MATLAB applications of trading rules and GARCH with wavelets analysis

Eleftherios Giovanis

Abstract

In this paper we provide MATLAB routines for two major used trading rules, the moving average indicator and MACD oscillator as also the GARCH univariate regression with Monte Carlo simulations and wavelets decomposition, which is an update of an older algorithm.

Keywords: MATLAB, moving average, MACD, trading rules, technical analysis, GARCH, wavelets, Monte-Carlo

1. Introduction

We won't get to the traditional literature reviews of what someone examines what the other one studies or someone else found or proposes an excellent model. There isn't the excellent model and if someone wants to read fancy words and huge referees in literature this is not the paper you want to read. We just present some MATLAB functions for use in Forex trading and to experiment with them. Unfortunately some of these techniques haven't yet been introduced in universities and the economic and finance departments, because some of them have been developed by practitioners and real financial traders and not academicians, and because they haven't been published the academicians. Since only the last year have been acknowledged and technical analysis can really lead to profits than the traditional econometric models of random walk, ARIMA, GARCH and others. As for the other procedures academicians claim that are very difficult to learn them. Anyway economist's academicians around the world claim that they are the leaders in social sciences, because they use models and as also claim that the differential or autoregressive econometric models, with a magical way can learn. How these models can learn. You won't get any answer or the answers will be childish. Unfortunately they haven't seen the progress of psychology or sociology scientists who use neural networks and genetics and other advanced models. These are the truly leaders of social sciences.

* I would like to give many thanks to Andrea Durelli for his important note to correct a wrong which was existed in the initial routine

2. Methodology

2.1. Moving Average

The simple moving average is defined as:

$$\mathrm{SMA}_t = (1/L)\sum_{i=0}^{L-1} p_{t-i} \tag{1}$$

Relation (1) can be used as a trading rule, which generates a buy signal when the current stock price is higher or above the moving average and a sell signal when it is below. Relation (1) is the simple moving average, while there are also additional modifications in moving average, as the exponential, the square root weighted, the weighted or the linear moving average. The results among the various modifications of moving average as usually the same, as also many financial traders claim that.

2.2 Moving Average Convergence / Divergence (MACD)

The MACD oscillator is computed by subtraction of the 26-period exponential moving average with the 12-period exponential moving average. In the first case the smoothing factor a is 0.075 and in the second case is 0.15 Then the 9-period exponential moving average of MACD, with smoothing factor α equal with 0.20, is used as the signal line. So if the MACD and 9-period moving average lines are crossed and if the MACD line falls below the other, then a sell signal is generated, while in the opposite situation we have a buy signal. More specifically is defined as:

$$\mathrm{MACD}_t = EMA_{t\text{-}12} - EMA_{t\text{-}26} \tag{2}$$

If MACD line rises above the 9-period EMA line then buy

Else If MACD line falls below the 9-period EMA line then sell

2.3 Moving Average Convergence / Divergence (MACD)

In this part we present the computational algorithm we propose in a simple manner. The steps of the computation are:

a) First we decompose data applying discrete wavelet transformation (DWT) with Daubechies wavelets on the initial stock returns (Daubechies, [1994]). More specifically the DWT of a signal *X* for 1 level is defined as:

$$y_{highpass}[n] = \sum_{\mathrm{k}=-\infty}^{\infty} X[k]g[2n - \mathrm{k}] \tag{3}$$

$$y_{lowpass}[n] = \sum_{\mathrm{k}=-\infty}^{\infty} X[k]h[2n - \mathrm{k}] \tag{4}$$

,where *g* and *h* denote the impulse response and the filter outputs are sub-sampled by two. The final convolution is:

$$y_{highpass} = X \cdot g \downarrow 2 \tag{5}$$

$$y_{lowpass} = X \cdot h \downarrow 2 \tag{6}$$

The DWT is computed by successive lowpass and highpass filtering of the discrete-time domain signal. Because the data sample might varies usually data of 4000-5500 and data length above 6000 and for short periods forecasts as 10-20, then a decomposition level 4-7 might be more appropriate. For longer predicted lengths 1-3 decomposition level might be better. There is not a specific theory, proof, to show the appropriate used level, you have to make your own tests, but even the empirical practice is much better of the theory or what we are learning in education, who they never test the model, that they teach, even in trial account or in a hobby way.

b) The second step is to estimate a GARCH (1,1) process where the mean and variance equations are presented in relations (10) and (12) respectively

$$R_t = c + \varepsilon_t \tag{7}$$

$$\varepsilon_t \sim (0, \sigma^2) \tag{8}$$

$$\sigma_t^{\,2} = \omega + a_0 u_{t-1}^{\,2} + a_1 \sigma_{t-1}^{\,2} \tag{9}$$

, where R_t is defined as stock returns, c is the constant and ε_t is the disturbance term which follow the distribution in (8). Parameter ω is the constant of the variance equation, α_0 is the coefficient of ARCH effects and similarly α_1 is the coefficient of ARCH effects.

c) The third step is to simulate the residuals or the innovations of GARCH (1,1) estimation with Monte-Carlo. First we should set up the parameters for the simulation process. First we should define the random number generator seed for reproducibility, which is set up at this way that every time that we run the algorithm the generator is reset to its initial state. Because we have long sample we set up the simulated samples roughly 4 times greater than the initial data. So for data ranging 4,600 through 6,500 we obtain a simulated sample equal with 20,000 and similarly for data around 10,000 we take a sample equal with 40,000.

d) The final step is to take a random permutation sample with length equal with the initial data sample and then this sample is selected from the residuals or innovations after a number of replications.

If the predicted value is positive then we buy

Else If the predicted value is negative then we sell

In the fourth method we get only the moving average of the wavelet decomposition in the initial data and then we compare with the actual as in moving average trading rule

2.4 Estimation of Net Profits-Losses

In order to test which of the approaches we examine is the most profitable we apply the following formula procedure to compute the net profits or losses. We discriminate three cases in stock trading. The first one is the case when the stock returns are positive, the second one when are zero and finally when stock returns are negative. We define a dummy variable D_{t+1}.

If $r_{t+1} > 0$ then $D_{t+1} = 1$ (10)

If $r_{t+1} = 0$ then $D_{t+1} = D_t$ (11)

If $r_{t+1} < 0$ then $D_{t+1} = -1$ (12)

The net profits-losses are given by relation:

$$R_{t+1} = D_{t+1} \cdot (P_{t+1} - P_t) - c(|D_{t+1} - D_t| \cdot P_t) \quad (13)$$

Variables D_{t+1} and D_t are defined as previously which is the forecasting and the current signal respectively, variables P_{t+1} and P_t are the future and current stock prices respectively and c is the commission rate which is charged for the services of trading and depending on whether it's an intraday trading or electronic trading or depending on the different rates offered by various companies and financial trading institutions in different countries. Usually the commissions rates can be varied between 0.008 and 0.01, but nowadays in the *Forex* trading the most companies don't charge any commission rate.

References

Daubechies, I. Ten Lectures on Wavelets, CBMS-NSF Regional Conference Series in Applied Mathematics Lecture Notes, 61, Philadelphia, Pennsylvania, Society for Industrial and Applied Mathematics, 1992

MATLAB routine (script file)

```
clear all;
% Load input data
load file.mat



method=3  % 1 for moving average, 2 for MACD, 3 for GARCH and wavelets where
          % gives a 'buy' signal for positive predicted values and 'sell' signal for
          % negative values, 4 for the same with 3 but use moving average of
          % predicted values, 5,6,7 and 8 the same with 1,2,3 and 4 respectively but
          %for real application and not for testing or backtesting.

lag=50    % Define the length of moving average. Usually 10,20,30,50, 70,100 and
          % 200 are used.


factor='e' % 0 for simple, 0.5 for square root weighted moving average,
       % 1 for linear moving average, 2 for square weighted moving average
       % end 'e' for exponential

risk_free=0.001

length_test=100 % length of sample for testing. This script is used for testing. If you
                % wish to apply for future purposes set up the value of length_test=0

decomposition_tree=1 % lenght of decomposition tree
M=length_test; % length of predicted data

if method==1


train_data=data(1:end-length_test-1,:)
[Short,Long]=movavg(train_data,1,lag,factor)


test_sample=data(end-length_test:end,:)
test_long=Long(end-length_test-1:end,:)
stock=data(end-length_test-1:end,:)


[nk1,ni]=size(test_sample)
for kk=1:nk1

   if test_sample(kk,:)>test_long(kk,:)
      s(kk,:)=1                        % buy


   elseif test_sample(kk,:)<test_long(kk,:)
```

```
      s(kk,:)=-1                          %sell
   end
end

for jj=2:nk1
   total(jj,:)=s(jj)*(stock(jj)-stock(jj-1))-0.001*(abs(s(jj)-s(jj-1))*stock(jj))
end

profit=sum(total)

average=mean(total)
standard_deviation=std(total)

sharpe_ratio=(average)/standard_deviation



elseif method==2

train_data=data(1:end-length_test-1,:)
[macdvec, nineperma] = macd(train_data)
test_sample=macdvec(end-length_test:end,:)
test_macd=nineperma(end-length_test:end,:)
stock=data(end-length_test-1:end,:)

[nk1,ni]=size(test_sample)
for kk=1:nk1

   if test_sample(kk,:)>test_macd(kk,:)
      s(kk,:)=1          % buy


   elseif test_sample(kk,:)<test_macd(kk,:)
      s(kk,:)=-1          % sell
   end
end

for jj=2:nk1
   total(jj,:)=s(jj)*(stock(jj)-stock(jj-1))-0.001*(abs(s(jj)-s(jj-1))*stock(jj))
end

profit=sum(total)

average=mean(total)
standard_deviation=std(total)

sharpe_ratio=(average)/standard_deviation
```

```
elseif method==3
y=price2ret(data)
N=length(y)
% We decompose our data with function db3
[XX,l] = wavedec(y,decomposition_tree,'db3');
% We define GARCH (1,1) process
[Kappa, Alpha, Beta] = ugarch(XX, 1, 1)
% We set the random number generator seed for reproducability
randn('state', 0)
NumSamples = 20000;
firstpoint=length_test
% We simulate the process with Monte Carlo
[U , H] = ugarchsim(Kappa, Alpha, Beta, NumSamples);
% Length of vector
%V=1%length(data);
% From current day we extract firstpoint data randomly selected
currentprice = randperm(N-M);
currentprice= currentprice+N;
for j=1:firstpoint
Y1 = currentprice(j);
Y0 = Y1-N+1;
p = U(Y0:Y1);
p = p(:);
Y1(1,:) = p(1,:);
prediction = U(Y1+1:Y1+M);
end

[nk1,ni]=size(prediction)
for kk=1:nk1

   if prediction(kk,:)>0
      s(kk,:)=1        % buy


   elseif prediction(kk,:)<0
      s(kk,:)=-1       % sell
   end
end
stock=data(end-length_test:end,:)
for jj=2:nk1
   total(jj,:)=s(jj)*(stock(jj)-stock(jj-1))-0.001*(abs(s(jj)-s(jj-1))*stock(jj))
end

profit=sum(total)

average=mean(total)
standard_deviation=std(total)

sharpe_ratio=(average)/standard_deviation
```

```
elseif method==4

N=length(data)

% We decompose our data with function db3
[XX,l] = wavedec(data,decomposition_tree,'db3');


train_data=XX(1:end-length_test-1,:)
[Short,Long]=movavg(train_data,1,lag,factor)


test_sample=data(end-length_test:end,:)
test_long=Long(end-length_test-1:end,:)
stock=data(end-length_test-1:end,:)


[nk1,ni]=size(test_sample)
for kk=1:nk1

   if test_sample(kk,:)>test_long(kk,:)
      s(kk,:)=1         % buy

   elseif test_sample(kk,:)<test_long(kk,:)
      s(kk,:)=-1       % sell

   end
end

for jj=2:nk1
   total(jj,:)=s(jj)*(stock(jj)-stock(jj-1))-0.001*(abs(s(jj)-s(jj-1))*stock(jj))
end

profit=sum(total)

average=mean(total)
standard_deviation=std(total)

sharpe_ratio=(average)/standard_deviation


elseif method==5


 [Short,Long]=movavg(data,1,lag,factor)

if data (end,:)> Long (end,:)
       s=1         % buy
```

```
   elseif data (end,:)< Long (end,:)
      s=-1       % sell

   end


elseif method==6

[macdvec, nineperma] = macd(data)

if macdvec (end,:)> nineperma (end,:)
      s=1         % buy

   elseif macdvec (end,:)< nineperma (end,:)
      s=-1       % sell

   end


elseif method==7

y=price2ret(data)
N=length(y)
% We decompose our data with function db3
[XX,l] = wavedec(y,decomposition_tree,'db3');
% We define GARCH (1,1) process
[Kappa, Alpha, Beta] = ugarch(XX, 1, 1)
% We set the random number generator seed for reproducability
randn('state', 0)
NumSamples = 20000;
firstpoint=length_test
% We simulate the process with Monte Carlo
[U , H] = ugarchsim(Kappa, Alpha, Beta, NumSamples);
% Length of vector
%V=1%length(data);
% From current day we extract firstpoint data randomly selected
currentprice = randperm(N-M);
currentprice= currentprice+N;
for j=1:firstpoint
Y1 = currentprice(j);
Y0 = Y1-N+1;
p = U(Y0:Y1);
p = p(:);
Y1(1,:) = p(1,:);
prediction = U(Y1+1:Y1+M);
end

if prediction>0
      s=1        % buy
```

```
   elseif prediction<0
      s=-1        % sell
   end



elseif method==8

N=length(data)

% We decompose our data with function db3
[XX,l] = wavedec(data,decomposition_tree,'db3');


 [Short,Long]=movavg(XX,1,lag,factor)



   if data(end,:)> Long (end,:)
      s=1         % buy

   elseif data (end,:)< Long (end,:)
      s=-1       % sell

   end
end
```