Smoothing Transition Autoregressive (STAR) Models with Ordinary Least Squares and Genetic Algorithms Optimization

Giovanis, Eleftherios

10 August 2008

Smoothing Transition Autoregressive (STAR) Models with Ordinary Least Squares and Genetic Algorithms Optimization

Eleftherios Giovanis

Abstract

In this paper we present, propose and examine additional membership functions as also we propose least squares with genetic algorithms optimization in order to find the optimum fuzzy membership functions parameters. More specifically, we present the tangent hyperbolic, Gaussian and Generalized bell functions. The reason we propose that is because Smoothing Transition Autoregressive (STAR) models follow fuzzy logic approach therefore more functions should be tested. Some numerical applications for S&P 500, FTSE 100 stock returns and for unemployment rate are presented and MATLAB routines are provided.

Keywords: Smoothing transition; exponential, logistic; Gaussian; Generalized Bell function; tangent hyperbolic; stock returns; unemployment rate; forecast; Genetic algorithms; MATLAB

1. Introduction

In the 1950s and the 1960s several computer scientists independently studied evolutionary systems with the idea that evolution could be used as an optimization tool for engineering problems. The idea in all these systems was to evolve a population of candidate solutions to a given problem, using operators inspired by natural genetic variation and natural selection. Genetic Algortihms (GA) is a method for moving from one population of "chromosomes" e.g., strings of ones and zeros, or "bits", to a new population by using a kind of "natural selection" together with the genetics−inspired operators of crossover, mutation, and inversion. Each chromosome consists of "genes", each gene being an instance of a particular "allele" (e.g., 0 or 1). The selection operator chooses those chromosomes in the population that will be allowed to reproduce, and on average the fitter chromosomes produce more offspring than the less fit ones. Crossover exchanges subparts of two chromosomes, roughly mimicking biological recombination between two single−chromosome organisms. Mutation randomly changes the allele values of some locations in the chromosome.

We propose some additional fuzzy membership functions as well the tangent hyperbolic function, which is used in neural networks with some appropriate modifications. We do not present the process and the linearity tests or the tests choosing either exponential or logistic smoothing functions. Additionally we apply

ordinary least squares with genetic algorithms in order to compute and choose the parameters of fuzzy membership functions.

2. Methodology

The smoothing transition auto-regressive (STAR) model was introduced and developed by Chan and Tong (1986) and is defined as:

$$y_t = \pi_{10} + \pi_1' w_t + (\pi_{20} + \pi_2' w_t) F(y_{t-d}; \gamma, c) + u_t \quad (1)$$

,where $u_t \sim (0,\sigma^2)$, π_{10} and π_{20} are the intercepts in the middle (linear) and outer (nonlinear) regime respectively, $w_t = (y_{t-1}, \ldots y_{t-j})$ *is* the vector of the explanatory variables consisting of the dependent variable with $j=1 \ldots p$ lags, y_{t-d} is the transition variable, parameter c is the threshold giving the location of the transition function and parameter γ is the slope of the transition function. The membership functions we examine are exponential, logistic, tangent hyperbolic, generalized bell function and Gaussian defined by (2)-(6) respectively.

$$\mu_{ij}(x_j; \gamma_{ij}, c_{ij}) = \frac{1}{1 + e^{-\gamma_{ij}(x_j - c_{ij})}} \quad (2)$$

$$\mu_{ij}(x_j; \gamma_{ij}, c_{ij}) = 1 - e^{-\gamma_{ij}(x_j - c_{ij})^2} \quad (3)$$

$$\mu_{ij}(x_j; \gamma_{ij}, c_{ij}) = \frac{2}{(1 + e^{-2\gamma_{ij}(x_j - c_{ij})}) - 1} \quad (4)$$

$$\mu_{ij}(x_{ij}; a_{ij}, b_{ij}, c_{ij}) = \exp\left(-\left(\frac{x_{ij} - c_{ij}}{\gamma_{ij}}\right)^{2b_{ij}}\right) \quad (5)$$

$$\mu_{ij}(x_j; c_{ij}, \sigma_{ij}) = \exp\left(-\frac{(x_j - c_{ij})^2}{2\gamma^2_{ij}}\right) \quad (6)$$

The STAR model estimation is consisted by three steps according to Teräsvirta (1994).

a) The specification of the autoregressive *(AR)* process of $j=1, \ldots p$. One approach is to estimate *AR* models of different order and the maximum value of j can be chosen based on the *AIC* information criterion Besides this approach, j value can be selected

by estimating the auxiliary regression (7) for various values of $j=1,...p,$ and choose that value for which the *P-value* is the minimum, which is the process we follow.

b) The second step is testing linearity for different values of delay parameter d. We estimate the following auxiliary regression:

$$R_t = \beta_o + \beta_1 w_t + ... + \sum_{j=1}^{p} \beta_{2j} R_{t-j} R_{t-d} + \sum_{j=1}^{p} \beta_{3j} R_{t-j} R_{t-d}^{\ 2} + \sum_{j=1}^{p} \beta_{4j} R_{t-j} R_{t-d}^{\ 3} + \varepsilon_t \tag{7}$$

The null hypothesis of linearity is H_0: $\beta_{2j} = \beta_{3j} = \beta_{4j} = 0$. In order to specify the parameter d the estimation of (7) is carried out for a wide range of values $1 \leq d \leq D$ and we choose $d=1,...,6$ In the cases where linearity is rejected for more than one values of d, then d is chosen by the minimum value of $p(d),$ where $p(d)$ is the *P-value* of the linearity test. After we find order p and d we estimate 1 with ordinary least squares and we minimize function (8) in order to find the optimum parameters.

$$\frac{1}{2} mse\ (y - y^t) = \frac{1}{2} mse\ (e) \tag{8}$$

, where y^t is the target-actual, y is network's output variable and *mse* is the mean squared error The steps of genetic algorithms are (Bäck, 1996; Mitchell, 1996)

1. Start with a randomly generated population of n-bit chromosomes, which are the candidate solutions. In the case we do not use bit or binary encoding, but we use real number encoding based on the range of the input data. The chromosomes are equally with the number of weights for both input-to-hidden layer and hidden-to-output layer.
2. Calculate the fitness $f(x)$ of each chromosome x in the population
3. Repeat the following steps until n offspring have been created:
    a. select a pair of parents chromosomes of the current population and compute the probability of selection being an increasing function of fitness. In this case we take the roulette wheel selection algorithm. Also the selection process is one with replacement meaning that the same chromosome can be selected more than once to become a parent.
    b. The next step is the crossover. We use one-point crossover process with probability p_c cross over the pair at a chosen point. If no crossover

takes place we form two offspring that are exact copies of their respective parents.

c. Mutate the two offspring with probability p_m and place the resulting chromosomes in the new population.

4. Replace the current population with the new population.
5. Go to step 2.

We choose only 10 iterations for faster computation time. The population size is 30. It should be noticed that genetic algorithms is a random process so it not absolutely always a good approach. But before we reject something, which unfortunately happens very often, we should try it.

3. Data

In the first example we examine two stock index returns, S&P 500 and FTSE 100 in daily frequency for the period March to December of 2009. The last 20 trading days are left for out-of-sample forecasts of the test period. In the second application example we examine the unemployment rate of USA for period 1995-2009 in monthly frequency and 2009 is left for testing.

4. Empirical results

We take and AR(1) for both S&P 500 and FTSE 100, while we take1 and 2 lags for transition function for S&P 500 and FTSE 100 respectively. The interval for parameters c, γ and b are respectively, [-0.03 0.03], [1 5] and [0.5 2]. The correct percentage sign for stock index returns are reported in table 1, where you can make your own conclusions. Additionally, for example in figures 1 and 2 we present the in-sample forecasts for FTSE 100 with TSTAR and GARCH, which the last one is mainly used. Even if we take a very long sample for GARCH, because of the statistic properties, the forecasts will be again a “dead-line”.

Table 1. Correct percentage sign for S&P 500 and FTSE 100

Indices	ESTAR	LSTAR	TSTAR	GBELL_STAR	GAUSS_STAR
S&P 500	70.00	70.00	70.00	75.00	70.00
c	-0.0238	-0.0401	-0.0213	-0.0454	-0.0498
γ	4.9277	3.6929	2.1523	3.1188	1.1740
b				1.8680	
p_c	0.20	0.20	0.20	0.20	0.20
p_m	0.05	0.01	0.01	0.005	0.001
FTSE 100	60.00	55.00	55.00	60.00	75.00
c	-0.0080	-0.0302	-0.0077	-0.0170	-0.0592
γ	3.5069	2.3413	3.4949	2.8428	1.1752
b				2.7847	
p_c	0.20	0.20	0.20	0.20	0.20
p_m	0.01	0.01	0.008	0.001	0.001

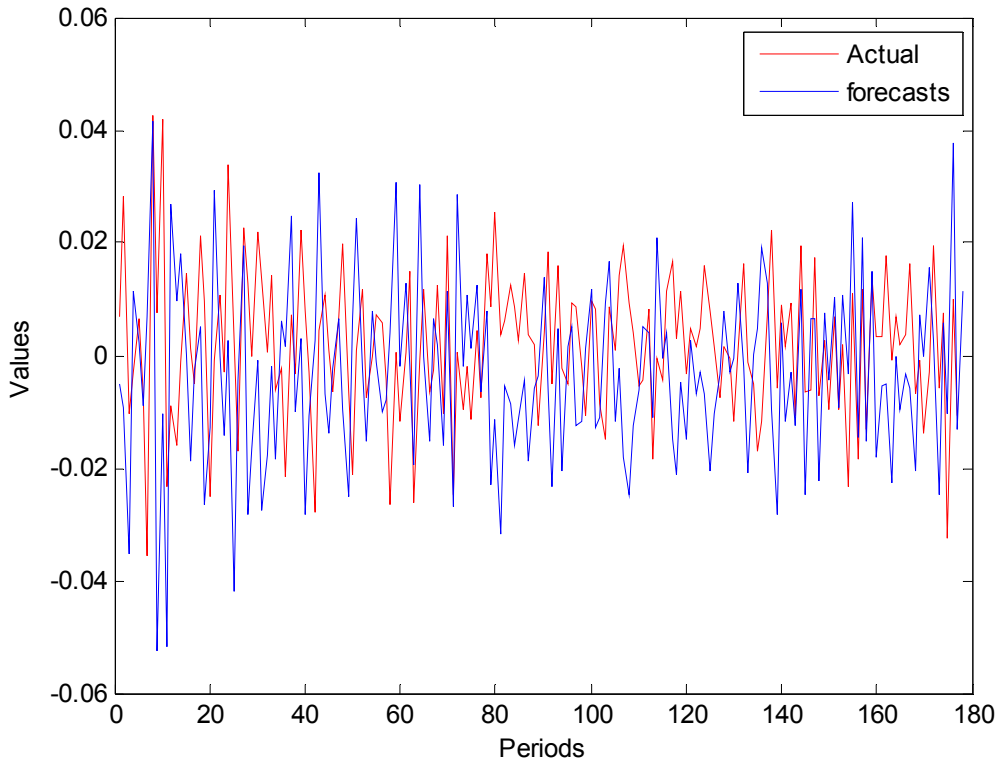


Fig. 1 In-sample forecasts for FTSE with TSTAR

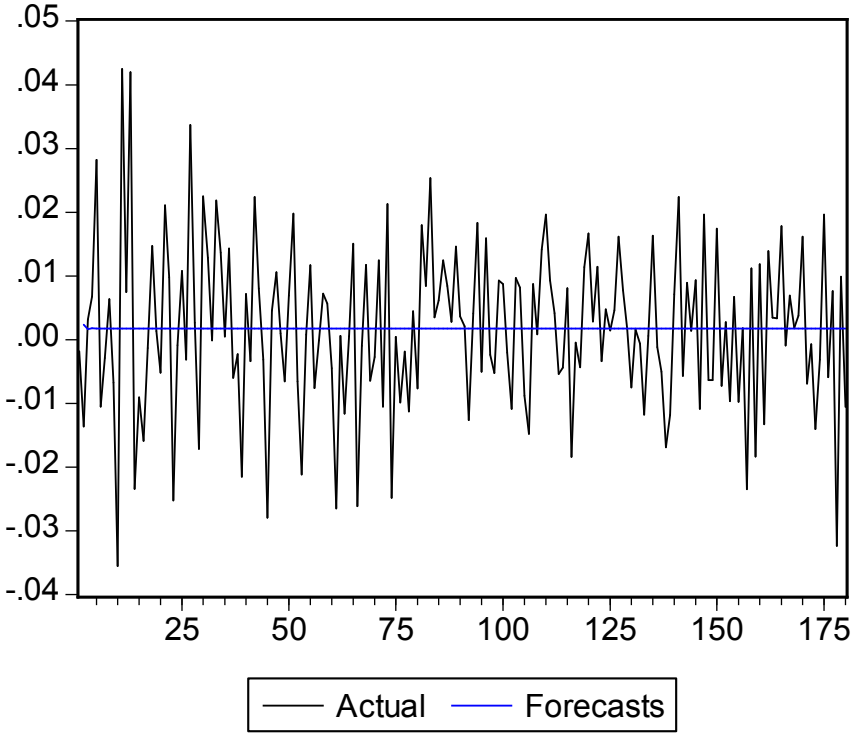


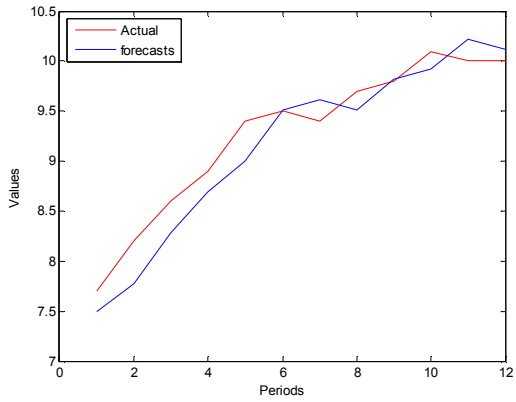
Fig. 2 In-sample forecasts for FTSE 100 with GARCH (1,1)

In the last example we examine the unemployment rate of USA during period 1995-2009 and year 2009 is left for forecasting, but for one-step ahead predictions. We estimate for AR(1) and we take delay lag order 1 for transition function. In table 2 we report the estimated results. The procedure for population initialization is the same with that we took in stock returns, except from ESTAR and LSTAR, where the initialization and the optimum values of parameters c and γ are based on the minimum and maximum values of inputs, which is the dependent variable with one lag. Additionally, it might be more appropriate to take the first differences for unemployment rate, because there is possibility to reject stationarity, but it is just an example in order to encourage the use and examination of alternatives procedures, as in any cases, as the practitioners and professionals know better, the conventional econometric modelling have failed in many cases. For literature review there are many case studies.

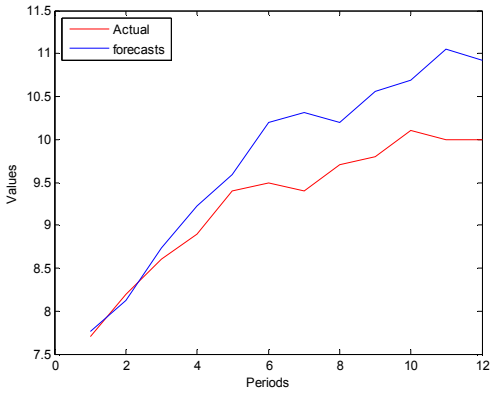
Table 2. Estimating results for US unemployment rate

Model	**Linear part**		**Nov-Linear part**	
ESTAR	π_{10}	π_{11}	π_{20}	π_{21}
	0.9394 (-1.4222)	1.1615 (10.364)*	0.8348 (1.2899)	-0.1374 (-1.7396)***
c	5.9593		p_c	0.2
γ	4.8039		p_m	0.01
LSTAR	π_{10}	π_{11}	π_{20}	π_{21}
	0.1041 (0.4057)	0.9798 (16.756)*	-0.5688 (-1.7543)***	0.1057 (1.6578)***
c	5.0330		p_c	0.2
γ	6.2709		p_m	0.01
TSTAR	π_{10}	π_{11}	π_{20}	π_{21}
	0.0159 (0.0661)	0.9943 (18.425)*	-1.2196 (-1.1011)	0.0458 (1.9928)**
c	4.6103		p_c	0.2
γ	5.9207		p_m	0.01
GBELL_STAR	π_{10}	π_{11}	π_{20}	π_{21}
	-0.5497 (-1.0424)	1.1583 (13.644)*	5.5001 (2.0908)**	-1.4226 (-2.1354)**
c	-0.0206		p_c	0.2
γ	4.6613		p_m	0.01
b	2.8522			
GAUSS_STAR	π_{10}	π_{11}	π_{20}	π_{21}
	0.5338 (2.4229)**	0.7811 (8.3290)*	-1.5062 (-1.8918)***	-0.4674 (2.2641)**
c	-0.0222		p_c	0.2
γ	3.7881		p_m	0.01

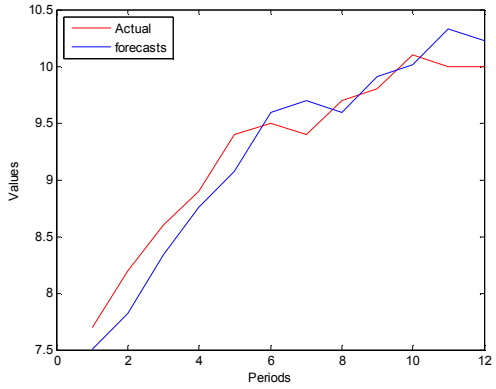
t-student in parentheses, *, ** and *** indicate statistical significant in 0.01, 0.05 and 0.10 respectively.



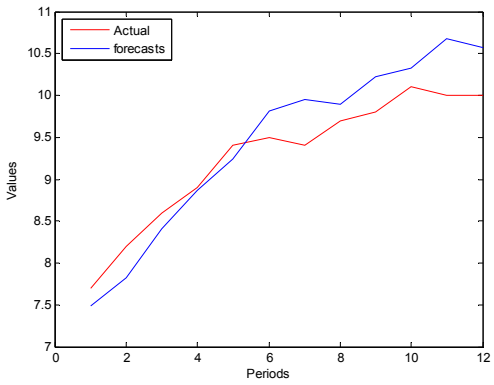
(a)



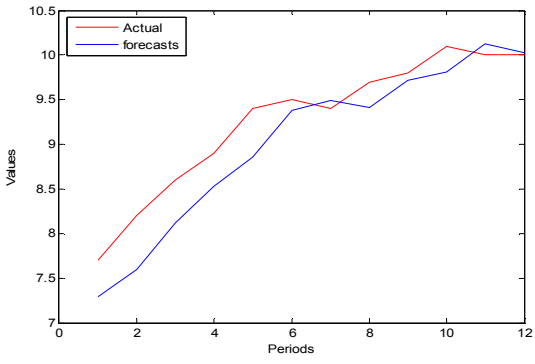
(b)



(c)



(d)



(e)

Fig. 3 Out-of-sample forecasts for US unemployment rate with a) ESTAR, b)LSTAR, c) TSTAR, d)GBELL_STAR and e)GAUSS_STAR

Conclusions

In this paper we proposed three additional membership functions for STAR modelling as also a very simple approach for computing the membership functions parameters using genetic algorithms. More functions can be used as the triangular, trapezoidal or s-shaped among others, as also fuzzy rules can be obtained on order to improve the estimations concerning the imprecision.

References

Bäck, T. (1996). *Evolutionary Algorithms in Theory and Practice*. Oxford University Press

Chan, K.S. and Tong, H. (1986). On estimating thresholds in autoregressive models. *Journal of Time Series Analysis*, Vol. 7, pp. 178-190

Janikow, C. Z., and Michalewicz, Z. (1991). An experimental comparison of binary and floating point representations in genetic algorithms. In R. K. Belew and L. B. Booker, eds., *Proceedings of the Fourth International Conference on Genetic Algorithms,* Morgan Kaufmann Publishers

Mitchell, M. (1996). *An Introduction to Genetic Algorithms*. MIT Press Cambridge, Massachusetts, London, England

Teräsvirta, T. (1994). Specification, Estimation, and Evaluation of Smooth Transition Autoregressive Models. *Journal of the American Statistical Association*, Vol. 89, No. 425, pp. 208–218.

Appendix

MATLAB routine
Smoothing transition functions with ordinary least squares and genetic algorithms optimization

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

nforecast=20
y=data(1:end-nforecast,1)
t=length(y)

d=2;
x=lagmatrix(y,d)
iii=1;
stdev=std(y)

%Some initial values for parameters c, gamma and b
c=0
gamma=1
be=2
% choose STAR function
model=5

% Set up the population size
popsize=30;

pc=0.2;
pm=0.01;

lb_c= 0.03 % bounds of the parameters c to be optimized
ub_c=-0.03

lb_gamma= 1 % bounds of the parameters gamma to be optimized
ub_gamma=5

lb_b=0.5 % bounds of the parameters b to be optimized
ub_b=2



 % Set up random population for c
Range_c = repmat((ub_c-lb_c),[popsize 1]);
Lower_c = repmat(lb_c, [popsize 1]);
pop_c= rand(popsize,1) .* Range_c + Lower_c;

% Set up random population for gamma
Range_gamma = repmat((ub_gamma-lb_gamma),[popsize 1]);
Lower_gamma = repmat(lb_gamma, [popsize 1]);
pop_gamma= rand(popsize,1) .* Range_gamma + Lower_gamma;

 % Set up random population for b
Range_b = repmat((ub_b-lb_b),[popsize 1]);
Lower_b = repmat(lb_b, [popsize 1]);
pop_b= rand(popsize,1) .* Range_b + Range_b;
```

```
% For ESTAR, GBELL_STAR and GAUSS_STAR is better to initialize as
%Range_c = repmat((ub_c-lb_c),[popsize 1]);
%pop_c= rand(popsize,1) .* Range_c

 %Range_gamma = repmat((ub_gamma-lb_gamma),[popsize 1]);
%pop_gamma= rand(popsize,1) .* Range_gamma

if model==4
 pop=[pop_c pop_gamma pop_b]
else
pop=[pop_gamma pop_c]
end

Chromosome=pop
for p=1:iii
ylag(:,p)=lagmatrix(y,p)
end
te=length(ylag)

ylag=ylag(iii+d+1:t,:)
y=y(iii+d+1:t,:)
x=x(iii+d+1:t,:)
for iterations =1:10
 [nkk,nii] = size(pop);
    Chromosome = pop;
    for jj = 1:nkk


X1=[ones(size(y)) ylag]
if model==1                % ESTAR (exponential)
ESTAR_1= 1-exp(-Chromosome(jj,1)./stdev.*((x-Chromosome(jj,2)).^2))
Y=X1

for p=1:iii+1
h(:,p)=ESTAR_1.*Y(:,p)
end


elseif model==2                % LSTAR (logistic)
LSTAR_1=1./(1+exp(-Chromosome(jj,1).*(x-Chromosome(jj,2))))
Y=X1

for p=1:iii+1
h(:,p)=LSTAR_1.*Y(:,p)
end

elseif model==3                % TAHN_STAR (tangent hyperbolic)

TSTAR_1=(2./(1+exp(-2*Chromosome(jj,1).*(x-Chromosome(jj,2))))-1)
Y=X1

for p=1:iii+1
h(:,p)=TSTAR_1.*Y(:,p)
end

elseif model==4              % Generalized Bell function
GBELL_STAR_1=(1./(1+abs((x-
Chromosome(jj,1))/Chromosome(jj,2)).^2*Chromosome(jj,3)))
```

```
Y=X1
for p=1:iii+1
h(:,p)=GBELL_STAR_1.*Y(:,p)
end

elseif model==5              % Gaussian

GAUSS_STAR_1=exp(-(x - Chromosome(jj,1)).^2/(2*Chromosome(jj,2)^2))
Y=X1
for p=1:iii+1
h(:,p)=GAUSS_STAR_1.*Y(:,p)
end



end
X=[X1 h]
X2=[X1 x]
bols=inv(X'*X)*X'*y


yy1=X*bols
e=yy1-y
error=sum(sum(e.^2))

object_value(jj,:)=1/2*mse(e)



 end
fitness_value = object_value;
total_fit=sum(fitness_value);
fitness_value=fitness_value/total_fit;
fitness_value=cumsum(fitness_value);
[nkk,nii]=size(pop);
ms=sort(rand(nkk,1));
fitin=1;
newin=1;
while newin<=nkk
if(ms(newin))<fitness_value(fitin)
newpop(newin,:)=pop(fitin,:);
newin=newin+1;
else
fitin=fitin+1;
end
end
% crossover between chromosoms
pop = newpop;

length_chrom = size(pop,2);
cp = ceil(rand(size(pop,1)/2,1)*(length_chrom-1));
cp = cp.*(rand(size(cp))<pc);
for i = 1:length(cp);
   newpop([2*i-1 2*i],:) = [pop([2*i-1 2*i],1:cp(i)) ...
pop([2*i 2*i-1],cp(i)+1:length_chrom)];
end

pop = newpop;
mutated_pop = find(rand(size(pop))<pm);
newpop = pop;
```

```
newpop(mutated_pop) = 1-pop(mutated_pop);

% finding the best individual
pop = newpop;

best_individual=pop(1,:);
best_fit=fitness_value(1);
for jj=2:nkk
if fitness_value(jj)<best_fit
best_individual=pop(jj,:);
best_fit=fitness_value(jj);
end
end

iterations
array_best ( iterations )= best_fit;
index_iter ( iterations ) = iterations;
end

best_gen=best_individual
if model==4
 c=best_gen(:,1)
 gamma=best_gen(:,2)
 be=best_gen(:,3)
else
    gamma=best_gen(:,1)
c=best_gen(:,2)

end

if model==1                % ESTAR (exponential)
ESTAR_2= 1-exp(-gamma.*((x-c).^2))
Y=X1
f=y(end,:)*ESTAR_2(end,:)

for p=1:iii+1
h(:,p)=ESTAR_2.*Y(:,p)
end


elseif model==2                % LSTAR (logistic)
LSTAR_2=1./(1+exp(-gamma.*(x-c)))
Y=X1
f=y(end,:)*LSTAR_2(end,:)

for p=1:iii+1
h(:,p)=LSTAR_2.*Y(:,p)
end


elseif model==3              % TAHN_STAR (tangent hyperbolic)

TSTAR_2=(2./(1+exp(-2*gamma.*(x-c)))-1)
Y=X1
f=y(end,:)*TSTAR_2(end,:)

for p=1:iii+1
h(:,p)=TSTAR_2.*Y(:,p)
```

```
end

elseif model==4              % Generalized Bell function

GBELL_STAR_2=(1./(1+abs((x-c)/gamma).^2*be))
Y=X1
f=y(end,:)*GBELL_STAR_2(end,:)

for p=1:iii+1
h(:,p)=GBELL_STAR_2.*Y(:,p)
end

elseif model==5              % Gaussian

GAUSS_STAR_2=exp(-(x - c).^2/(2*gamma^2))
Y=X1
f=y(end,:)*GAUSS_STAR_2(end,:)

for p=1:iii+1
h(:,p)=GAUSS_STAR_2.*Y(:,p)
end

end

X_new=[Y h]
[nk ni]=size(X_new)
bols_new=inv(X_new'*X_new)*X_new'*y
res=y-X_new*bols_new
s2 = (y-X_new*bols_new)'*(y-X_new*bols_new)/(nk-ni);
Vb=s2*inv(X_new'*X_new);              % Get the variance-covariance
matrix
se=sqrt(diag(Vb));              % Get coefficient standard errors
tstudent=bols_new./se;
yf=X_new*bols
[H,pValue,ARCHstat,CriticalValue] = archtest(res,2,0.05)

% Heteroskedasticity test
e2=res.*res;
x2=x.*x;
v=e2-x2*(e2'/x2')';
e2=e2-mean(e2)';
te=length(res(:,1))*(1-(v'*v)/(e2'*e2));
ht=1-chi2cdf(te,length(x(1,:)));

%ljung_box statistic for autocorrelation

[H,p_Jung,Qstat,CriticalValue] =lbqtest(res,2,0.05)

for kkk=1:nk
    if yf(kkk,:)>0
        S_in_sample(kkk,:)=1

    elseif yf(kkk,:)<0
        S_in_sample(kkk,:)=0
    end
end
Actual_positive_in_sample=find(y==1)
Actual_negative_in_sample=find(y==0)
```

```
Predicted_positive_in_sample=find(S_in_sample==1)
Predicted_negative_in_sample=find(S_in_sample==0)

Sum_actual_positive_in_sample=length(Actual_positive_in_sample)
Sum_actual_negative_in_sample=length(Actual_negative_in_sample)

Sum_predicted_positive_in_sample=length(Predicted_positive_in_sample)
Sum_predicted_negative_in_sample=length(Predicted_negative_in_sample)

Total_predicted_in_sample=find(y==S_in_sample)

Perc=(length(Total_predicted_in_sample)/nk)*100

for ii=nforecast:-1:1
y=data(1:end-ii,1)

x=lagmatrix(y,d)

clear ylag
for p=1:iii
ylag(:,p)=lagmatrix(y,p)
end
te=length(ylag)
t=length(y)

ylag=ylag(iii+d+1:t,:)
y=y(iii+d+1:t,:)
x=x(iii+d+1:t,:)

X1=[ones(size(y)) ylag]


if model==1                % ESTAR (exponential)
ESTAR_1= 1-exp(-gamma.*((x-c).^2))
Y=X1
f=y(end,:)*ESTAR_1(end,:)
clear h
for p=1:iii+1
h(:,p)=ESTAR_1.*Y(:,p)
end


elseif model==2                % LSTAR (logistic)
LSTAR_1=1./(1+exp(-gamma.*(x-c)))
Y=X1
f=y(end,:)*LSTAR_1(end,:)
clear h
for p=1:iii+1
h(:,p)=LSTAR_1.*Y(:,p)
end


elseif model==3               % TAHN_STAR (tangent hyperbolic)

TSTAR_1=(2./(1+exp(-2*gamma.*(x-c)))-1)
Y=X1
f=y(end,:)*TSTAR_1(end,:)
```

```
clear h
for p=1:iii+1
h(:,p)=TSTAR_1.*Y(:,p)
end

elseif model==4              % Generalized Bell function

GBELL_STAR_1=(1./(1+abs((x-c)/gamma).^2*be))
Y=X1
f=y(end,:)*GBELL_STAR_1(end,:)
clear h
for p=1:iii+1
h(:,p)=GBELL_STAR_1.*Y(:,p)
end

elseif model==5              % Gaussian

GAUSS_STAR_1=exp(-(x - c).^2/(2*gamma^2))
Y=X1
f=y(end,:)*GAUSS_STAR_1(end,:)
clear h
for p=1:iii+1
h(:,p)=GAUSS_STAR_1.*Y(:,p)
end

end

X=[Y h]

yhat(ii,:)=[X(end,1) y(end,:)  X(end,3) f ]*bols

end

for iii=1:nforecast
yfore(iii,:)=yhat(end-iii+1,:)
iii=iii+1
end

test_y=data(end-nforecast+1:end,1)

X3=[X1 h]
predict=X3*bols



tt=length(test_y)
for kkk=1:tt
    if test_y(kkk,:)>0
        S_out_of_sample(kkk,:)=1

    elseif test_y(kkk,:)<0
        S_out_of_sample(kkk,:)=0
    end
end



for kkk=1:tt
    if yfore(kkk,:)>0
```

```
        S(kkk,:)=1

    elseif yfore(kkk,:)<0
        S(kkk,:)=0
    end
end
Actual_positive_out_of_sample=find(S_out_of_sample==1)
Actual_negative_out_of_sample=find(S_out_of_sample==0)

Predicted_positive_out_of_sample=find(S==1)
Predicted_negative_out_of_sample=find(S==0)
Total_predicted_out_of_sample=find(S_out_of_sample==S)
Perc_out_sample=(length(Total_predicted_out_of_sample)/tt)*100

figure, plot(y(1:end-nforecast,:),'-r'); hold on; plot(yf,'-b');
xlabel('Periods')
ylabel('Values')
h1 = legend('Actual','forecasts',1);
%title('Out_of_sample forecasts')
figure, plot(test_y,'-r'); hold on; plot(yfore,'-b');
figure, plot (index_iter , array_best );
```

The initialization for ESTAR, LSTAR and TSTAR is

```
lb= min(min(x)) % bounds of the parameters c to be optimized
ub = max(max(x))


if model==4
 pop=[pop_c pop_gamma pop_b]
else
pop=[pop_gamma pop_c]
end
Range = repmat((ub-lb),[popsize chromlength]);
Lower = repmat(lb, [popsize chromlength]);
pop= rand(popsize,chromlength) .* Range + Lower;
```