we have to

[3] The programs presented in this book are freely obtainable upon an e-mail request to the author at ales.kresta@vsb.cz.
[4] See e.g. Cont (2001) for the summary of empirical properties.

deal with the dependency among particular time series. Generally, the returns are not correlated strongly when they are around zero, however, in the tails the correlation increases. Appropriate tool for dependency modelling are copula functions based on Sklar's theorem, see Sklar (1959, 1973), which allows to decompose the joint distribution into marginal distributions and copula function. The distribution of particular time series is then modelled by marginal distributions, while dependency is tackled only by copula function. The copula function and dependency modelling are discussed in subchapter 2.5.

In the third chapter we introduce simple mean-variance portfolio optimization framework as introduced by Markowitz (1952). In this framework the portfolios are described by only two parameters – the mean and the standard deviation of future one-period-return probability distribution. Concerning these two parameters we can distinguish three set of portfolios: feasible, efficient and optimal. Knowing the particular investor's risk profile we can optimize portfolio composition and find directly optimal portfolio. However, without this knowledge, the best we can obtain is only Pareto efficient set of portfolios. The search for efficient set of portfolios is first illustrated by a rather naive method (subchapter 3.2) consisting of stratification and brute-force method.[5] Then, the efficient set of portfolios is obtained as the solution to the quadratic optimization problem (subchapter 3.3). At the end of the chapter we define optimal portfolio optimization problem for the case that we know investor's risk profile.

In the fourth chapter we apply the optimal portfolio optimization problem to real datasets of American stocks traded at NYSE and Nasdaq (description of dataset can be found in subchapter 1.3). However, before presenting empirical results (subchapter 4.3), we first define basic performance measures (subchapter 4.2) such as maximum drawdown and Sharpe and Rachev ratios. Then the portfolios performances are studied for different risk profiles, portfolio recalibration periods and historical windows of parameters estimation.

When holding the particular portfolio, the investors are interested in the risk which is connected to their position. The same holds for financial institutions, which have to compute the risk arising from the portfolio of financial instruments they hold. The risk estimation and its backtesting is described in the fifth chapter. Although there were introduced many risk measures in the literature, the most discussed measures nowadays are Value at Risk (see subchapter 5.1.1) and Conditional Value at Risk (see subchapter 5.1.2). Particularly Value at Risk is nowadays mostly utilized measure of risk – for financial institutions even obligatory, see Basel II and Solvency II regulations. We can generally distinguish three groups of methods for risk estimation. They are historical simulation, analytical solution and Monte Carlo simulation. While historical simulation

[5] In computer science, brute-force search is a very general problem-solving technique that consists of systematically enumerating all possible candidates for the solution and checking whether particular candidates satisfy the problem's statement.

(subchapter 5.2.1) is a nonparametric[6] method, in the latter two methods we assume returns to be distributed according to particular parametric probability distribution. The difference is whether we are able to obtain the analytical formula of risk quantification (subchapter 5.2.3) or Monte Carlo simulation has to be applied (subchapter 5.2.4). These methods have their advantages and disadvantages. Also their accuracy can differ for different portfolios. The accuracy is evaluated by the so-called backtesting procedure (see subchapter 5.3). At the end of the chapter we provide the empirical results of particular estimation methods for dataset consisting of American stocks included in Dow Jones Industrial Average index. In performed analyses we back-tested the methods in the period from November 30, 1998 until September 1, 2014, i.e. almost sixteen years of historical data.

The book was written as the part of the research supported by the European Social Fund under the Opportunity for young researchers project (CZ.1.07/2.3.00/30.0016) as well as by GA ČR (Czech Science Foundation – Grantová Agentura České Republiky) under the project no. 13-18300P. All the support is greatly acknowledged and appreciated.

[6] No specific parametric distribution of returns is assumed. We rather work with the empirical distribution of returns.

Contents

Denotations, symbols and abbreviations

C	copula function; C_{ρ}^{Ga} express Gaussian copula function, $C_{\rho,\nu}^{St}$ express Student copula function and $C_{\varphi,\varphi^{[-1]}}^{Arch}$ express Archimedean copula function
$CVaR_{\alpha}$	Conditional Value at Risk at given probability level α
e	the standardized residuals in filtered historical simulation method
$E(R)$	expected return, i.e. the mean of the probability distribution of returns
f	probability density function; f_t for Student probability distribution, f_N for Gaussian probability distribution
F	cumulative distribution function of a random variable; F_t for Student probability distribution, Φ for Gaussian probability distribution
k	parameter stating the level of risk aversion of the investor
$l(t)$	subordinator process in Lévy models
L	likelihood function
LR	specified likelihood ratio, e.g. LR_{Kupiec} for likelihood ratio of Kupiec's test
m	the length of time series left for the parameters estimation, i.e. the length of data not utilized for backtesting procedure as they are utilized for initial parameters estimation
n	number of backtesting observations; n_1 refers to quantity of exceptions (VaR violations), n_0 refers to quantity of observations in which exception did not happen
P_i	price of the *i*-th asset; the price of the *i*-th asset in specified time *t* is addressed as $P_{i,t}$
P_P	portfolio value at time; the value of the portfolio in specified time *t* is addressed as $P_{P,t}$
$\mathbf{Q}$	covariance matrix
r	continuously compound return; the portfolio returns are referred as r_P and assets returns as r_i
R	discrete return; the portfolio returns are referred as R_P, assets returns as R_i, risk-free rate as R_{RF} and returns of a benchmark as R_B

t	current time (or alternatively some specified time)
T	final time of the interval
u	random returns normalized by cumulative distribution function
v	quantity of particular asset in portfolio
VaR_α	Value at Risk at given probability level α
w	relative amount of wealth invested in particular asset when referred ex-post
W	wealth and also the value of the portfolio; wealth in particular time is addressed either as $W(t)$ or W_t
x	relative amount of the wealth invested in particular asset when referred ex-ante
X, Y	random variables
α	probability level of VaR forecast; $1-\alpha$ express the confidence level of VaR, i.e. the probability with which the loss will not exceed the estimated VaR
Γ	gamma function
Δ	discrete time step
ν	degrees of freedom in Student distribution
π	probabilities of exceptions occurring; π_{obs} observed probability of exceptions occurring, π_{ex} expected probability of exceptions occurring, π_{01} conditional probability of exceptions occurring
ρ	correlation coefficient ($\rho_{X,Y}$ states the correlation coefficient of random variables *X* and *Y);* alternatively risk measure
σ	standard deviation
σ^2	variance of random variable
$\sigma_{X,Y}$	covariance between random variables *X* and *Y*
τ	some specified time
φ	generator function utilized in the definition of Archimedean copula functions; alternatively characteristic function of Lévy models distributions
Φ	cumulative distribution function of normal distribution; Φ_s refer to cumulative distribution function of standard normal distribution

AR(*O*)-GARCH(*P*,*Q*)	autoregressive process of order *O* with innovations modelled by GARCH process of order *P*, *Q*
$N(\mu,\sigma)$	normal distribution with mean μ and standard deviation σ; $N(0,1)$ is standard normal distribution, i.e. normal distribution with zero mean and unit variance
$NIG(\mu,\alpha,\beta,\delta)$	normal inverse Gaussian distribution defined by parameters μ, α, β and δ
$NIG(\mu,\vartheta,\nu,\theta)$	alternative definition of Normal inverse Gaussian distribution as a subordinated Lévy model with parameters μ, ϑ, ν and θ
$VG(\mu,\vartheta,\nu,\theta)$	variance-gamma probability distribution defined as a subordinated Lévy model with parameters μ, ϑ, ν and θ
$\Pr(x)$	probability that *x* happens
cdf	cumulative distribution function
inf	infinity
pdf	probability density function
AIC	Akaike information criterion
AR	autoregressive model
BIC	Bayesian information criterion
CML	canonical maximum likelihood
CNB	Czech National Bank
CVaR	Conditional Value at Risk
CZK	Czech crown currency
DD	drawdown
DJIA	Dow Jones industrial average price index
EMLM	exact maximum likelihood method
EUR	euro currency
FHS	filtered historical simulation
GARCH	generalized autoregressive conditional heteroskedasticity model
HS	historical simulation method
IFM	inference function for margins
K-test	Kupiec's test

LR	likelihood ratio
MC simulation	Monte Carlo simulation
MDD	maximum drawdown
NYSE	New York stock exchange
OHLC	open, high, low, close values
PSE	Prague stock exchange
PX	Prague stock price index
PX-GLOB	Prague stock broad-based price index
PX-TR	Prague stock total return index
RR	Rachev ratio
S&P 500	Standard & Poor's 500 price index
SL	stop loss
SR	Sharpe ratio
TP	take profit
URL	uniform resource locator
USD	US dollar currency
VaR	Value at Risk

List of Utilized Matlab Built-in Functions

aicbic	returns the values of Akaike information criteria (AIC) and Bayesian information criteria (BIC)
area	produces a 2-D stacked area plot suitable for showing the contributions of various components to a whole
axis	controls axis scaling and appearance
binocdf	returns the value of binomial cumulative distribution function
binopdf	returns the value of binomial probability density function
break	terminates the execution of *while* or *for* loops
cdf	returns the value of cumulative distribution function of the specified probability distribution
ceil	rounds the input toward plus infinity
clc	clears the Command Window (previously submitted commands are still stored in Command History panel)
clear	clears (the specified) variables and functions from the memory
clear all	clears all variables and functions from the memory
continue	skips the execution of actual *for* or *while* loop's iteration and passes the control to the next iteration
contour	plots the contour graph
copulacdf	returns the value of cumulative distribution function for the specified copula function
copulafit	fits the parameters of specified copula function to data
copulapdf	returns the value of probability density function for the specified copula function
copularnd	generates random numbers from the specified copula function
corr	returns correlation matrix of the specified data
cov	returns the covariance matrix of the specified data
cumprod	returns cumulative products of the input elements, works both with vectors and matrices
cumsum	returns cumulative sums of the input elements, works both with vectors and matrices
dateaxis	formats specified axis labels to the specified date/time format, interchangeable with *datetick*
datenum	converts date vector or char array into serial date number
datestr	converts serial date number to the date string of specified format

datetick	formats specified axis labels to the specified date/time format, interchangeable with *dateaxis*
diff	returns the differences between subsequent elements in the specified vector or matrix
dir	returns the list of files in the specified directory
eps	returns the smallest distinguishable change in floating point number
exp	returns the value of exponential function
eye	returns the identity matrix of the specified size
fclose	closes the specified file, see also *fopen*
figure	creates a new figure window
fix	rounds the input toward zero
flipud	flips the matrix or vector upside down
floor	rounds the input toward minus infinity
fmincon	finds the minimum of the function under the linear and nonlinear constraints
fopen	opens the specified file for read/write access, see also *fclose*
fprintf	writes specified formatted data as the output to the Command Window or a text file
garchfit	estimates the parameters of specified ARMAX-GARCH model based on the input data
garchpred	forecasts specified ARMAX-GARCH model responses
garchset	sets the structure of ARMAX-GARCH models
garchsim	simulates ARMAX-GARCH model responses
hold on	holds the current plot and axes properties so that subsequent graphs are plotted over the existing graph
chi2cdf	returns value of chi-square cumulative distribution function
icdf	returns the value of inverse cumulative distribution function for a specified probability distribution
importdata	loads the data from the specified file into the workspace
inf	returns the IEEE arithmetic representation for positive infinity, which is produced by operations like dividing by zero, e.g. 1.0/0.0, or from overflow, e.g. *exp(1000)*
interp1	returns the 1-D interpolated data
kurtosis	returns the sample kurtosis of the input values
legend	displays specified legend in the current graph
length	returns the length of a vector or the maximum size of an array, *length(X)* is equivalent to the command *max(size(X))*
log	returns the value of natural logarithm
log10	returns the value of the decimal logarithm

max	returns the maximum value of the vector or maximum values in the arrays along the specified dimension
mean	returns the average (mean) value of the vector or average values in the arrays along the specified dimension
median	returns the median value of the vector or median values in the arrays along the specified dimension
min	returns the smallest value of the vector or smallest values in the arrays along the specified dimension
mle	returns maximum likelihood estimates of parameters of the specified probability distribution
nan	returns the array of the specified size containing values of NaN – the IEEE arithmetic representation for Not-a-Number, which is obtained as a result of mathematically undefined operations like *0.0/0.0* and *inf–inf*
normfit	estimates parameters and confidence intervals for normal distribution
normrnd	returns the array of the specified size containing pseudorandom numbers drawn from the normal (Gaussian) distribution with specified mean and standard deviation, see also *randn*
nthroot	returns the specified root of the input, *nthroot(a,b)* provides the same result as *power(a,1/b)*
num2str	converts number into string
ones	returns the array of specified size containing values of one
optimset	creates or changes the structure of optimization parameters
portopt	computes the mean-variance efficient frontier
pdf	returns the value of probability density function of the specified probability distribution
plot	plots vector or matrix data (vertical axis) versus a vector data (horizontal axis) preserving the linear scales of the axes
power	returns specified input real number powered by another input real number
prod	returns the products of the elements in the array along the specified dimension
rand	returns the matrix of specified size containing pseudorandom numbers drawn from the uniform distribution on the open interval (0,1)
randn	returns the matrix of specified size containing pseudorandom numbers drawn from the standard normal (Gaussian) distribution, see also *normrnd*
repmat	creates a larger array, which consists of specified quantity of input array copies
reshape	reshapes the input array in the specified manner

round	rounds the input toward nearest integer
save	saves all the variables from the current workspace to the specified Matlab formatted binary file
semilogx	is the same as *plot* function, except for that a logarithmic scale is used for the horizontal axis
semilogy	is the same as *plot* function, except for that a logarithmic scale is used for the vertical axis
size	returns array size – a vector of dimension lengths
skewness	returns the sample skewness of the input values
sort	sorts the input array along the first non-singleton dimension in ascending order
sortrows	sorts the rows of the matrix as a groups in ascending order
sqrt	returns the value of the square root of the input variable
squeeze	removes all the singleton dimensions (dimensions of the size one) from the array, matrices are unaffected by the function
std	returns the values of standard deviations for each column of the input matrix
subplot	breaks the figure window into the specified quantity of tiles, plot the axes in each tile and selects the specified tile for the current plot
sum	returns the sums of the elements in the array along the specified dimension
surf	plots 3-D coloured surface graph
textscan	reads formatted data from the specified text file or string
tic	starts a stopwatch timer, see function *toc*
toc	reads the value of stopwatch timer, see function *tic*
union	returns the combined values of the two vectors (input variables) without repetitions
upper	converts string to uppercase
urlwrite	downloads the URL content and saves it as a file
xlabel	writes the label to the x-axis
ylabel	writes the label to the y-axis
zeros	returns the array of specified size containing values of zero

Chapter 1

Data Acquisition

For any analysis the first inevitable step is the data acquisition. In this chapter we explain from where and how to download the data. We utilize free sources of data. As the obtained data are usually isolated time series it is necessary to combine them together. Sometimes it can be an easy task, as there are no missing data and downloaded time series can be directly put together into a matrix (see subchapters 1.3.2 and 1.3.3), however, in most cases there are missing data, which have to be determined. In the example of small indices dataset we utilize the linear interpolation method (subchapter 1.3.1).

The datasets obtained in this chapter are used in this book for portfolio optimization backtesting (chapter 4) and risk estimation backtesting (chapter 5).

1.1 Imports from Free Sources

There are many available free data sources on the internet. In this book we utilize following freely available data sources: Prague Stock Exchange website (subchapter 1.1.1), Yaohoo Finance website (subchapter 1.1.2), Czech National Bank website (subchapter 1.1.3) and TrueFX website (subchapter 1.1.4). Our intention is not to provide the complete list of the available data sources, but rather to give the examples of some easily utilized ones.

1.1.1 Prague Stock Indices from pse.cz

The Prague Stock Exchange (henceforth PSE) calculates and publishes three national indices in order to provide investors and the wider public with concise information on the performance of the Czech regulated market. They are:

- PX – a tradable price index made up of the most actively traded blue-chips of the Prague Stock Exchange;
- PX-TR – a total return index with the same base as PX index;
- PX-GLOB – a broad-based price index comprising stocks traded on a regulated stock market of the Prague Stock Exchange.

Further we focus on PX and PX-TR indices as they are composed from the most actively traded blue-chips for which the liquidity is not an issue. Both PX

and PX-TR indices are the official indices of the Prague Stock Exchange. They are a capitalization-weighted price and total return[7] indices made up of the most traded blue chips at the Prague Stock Exchange.[8] The indices are calculated in CZK and disseminated in real-time by the Prague Stock Exchange. They are designed as a tradable indices to be used as an underlying asset for structured products and for standardized derivatives.[9] The base of the indices is mutual. Although PX-TR index was launched on March 24, 2014, its values were calculated back to March 20, 2006. At that date it took the same value as PX index.

The history of the indices can be downloaded from the following addresses:

- http://ftp.pse.cz/Info.bas/Cz/PX.csv,
- http://ftp.pse.cz/Info.bas/Cz/PX-TR.csv,
- http://ftp.pse.cz/Info.bas/Cz/PX-GLOB.csv,

and the format of the files possess following characteristics:

- file is in *date, value, change* format,
- the comma is used as the delimiter,
- dot is used as the decimal sign,
- date is in format *dd.mm.yyyy*, where *dd* are days, *mm* are the month numbers and *yyyy* stands for years.

Due to these characteristics, it is easy to import the data as shown by Program 1–1, which also downloads the csv file from the internet and saves it to the current folder.

Program 1–1 Import of PX index history values

```
urlwrite('http://ftp.pse.cz/Info.bas/Cz/PX.csv','data_PX.csv');
import=importdata('data_PX.csv'); %import the file
PX.values=import.data(:,1); %obtain the values of PX index
for a=1:size(import.textdata,1)
   PX.dates(a)=datenum((import.textdata{a}),'dd.mm.yyyy');
end;
clear import;

save data_PX;
%%plot the imported data
figure;
plot(PX.dates, PX.values);
datetick('x');
xlabel('Date');
ylabel('PX Value');
```

Obviously also other indices can be downloaded and imported in the same way, see Program 1–2 for PX-TR index history download and Program 1–3 for PX-GLOBAL index history download.

[7] Due to the consideration of dividend payments the index reflects the total return of the underlying portfolio.

[8] The actual base of PX index and PX-TR index can be found at http://ftp.pse.cz/Info.bas/Cz/PX.pdf and http://ftp.pse.cz/Info.bas/Cz/PX-TR.pdf.

[9] http://en.indices.cc/cooperations/pse/px-tr/

Program 1–2 Import of PX-TR index history values

```
urlwrite('http://ftp.pse.cz/Info.bas/Cz/PX-TR.csv',...
'data_PX-TR.csv');
import=importdata('data_PX-TR.csv'); %import the file
PXTR.values=import.data(:,1); %obtain the values of PX-TR index
for a=1:size(import.textdata,1)
   PXTR.dates(a)=datenum((import.textdata{a}),'dd.mm.yyyy');
end;
clear import;


save data_PX-TR;
%%plot the imported data
figure;
plot(PXTR.dates, PXTR.values);
datetick('x');
xlabel('Date');
ylabel('PX-TR Value');
```

When we have downloaded the data, it would be interesting to compare the evolution of the both PX and PX-TR indices. The comparison is made in Figure 1–1. The graph starts on March 20, 2006 when both indices have the same value of 1,554.6 and ends on September 30, 2014 when the value of PX index was 991.4 and PX-TR 1,441.01. We see that both indices decreased over the examined period with the huge drop in the second half of 2008 (financial crisis). While PX index hasn't recovered yet, PX-TR index values has been steadily increasing to its starting value. The difference in the indices' values is caused by the inclusion of dividend payments in PX-TR index.[10]

Program 1–3 Import of PX-GLOBAL index history values

```
urlwrite('http://ftp.pse.cz/Info.bas/Cz/PX-GLOB.csv',...
'data_PX-GLOB.csv');
import=importdata('data_PX-GLOB.csv'); %import the file
PXGLOB.values=import.data(:,1); %obtain the values of PX-GLOBAL
for a=1:size(import.textdata,1)
   PXGLOB.dates(a)=datenum((import.textdata{a}),'dd.mm.yyyy');
end;
clear import;


save data_PX-GLOB;
%%plot the imported data
figure;
plot(PXGLOB.dates, PXGLOB.values);
datetick('x');
xlabel('Date');
ylabel('PX-GLOBAL Value');
```

[10] Many blue-chip stocks traded at Prague Stock Market pays high dividends. Note that dividend pay-out ratios was high for ČEZ, O2 C.R., PHILIP MORRIS ČR and others.

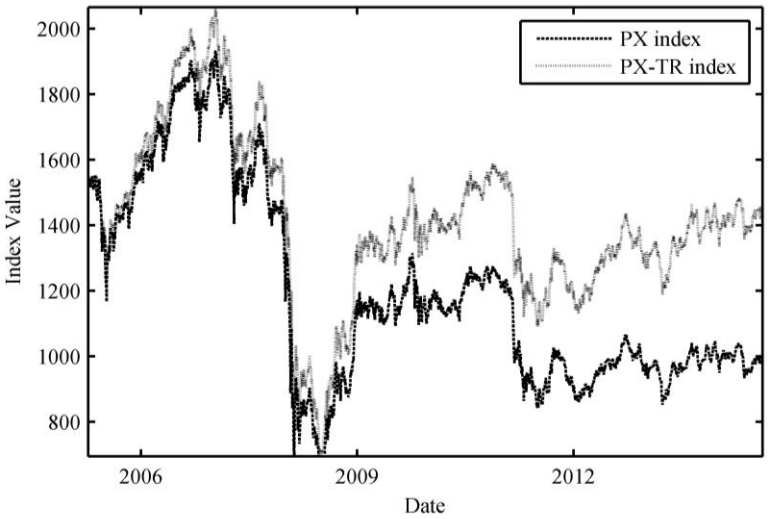

Figure 1–1 Evolution of Prague stock indices

1.1.2 Stock Financial Time Series from Yahoo Finance

In order to obtain the financial time series of stocks or market indices, also the finance.yahoo.com web site can be utilized. Quotes of many stocks can be obtained through this website and the available stocks are not limited to US markets but also European and Asian stock data can be obtained there.

The advantage of this source can be found in the fact that finance.yahoo.com provides data, which are adjusted for splits and dividends paid.[11] The disadvantage is that the YAHOO provides data only for stocks which are currently being traded, i.e. it is impossible to obtain data for the companies which went bankruptcy or left the market from any reason. This makes data, if not threated correctly, prone to the so-called survivorship bias – for further explanation see Chan (2008).

Data in Yahoo Finance data source can be accessed through *.csv files as in the case of Prague stock market indices. It is available at the following URL: http://ichart.finance.yahoo.com/table.csv followed by the specification of symbol, periodicity of the data and period we want to download. This interface is utilized in Program 1–4 which downloads the data for specified *symbol* in specified *periodicity* over the periods specified by the rest of the input parameters.[12]

[11] When looking at the historical prices refer to the last column *Adj Close*.

[12] Another code for downloading YAHOO Finance data can be found for instance at: http://luminouslogic.com/matlab_stock_scripts/get_hist_stock_data.m. The mentioned algorithm does not save the downloaded data to local folder as it applies other Matlab functions for data download.

Program 1–4 Function downloadyahoo.m

```
function [date,open,high,low,close,vol,adjclose]=...
downloadyahoo(symbol, periodicity, from_year, from_month,...
from_day, to_year, to_month, to_day)
% download the data from finance.yahoo.com
% period can be 'd' for daily data, 'w' for weekly data, 'm' for
% monthly data

% Create URL string and download csv file
url_string = ['http://ichart.finance.yahoo.com/table.csv?s='...
upper(symbol) '&a=' num2str(from_month-1) '&b='...
num2str(from_day) '&c=' num2str(from_year) '&d='...
num2str(to_month-1) '&e=', num2str(to_day) '&f...
num2str(to_year) '&g=' periodicity '&ignore=.csv'];
urlwrite(url_string,['data_' upper(symbol) '.csv']);
import=importdata(['data_' upper(symbol) '.csv']);
delete(['data_' upper(symbol) '.csv']);
% Reverse to normal chronological order
open     = flipud(import.data(:,1));
high     = flipud(import.data(:,2));
low      = flipud(import.data(:,3));
close    = flipud(import.data(:,4));
vol      = flipud(import.data(:,5));
adjclose = flipud(import.data(:,6));
for a=length(open):-1:1
  date(length(open)-a+1)=datenum((import.textdata{a+1}),...
'yyyy-mm-dd');
end;
end
```

This function returns OHLC[13] time series as well as the time series of volumes (*vol*), adjusted close (*adjclose*) and corresponding dates. Reader should note that while general convention is to sort time series from the oldest to the most recent record, the Yahoo provides the data in opposite order, thus data have to be flipped (function *flipud*).

1.1.3 Foreign Exchange Rates from CNB

Another source of the files which can be directly imported into Matlab is the website of the Czech National Bank (CNB).[14] This time series consist of foreign exchange rates (FX rates) of foreign currencies to Czech crown with the daily periodicity. The data can be viewed in a web browser or downloaded in two formats: Excel spreadsheet or text file. We can utilize text file option and create a function which will download and import the data, see Program 1–5.

[13] Open-high-low-close prices.

[14] http://www.cnb.cz/en/financial_markets/foreign_exchange_market/exchange_rate_fixing/selected_form.jsp

Program 1–5 Function downloadCNB.m

```
function [date,FXrate]=downloadCNB(symbol,from_year,...
from_month,from_day,to_year,to_month,to_day)

% Create URL string and download csv file
url_string=['http://www.cnb.cz/miranda2/m2/en/financial' ...
'_markets/foreign_exchange_market/exchange_rate_fixing/' ...
'selected.txt?code=' upper(symbol) '&from=' num2str(from_day)...
'.' num2str(from_month) '.' num2str(from_year) '&to='...
num2str(to_day) '.', num2str(to_month) '.' num2str(to_year)];

% Import csv file and obtain data
urlwrite(url_string,['data_' upper(symbol) 'CZK.csv']);
import=importdata(['data_' upper(symbol) 'CZK.csv']);
delete(['data_' upper(symbol) 'CZK.csv']);
FXrate=import.data;
for a=length(FXrate):-1:1
   date(a)=datenum((import.textdata{a+2}),'dd.mmm yyyy');
end;
end
```

1.1.4 Foreign Exchange Rates from TrueFX.com

In this book we utilize only daily data obtained from Yahoo Finance and CNB website. Daily periodicity is enough for most of the analyses, however, it can be sometimes useful to obtain high frequency data. Even high frequency data for the most traded FX pairs can be obtained freely from the internet, namely two following sources: TrueFX[15] and GAIN Capital.[16] Data from both sources have to be downloaded manually. In the book we focus on TrueFX data source as it provides higher-speed download.

As we mentioned in the previous text, the data have to be downloaded manually. It is due to the obligatory registration into the website. After the free registration we can download the data into the local folder. In Figure 1–2 we show the example of the data downloaded into the folder *c:\import*. The reader should note that due to high frequency of the data the size of files is around 100 MB each. We observed that Matlab (run on standard personal computer) can work with at maximum eight of these files at the same time. Thus for the proper analysis an access to a supercomputing facility is needed.

[15] http://truefx.com/?page=downloads
[16] http://ratedata.gaincapital.com

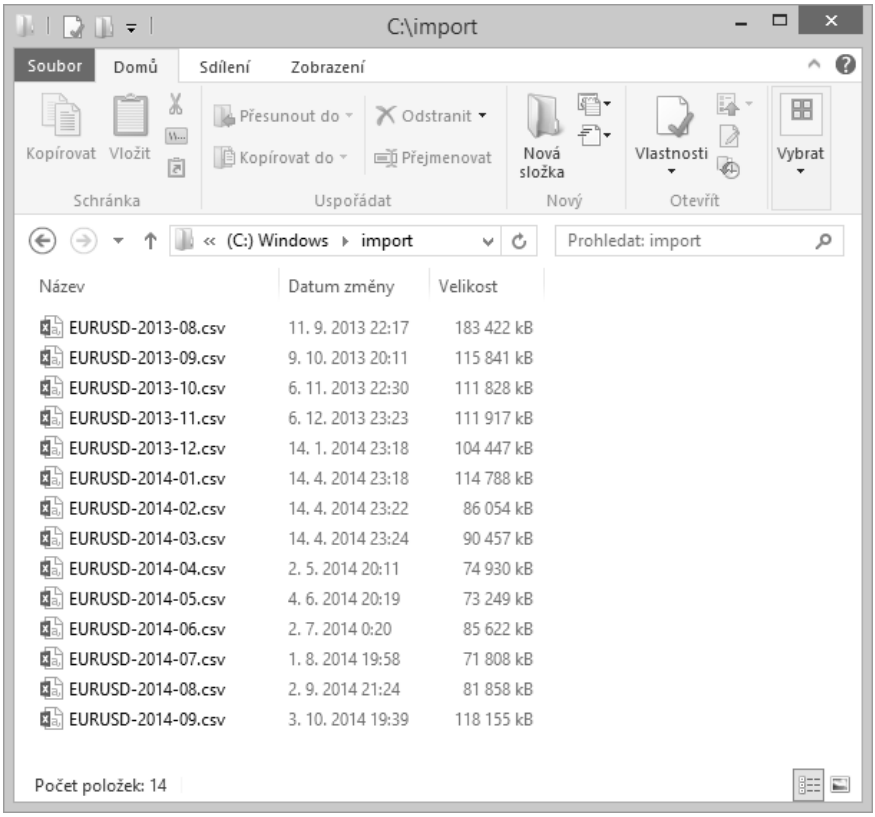

Figure 1–2 Example of the directory containing data downloaded from TrueFX website

The format of each file possess following characteristics:

- file is in *name*, *date-time*, *bid*, *ask* format,
- comma is used as the delimiter,
- dot is used as the decimal sign,
- *date-time* column is in format *rrrrmmdd HH:MM:SS.FFF*, where *dd* are days, *mm* are the month numbers, *yyyy* stands for years, *HH* are the hours value (in 24-hour format), *MM* minutes, *SS* second and *FFF* stands for milliseconds.

Knowing the structure of the files we can employ function *importfxdata* shown in Program 1–6 for data import.

Program 1–6 Function importfxdata.m

```
function [data] = importfxdata(filename)
%imports data from file specified as an imput
if exist(filename,'file')
  fid = fopen(filename);
  import=textscan(fid,'%s %s %f %f','delimiter', ',');
  data.name=import{1}{1};
  data.time=datenum(import{2},'yyyymmdd HH:MM:SS.FFF');
  data.bid=import{3};
  data.ask=import{4};
  fclose(fid);
else
  data=NaN;
end
end
```

However, as we have seen in Figure 1–2 the financial time series are split into months, i.e. one file contain one month of data. However, we generally want to work with the data history as long as possible to obtain.[17] In order to combine data from more than one file and import all the data files in one directory, the *importfxdirectory* function (see Program 1–7) can be applied. The input of the function is string variable specifying the directory in which the files are saved. Note that requirement of the function is that no other files (than those containing data and having specific format of TrueFX data source) can be saved in the specified directory. The function works in a simple way: first it obtains the list of files in the specified directory; then goes through the list and imports particular files applying function *importfxdata* (see Program 1–6).

Program 1–7 Function importfxdirectory.m

```
function [ data ] = importfxdirectory(directoryname)
% imports the whole directory
list=dir([directoryname '\*.csv']);
for a=1:length(list)
  lists{a}=[directoryname '\' list(a).name];
end
lists=sortrows(lists');


for a=1:length(lists)
  if a==1
    data=importdata(lists{1});
  else
    dataimported=importdata(lists{a});
    data.time=[data.time; dataimported.time];
    data.ask=[data.ask; dataimported.ask];
    data.bid=[data.bid; dataimported.bid];
  end;
end;
end
```

[17] This is generally true, however, when working on personal computer we also have to take into consideration computational and memory requirements.

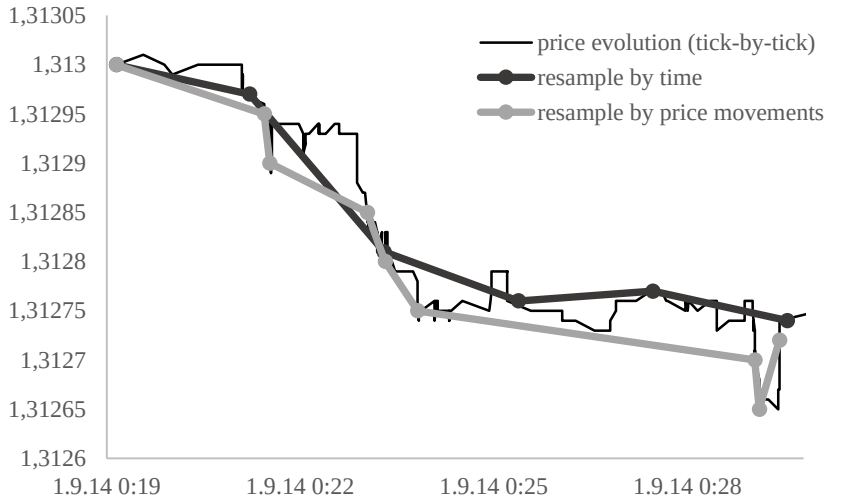

Figure 1–3 Price evolution and different types of resampling of EUR/USD FX pair

1.2 Data Resampling

While tick-by-tick data provide the most precise information about the evolution of the time series, they are also of huge size to store. Usually not all the information is needed and thus they can be compressed. The mostly utilized and well-known method is a time based resampling. Under this approach we take the values of the prices/values in equidistant time moments. However, we can do a price based resampling and instead of taking the fixed period, we fix the price changes between two subsequent points, i.e. the price can go either up or down by a given increment (threshold), see Figure 1–3.

In order to express the resampling mathematically, assume a continuous path $\left(t, X_t\right)_{t=0}^{T}$, which we want to approximate with a finite number of points $\left\{t_i, X_{t_i}\right\}$. Then, by means of time based resampling we obtain $\left\{i \cdot \Delta, X_{i\Delta}\right\}_{i=0}^{T/\Delta}$ and by means of price based resampling we obtain $\left\{\tau_i, X_{\tau_i}\right\}$, where $\tau_i = \inf\left\{t \geq \tau_{i-1} : \left|X_\tau - X_{\tau-1}\right| \geq \text{treshold}\right\}$.

As can be seen from the Figure 1–3 in both resampling methods some information is lost. In order not to discard important information in time based resampling, we usually save not only the prices at the equidistant time points, but also minimum and maximum prices between two subsequent points (in the period between them). By this way we obtain open/high/low/close (OHLC) prices. The typical period lengths are 1, 5 and 15 minutes, 1, 4 and 8 hours, a day, a week or a month.

1.2.1 Time Based Resampling

If we want to reduce the amount of data stored, we usually resample the data in time. This is practical reasoning as we are later usually concerned with the question what the price was at some particular moment of time. Then we look-up the price at the closest sampled time point. In Program 1–8 we included the function *resamplebytime* which can be applied to resample the tick-by-tick data obtained from TrueFX data source.

Program 1–8 Function resamplebytime.m

```
function [open,high,low,close]=resamplebytime(time,data,...
timeresampled)
% resamples given time series by time

open =zeros(1,length(timeresampled)-1);
high =zeros(1,length(timeresampled)-1);
low  =zeros(1,length(timeresampled)-1);
close=zeros(1,length(timeresampled)-1);

counter=1;
while (time(counter)<timeresampled(1))
    counter=counter+1;
end

for a=2:length(timeresampled)
    open(a-1)=data(counter);
    low(a-1)=open(a-1);
    high(a-1)=open(a-1);
    while (time(counter)<timeresampled(a))
        low(a-1) =min(low(a-1) ,data(counter));
        high(a-1)=max(high(a-1),data(counter));
        if (counter<length(time))
            counter=counter+1;
        else
            break;
        end;
    end;
    close(a-1)=data(counter);
end;
end
```

1.2.2 Resampling by Price Movements

However, for some applications it can be useful to undertake the different type of resampling. Assume for instance automated trading system, which trades (buys the asset) with predefined *take profit*[18] (henceforth TP) and *stop loss*[19] (henceforth SL). If the difference between TP and SL is small, for some periods (in time based resampling) the following situation can happen: the highest price is higher than TP

[18] The price level at which the trade is closed with the predefined profit. In the case of long trade position the take profit level is equal to the buy price plus minimum (predefined) profit.

[19] The price level at which the trade is closed in order to avoid big losses. In the case of long trade position the stop loss is equal to the buy price minus the maximum loss.

and lowest price is lower than SL. If we are in this period in the long position, we cannot know what happened first – whether the price went up and we closed the profitable trade or the price went down and the trade was closed with the loss. This loss of information can be important when we are backtesting the automated trading system with the small spread between take profit and stop loss.

In such case, the price based resampling would be more efficient. The price increment (threshold) should be chosen in order to be equal to the predefined value of minimum profit or maximum loss (or better to its fraction such as one third etc.). In Program 1–9 we included the function *resamplebyprice* which can be applied to resample the tick-by-tick data obtained from TrueFX data source (see subchapter 1.1.4).

Program 1–9 Function resamplebyprice.m

```
function [timesampled,datasampled]=resamplebyprice(time,data,...
starttime,tresholdup,tresholddown)
% resamples given time series by the price movements

counterdata=0; %counter for timesampled and datasampled time series
timesampled=zeros(length(time),1); %pre-alocation of variable
datasampled=zeros(length(data),1); %pre-alocation of variable

for counter=1:length(time)
    if (time(counter)<starttime) %exclude the part of time series
        continue;
    end
    %first observation (dependent on starttime)
    if (counterdata==0)
        counterdata=counterdata+1;
        timesampled(counterdata)=time(counter);
        datasampled(counterdata)=data(counter);
        continue;
    end;
    %if the increase is greater than treshold then add new
    if (data(counter)-datasampled(counterdata)>tresholdup)
        counterdata=counterdata+1;
        timesampled(counterdata)=time(counter);
        datasampled(counterdata)=data(counter);
        continue;
    end;
    %if the decrease is greater than treshold then add new
    if (data(counter)-datasampled(counterdata)<-tresholddown)
        counterdata=counterdata+1;
        timesampled(counterdata)=time(counter);
        datasampled(counterdata)=data(counter);
        continue;
    end;
end;
timesampled=timesampled(1:counterdata);
datasampled=datasampled(1:counterdata);
end
```

1.3 Combination of Time Series

In the previous subchapter we discussed the data import possibilities from various freely available sources. After running some of the import functions we obtain the one-dimensional data time series. However, for the further analysis it is necessary to combine these data-series into a matrix so that the particular rows represent the days and the columns represent particular assets.

Further we prepare three different datasets (i.e. matrices):

- a small indices dataset of three stock market indices (PX index, American Standard & Poor's 500 and Japanese Nikkei 225) denominated in CZK,
- DJIA dataset of the stocks incorporated in Dow Jones Industrial Average index, prices are denominated in USD,
- S&P 500 dataset of the stocks incorporated in Standard & Poor's 500 index, prices are denominated in USD.

As we combine different sources in the small indices dataset, the combination procedure is the most complex one. On the other hand, for DJIA and S&P 500 datasets only Yahoo Finance data source is utilized, so the combination of the data series is easier.

1.3.1 Small Indices Dataset

In the small indices dataset we want to combine the indices of US stock market, Prague stock market and Japanese stock market. As each index is denominated in different currency, we need to recalculate their values (i.e. the prices) to be in CZK. Thus we have to combine various data sources: PSE (Program 1–1), Yahoo Finance (Program 1–4) and CNB (Program 1–5), see Program 1–10.

In the first part of the program the particular time series are obtained. However, the most important part of the algorithm is the combination of obtained financial time series into one matrix. As these time series are of different length[20] the missing values have to be determined. In Program 1–10 we utilize linear interpolation by means of *interp1* function. Another approach would be to substitute missing values by the previous last known value. Although the second approach provides more correct combination of time series, the linear interpolation is simpler but sufficient.

The evolution of indices is depicted in Figure 1–4. In the figure the values of indices were normalized so that they all start at the level of one. Before discussing the figure, we should note that all the indices are price indices, i.e. the dividend payments are not included in the values of the indices. We should also note that the examined period is from January 1, 1994 until September 30, 2014. From the figure we can see the effect of dot-com bubble[21] with a climax on March 10, 2000 and global financial crisis in 2008. As can be seen, PX index was not hit by the

[20] Due to the different public holidays in US, Czech Republic and Japan, data are missing for some days.

[21] Also referred to as the dot-com boom, the internet bubble or the information technology bubble.

dot-com bubble at all, while S&P 500 and Nikkei 225 were hit seriously. On the other hand, both S&P 500 and Nikkei 225 have already managed to recover from the drop in 2008, while PX index has not. If we compare the final values of the indices, we can see that S&P 500 managed to triple its value, PX index increased by half and Nikkei 225 decreased by 20% over previous almost 20 years.

Program 1–10 Historical dataset of stock market indices

```
program_1_1; %import PX index history
index=(PX.dates>=datenum('1-Jan-1994'))&...
(PX.dates<=datenum('30-Sep-2014'));
PX.dates=PX.dates(index);%we want only period 1.1.1994-30.9.2014
PX.values=PX.values(index);%we want only period 1.1.1994-30.9.2014
%import S&P 500 index history
[SP500.dates,~,~,~,SP500.close,~,~]=downloadyahoo('^GSPC',...
'd',1994,1,1,2014,9,30);
%import Nikkei 255 index history
[N225.dates,~,~,~,N225.close,~,~]=downloadyahoo('^N225',...
'd',1994,1,1,2014,9,30);
%import USD/CZK FX rate history
[USD.dates,USD.fxrate]=downloadCNB('USD',1994,1,1,2014,9,30);
%import JPY/CZK FX rate history
[JPY.dates,JPY.fxrate]=downloadCNB('JPY',1994,1,1,2014,9,30);

dates=union(PX.dates,union(union(SP500.dates,USD.dates),...
union(N225.dates,JPY.dates)));
prices(:,1)=interp1(PX.dates,PX.values,dates,'linear','extrap');
prices(:,2)=interp1(SP500.dates,SP500.close,dates,'linear',...
'extrap');
prices(:,3)=interp1(USD.dates,USD.fxrate,dates,'linear','extrap');
prices(:,4)=interp1(N225.dates,N225.close,dates,'linear','extrap');
prices(:,5)=interp1(JPY.dates,JPY.fxrate,dates,'linear','extrap');

% convert the indices into CZK and normalize the data
prices=[prices(:,1) prices(:,2).*prices(:,3)...
prices(:,4).*prices(:,5)];
prices_normalized=(prices./repmat(prices(1,:),size(prices,1),1));
list={'PX','S&P 500', 'Nikkei 225'};

% plot the data
figure;
plot(dates, prices_normalized);
datetick('x');
xlabel('Date');
ylabel('Index value (relative)');
legend(list);
%% save data
returns=prices(2:end,:)./prices(1:end-1,:)-1;
save data_PX_SP500_N225;
```

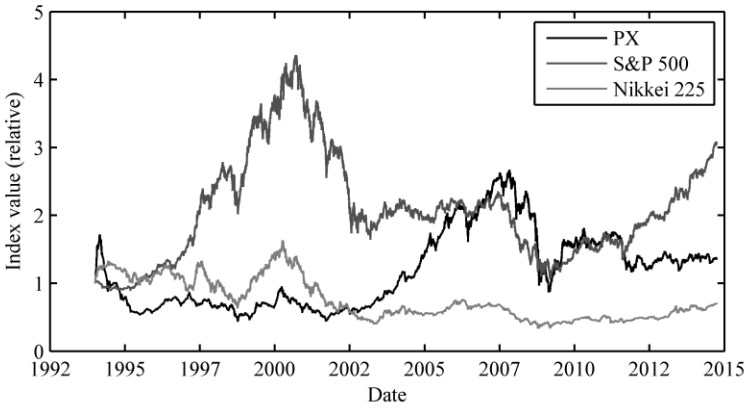

Figure 1–4 Evolution of indices in time

1.3.2 Dow Jones Industrial Average dataset

Another dataset we create consists solely of the stocks incorporated in one of the American stock market indices – Dow Jones Industrial Average (henceforth DJIA). The components of the index are listed in Appendix B. The dataset we want to create should cover the period from January 1, 1991 until September 30, 2014. Due to the lack of the historical data we do not include into the dataset the data of The Goldman Sachs Group, Inc. (Yahoo Finance ticker *GS*) and Visa Inc. (Yahoo Finance ticker *V*). Thus, the dataset consists of only the remaining 28 stocks.

The practical implementation is depicted in Program 1–11. The program cycles through the list of remaining DJIA components, *DJIA_list*, downloading each of them by function *downloadyahoo* (Program 1–4). Due to the fact that all the time series are of the same length they can be simply combined into the matrix *prices* without dealing with the missing data. At the end of the program discrete returns are calculated and all the data are saved to the file *data_DJIA.mat*.

Program 1–11 Historical dataset of the stocks incorporated in DJIA (as of October 6, 2014)

```
DJIA_list={'MSFT', 'CSCO', 'INTC', 'PFE', 'GE', 'T', 'JPM', 'KO',
'VZ', 'XOM', 'MRK', 'DIS', 'JNJ', 'PG', 'CVX', 'HD', 'WMT', 'AXP',
'NKE', 'DD', 'MCD', 'CAT', 'MMM', 'TRV', 'UNH', 'BA', 'IBM',
'UTX'};
for a=1:length(DJIA_list)
  tic;
  [dates,~,~,~,~,~,prices(:,a)]=downloadyahoo(DJIA_list{a},...
'd',1991,1,1,2014,9,30);
  fprintf(['Symbol ' DJIA_list{a} ' imported in %5.2f'...
'seconds.\n'],toc);
end
returns=prices(2:end,:)./prices(1:end-1,:)-1;
clear a; save data_DJIA;
```

1.3.3 Standard & Poor's 500 Dataset

In the same way as DJIA dataset, we can obtain S&P 500 dataset – the historical data of the stocks incorporated in the Standard & Poor's 500 index. The program for data acquisition and combination is depicted in Appendix C.

The program is improved version of the Program 1–11. At the beginning of the algorithm we start with the full list of 501 stock symbols and cycle trough them. For each symbol we download the data for the period from September 30, 2004 until September 30, 2014.[22] If the historical data are downloaded for the whole period (i.e. we obtain 2,518 daily observations) the stock is included in the dataset. If there are missing data, the stock is ommited from the dataset.

By this way, the following 53 stocks (tickers) were omitted from the dataset: ABBV, ADT, ALLE, AMP, AVGO, CFN, CBS, CF, CMG, COV, DLPH, DAL, DFS, DISCA, DISCK, DG, DPS, EXPE, FB, FSLR, GM, GS, GOOG, ICE, KIM, KMI, KRFT, LO, LYB, MNK, MPC, MA, MJN, NAVI, NWSA, NLSN, KORS, PM, PSX, QEP, SNI, SE, TEL, TDC, TWC, TRIP, UA, VIAB, V, WU, WIN, WYN, XYL, ZTS. The remaining stocks, i.e. 448 stocks, build-up the dataset.

Finally we obtained the matrix in which we have 2,518 daily price observations (rows) of 448 stocks (columns). From the prices we calculate the discrete returns and save the data into the file *data_SP500.mat*.

[22] The length of the period was chosen as the compromise between maximum length and quantity of stocks without missing data over the selected period.