Additional Smoothing Transition Autoregressive Models

Giovanis, Eleftherios

10 August 2008

Additional Smoothing Transition Autoregressive Models

Eleftherios Giovanis

Abstract

In this paper we present, propose and examine additional membership functions. There is no reason why more functions cannot be proposed. More specifically, we present the tangent hyperbolic, Gaussian and Generalized bell functions. Because Smoothing Transition Autoregressive (STAR) models follow fuzzy logic approach, more functions should be tested. Furthermore, fuzzy rules can be incorporated or other training or computational methods can be applied as the error backpropagation algorithm instead to nonlinear squares. Some numerical applications are presented and MATLAB routines are provided.

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

1. Introduction

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. We are interesting in forecasting, because a good model is judged on its forecasting performance, so hence its estimations would be more reliable. Furthermore, the tests do not guarantee that the specific smoothing function is appropriate or not. In section 2 we present the methodology of STAR models and the membership functions used in this study, while in section 3 we present some numerical applications.

2. Methodology

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

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

,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.

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

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

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

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

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

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 $j=1,\ldots p$, 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:

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

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 $d=1,\ldots,6$ 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. We examine for $j=1,\ldots, 5$ and we choose those values of j and d , where the *P-value* is minimized. We apply a grid search procedure for equation (1) with non linear squares and Levenberg-Marquardt algorithm.

3. Numerical Applications

First we apply non linear squares with no grid search. In some cases allowing free space without restricting the values that the fuzzy parameters can take we can improve the forecasts. Applications of STAR models in macroeconomic variables and one-step ahead forecasting provides the same performance with linear models, as Autoregressive and ARIMA. For this reason we apply one-step ahead forecasts for stock returns which is much more difficult to predict the correct sign. In the first example we examine S&P 500, FTSE 100, CAC 40 and DAX index returns with no grid search and the MATLAB routine 1. We do not test for STAR specification test as we are interesting to test them all. We found that STAR models are autoregressive of order 1. We use the last 100 trading days of 2009. The first 80 are used for estimation and the last 20 for test period or out-of-sample forecasts. Firstly, it should be noticed that we apply one-step ahead predictions. Secondly, the sample might be small, but the forecasts are much more superior if we would have taken a bigger sample. This indicates that STAR models are not based absolutely in statistics, where we need a big sample, but more in fuzzy logic, where we do not need long sample in order to improve the forecasts. In table 1 we present the percentage of the correct sign. We found that transition function for S&P and DAX indices is 1, for CAC 40 is 5 and for FTSE 100 is 2.

Table 1. Correct percentage sign for S&P 500, FTSE 100, DAX and CAC 40

Indices	ESTAR	LSTAR	TSTAR	GBELL_STAR	GAUSS_STAR
S&P 500	65.00	55.00	60.00	65.00	60.00
FTSE 100	45.00	60.00	60.00	55.00	60.00
CAC 40	70.00	65.00	55.00	70.00	65.00
DAX	55.00	45.00	50.00	60.00	60.00

In table 1 we observe that generally the results are mixed, but in some cases the functions we propose outperform either ESTAR or LSTAR. Furthermore the correct sign is rather low in relation with studies supporting artificial intelligence.

In the next example we examine unemployment rate for USA in monthly data. We take period 1947-2009, where the period September- December 2009 is left for out-of-sample period, so we apply a four-step ahead forecast. We do not follow Monte-Carlo or any other simulations, but initially we forecast for one period. Then we re-estimate with nonlinear squares based on the new data, including the last forecasting values, and then we proceed for the new one. All the models do not work well with no grid search, except from Gbell function.

Gbell function is ideal, because we need no grid search or we do not have to divide by standard deviation or taking any other measure. In figures 1 and 2 we present the in-sample and out of-sample forecasts for US unemployment rate and with Gbell membership function. The same conclusion is derived by testing the six-monthly treasury bills of USA, as also the gross domestic products of USA and Canada, but we do not present the results.

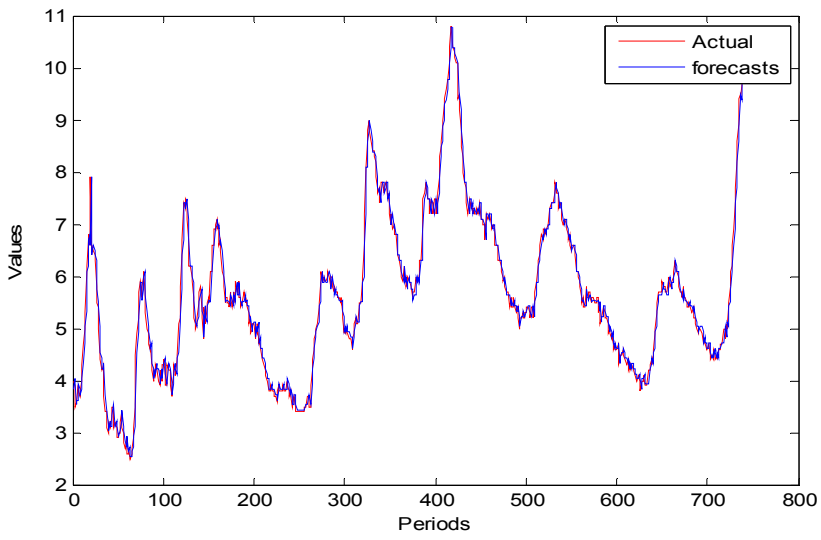

Fig. 1. In-sample forecasts for US unemployment rate and Gbell function.

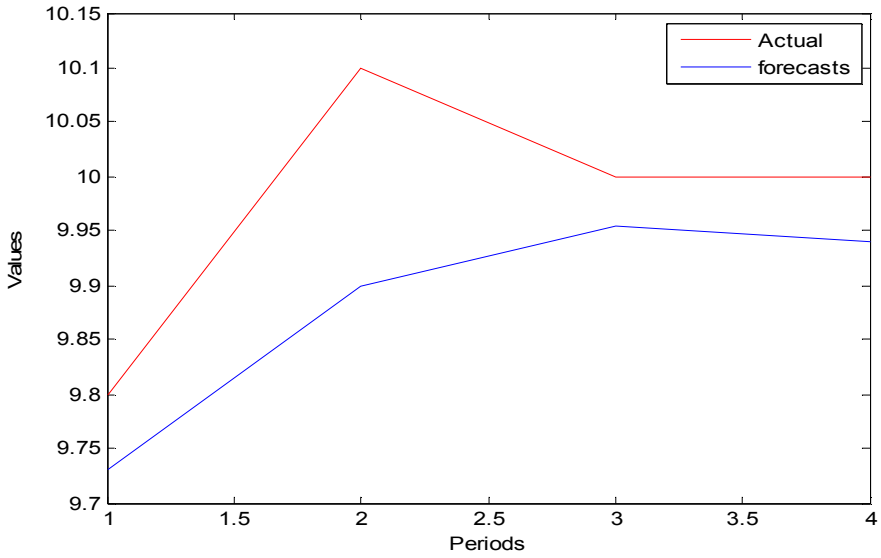

Fig. 2. Out-of-sample forecasts for US unemployment rate and Gbell function.

In the next example we examine again the same stock index returns, with the same settings and the same period, but we apply a grid search procedure. For parameter γ we search in the interval [1 3], with increment 0.5 and for parameter c we search in the interval [-0.03 0.03], with increment 0.01. Additionally for parameter b, which is needed for Generalized Bell (Gbell) function we apply a grid search in the interval [0.5 2], with increment 0.1.

We apply grid search procedure in short intervals, because it is just an example. The reason why we are doing this is because of the long computation time needed to compute the fuzzy parameters especially in Gbell function, where we have three parameters. re-estimate each time and make a one-step ahead forecast. Additionally, we set up the length for parameters c and γ equal for simplicity. These intervals can be expanded or can be changed based on the time-series we examine each time. In programming routine 2 we provide this procedure. We could assign to take these values of fuzzy parameters where the error is the minimum; but the error can be the same in more than one cases.

Table 2. Correct percentage sign for S&P 500, FTSE 100, DAX and CAC 40

Indices	ESTAR	LSTAR	TSTAR	GBELL_STAR	GAUSS_STAR
S&P 500	65.00	65.00	60.00	65.00	65.00
c	0.000	0.0241	0.0224	0.000	0.010
γ	1.000	1.6415	1.5326	1.000	1.000
b				1.6318	
FTSE 100	40.00	50.00	60.00	55.00	55.00
c	0.030	0.0226	0.006	0.000	0.010
γ	1.000	1.3253	1.500	1.000	1.500
b				1.6318	
CAC 40	50.00	35.00	55.00	60.00	50.00
c	0.010	-0.030	0.030	0.030	-0.010
γ	1.500	1.500	3.000	1.500	1.010
b				1.500	
DAX	50.00	60.00	60.00	55.00	55.00
c	0.010	0.0252	0.0236	-0.030	-0.010
γ	1.500	1.4307	1.3517	1.3433	1.000
b				2.000	

In table 2 we observe that all models present almost the same performance, with the exception of few cases. Additionally, the models we propose in some cases outperform ESTAR and LSTAR, like the TSTAR in FTSE 100, TSTAR and GBELL_STAR in CAC 40, while in the other cases present the same performance, as GBELL_STAR and GAUSS_STAR in S&P 500.This indicates that additional STAR functions should be examined. Furthermore, fuzzy rules should be obtained to capture for imprecision. Similarly a routine can be written for multi-step ahead period forecasts. Of course there are researchers who claim that economics and finance are based on statistics and probability. Firstly, this is wrong or not absolute as the researchers who claim that are not experts and do not know from artificial intelligence procedure and have not even tested or examined them. Secondly, STAR functions as we mentioned previously, contain fuzzy logic as the non linear part is actually the fuzzy membership values or grades, even if the researchers who make use of them they do not know that. Additionally, people have the weird and bad habit and need of belonging somewhere. So they are discriminate to parametric analysts, Bayesian analysts, Keynesians or Neo-Keynesians, liberalists of no- liberalists Marxists, non-parametric analysts and many others. There is not correct theory or philosophy in

scientific thought. Even in dialogue between Socrates and Protagoras (in Protagoras written by Plato in 380 B.C.E) the optimum or the best solution is the combination of Socrate's and Protagora's philosophic positions and not the choice of one of them.

Conclusions

Furthermore the STAR models are incomplete. To be specific the nonlinear part accounts for the membership grades of inputs but no rules are included. Additionally STAR models assume only one or even no linguistic term and it should be for example for exchange rates or stock returns a fuzzy set of linguistic terms like {negative returns, zero returns, positive returns}, or even assigning more or different linguistic terms like {very negative returns, negative returns, zero returns, positive returns, very positive returns} and therefore taking the appropriate operator, max, min or product. More over additional membership function can be examined as the triangular, trapezoidal, the S-shaped or Z-shaped among others. Finally, other optimization methods can be applied in order to find the fuzzy parameters as the error backpropagation or genetic algorithms instead to nonlinear squares and Levenberg-Marquardt algorithm.

References

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

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 1

Nonlinear squares with no grid search and one-step ahead forecasts

```
clear all;
% Load input data
load file.mat       % load the file with data
% Set up the forecasting period
nforecast=20

for ii=nforecast:-1:1
y=data(end-100:end-ii,:)
g=mean(y)

t=length(y)
% Tset up the delay
d=1;
x=lagmatrix(y,d)
% Set up the lag order
i=1;
% Some initial values for c, gamma and be
c=0
gamma=1
model=5
be=2
stdev=std(y)
clear ylag
for p=1:i
ylag(:,p)=lagmatrix(y,p)
end
te=length(ylag)


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

X1=[ones(size(y)) ylag]
if model==1                % ESTAR (exponential)
ESTAR_1= 1-exp(-gamma.*((x-c).^2))
Y=X1

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


elseif model==2                % LSTAR (logistic)
LSTAR_1=1./(1+exp(-gamma.*(x-c)))
Y=X1
clear h
for p=1:i+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
clear h
for p=1:i+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
clear h
for p=1:i+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
clear h
for p=1:i+1
h(:,p)=GAUSS_STAR_1.*Y(:,p)
end

end
X=[X1 h]
X2=[X1 x]

% Estimate if you want with OLS to take initial values for estimated
% coefficients
bols=inv(X'*X)*X'*y


if model==1

bols1=[bols(1);bols(2);bols(3);bols(4);1;g]

[sol,r,J,COVB] = nlinfit(X2,y,@ESTAR,bols1);
c1=sol(end,:)
gamma1=sol(end-1,:)
bols2=sol(1:end-2,:)

elseif model==2
bols1=[bols(1);bols(2);bols(3);bols(4);1;g]

[sol,r,J,COVB] = nlinfit(X2,y,@LSTAR,bols1);
c1=sol(end,:)
gamma1=sol(end-1,:)
bols2=sol(1:end-2,:)

elseif model==3
bols1=[bols(1);bols(2);bols(3);bols(4);1;g]

[sol,r,J,COVB] = nlinfit(X2,y,@TSTAR,bols1);
c1=sol(end,:)
gamma1=sol(end-1,:)
```

```
bols2=sol(1:end-2,:)

elseif model==4
bols1=[bols(1);bols(2);bols(3);bols(4);g;1;2]

[sol,r,J,COVB] = nlinfit(X2,y,@GBELL_STAR,bols1);
be1=sol(end,:)
gamma1=sol(end-1,:)
c1=sol(end-2,:)
bols2=sol(1:end-3,:)
elseif model==5
bols1=[bols(1);bols(2);bols(3);bols(4);g;1]

[sol,r,J,COVB] = nlinfit(X2,y,@GAUSS_STAR,bols1);
c1=sol(end,:)
gamma1=sol(end-1,:)
bols2=sol(1:end-2,:)

end

if model==1                % ESTAR (exponential)
ESTAR_1= 1-exp(-gamma1.*((x-c1).^2))
Y=X1
f=y(end,:)*ESTAR_1(end,:)
for p=1:i+1
h(:,p)=ESTAR_1.*Y(:,p)
end


elseif model==2               % LSTAR (logistic)
LSTAR_1=1./(1+exp(-gamma1.*(x-c1)))
Y=X1
f=y(end,:)*LSTAR_1(end,:)
for p=1:i+1
h(:,p)=LSTAR_1.*Y(:,p)
end


elseif model==3              % TAHN_STAR (tangent hyperbolic)

TSTAR_1=(2./(1+exp(-2*gamma1.*(x-c1)))-1)
Y=X1
f=y(end,:)*TSTAR_1(end,:)
for p=1:i+1
h(:,p)=TSTAR_1.*Y(:,p)
end

elseif model==4              % Generalized Bell function

GBELL_STAR_1=(1./(1+abs((x-c1)/gamma1).^2*be))
Y=X1
f=y(end,:)*GBELL_STAR_1(end,:)
for p=1:i+1
h(:,p)=GBELL_STAR_1.*Y(:,p)
end

elseif model==5              % Gaussian
```

```
GAUSS_STAR_1=exp(-(x - c1).^2/(2*gamma1^2))
Y=X1
f=y(end,:)*GAUSS_STAR_1(end,:)
for p=1:i+1
h(:,p)=GAUSS_STAR_1.*Y(:,p)
end

end
%yhat(ii,:)=[X(end,:) y(end,:) y(end,:)*  *bols2
yhat(ii,:)=[X(end,1) y(end,:) X(end,3) f]*bols2

end

for iii=1:nforecast
yfore(iii,:)=yhat(end-iii+1,:)
iii=iii+1
end
se=sqrt(diag(COVB));              % Get coefficient standard errors
tstudent=sol./se;

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


X3=[X1 h]
predict=X3*bols2



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,'-r'); hold on; plot(predict,'-b');
figure, plot(test_y,'-r'); hold on; plot(yfore,'-b')
```

```
function y = ESTAR(beta,x1,y)


y=beta(1)*x1(:,1)+ beta(2)*x1(:,2)+
(beta(3)*x1(:,1)+beta(4)*x1(:,2)).*...
    (1-exp(-beta(5).*(x1(:,end)-beta(6)).^2))



function y = LSTAR(beta,x1,y)

y=beta(1)*x1(:,1)+ beta(2)*x1(:,2)+
(beta(3)*x1(:,1)+beta(4)*x1(:,2)).*...
    (1./(1+exp(-beta(5).*(x1(:,end)-beta(6)))))



function y = TSTAR(beta,x1,y)

y=beta(1)*x1(:,1)+ beta(2)*x1(:,2)+
(beta(3)*x1(:,1)+beta(4)*x1(:,2)).*...
    (2./(1+exp(-2*beta(5).*(x1(:,end)-beta(6))))-1)



function y = GBELL_STAR(beta,x1,y);


y=beta(1)*x1(:,1)+ beta(2)*x1(:,2)+
(beta(3)*x1(:,1)+beta(4)*x1(:,2)).*...
    (1./(1+abs((x1(:,end)-beta(5))/beta(6)).^2*beta(7)))



function y = GAUSS_STAR(beta,x1,y)

y=beta(1)*x1(:,1)+ beta(2)*x1(:,2)+
(beta(3)*x1(:,1)+beta(4)*x1(:,2)).*...
    exp(-(x1(:,end)-beta(5)).^2/(2*beta(6)^2))
```

MATLAB routine 2

Nonlinear squares with no grid search and multi-step ahead forecasts

```
nforecast=4
for jjj=1:nforecast
clear all;
% Load input data
load train_file.mat       % load the file with data

y=train_data
g=mean(y)

t=length(y)
% Tset up the delay
d=1;
x=lagmatrix(y,d)
% Set up the lag order
i=1;
% Some initial values for c, gamma and be
c=0
gamma=1
model=4
be=2
stdev=std(y)

for p=1:i
ylag(:,p)=lagmatrix(y,p)
end
te=length(ylag)


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

X1=[ones(size(y)) ylag]
if model==1               % ESTAR (exponential)
ESTAR_1= 1-exp(-gamma.*((x-c).^2))
Y=X1


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


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

for p=1:i+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

for p=1:i+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

for p=1:i+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

for p=1:i+1
h(:,p)=GAUSS_STAR_1.*Y(:,p)
end

end
X=[X1 h]
X2=[X1 x]

% Estimate if you want with OLS
bols=inv(X'*X)*X'*y


if model==1

bols1=[bols(1);bols(2);bols(3);bols(4);1;g]

[sol,r,J,COVB] = nlinfit(X2,y,@ESTAR,bols1);
c1=sol(end,:)
gamma1=sol(end-1,:)
bols2=sol(1:end-2,:)

elseif model==2
bols1=[bols(1);bols(2);bols(3);bols(4);1;g]

[sol,r,J,COVB] = nlinfit(X2,y,@LSTAR,bols1);
c1=sol(end,:)
gamma1=sol(end-1,:)
bols2=sol(1:end-2,:)

elseif model==3
bols1=[bols(1);bols(2);bols(3);bols(4);1;g]

[sol,r,J,COVB] = nlinfit(X2,y,@TSTAR,bols1);
```

```
c1=sol(end,:)
gamma1=sol(end-1,:)
bols2=sol(1:end-2,:)

elseif model==4
bols1=[bols(1);bols(2);bols(3);bols(4);g;1;2]

[sol,r,J,COVB] = nlinfit(X2,y,@GBELL_STAR,bols1);
be1=sol(end,:)
gamma1=sol(end-1,:)
c1=sol(end-2,:)
bols2=sol(1:end-3,:)
elseif model==5
bols1=[bols(1);bols(2);bols(3);bols(4);1;g]

[sol,r,J,COVB] = nlinfit(X2,y,@GAUSS_STAR,bols1);
c1=sol(end,:)
gamma1=sol(end-1,:)
bols2=sol(1:end-2,:)

end

if model==1                % ESTAR (exponential)
ESTAR_1= 1-exp(-gamma1.*((x-c1).^2))
Y=X1
f=y(end,:)*ESTAR_1(end,:)
for p=1:i+1
h(:,p)=ESTAR_1.*Y(:,p)
end


elseif model==2                % LSTAR (logistic)
LSTAR_1=1./(1+exp(-gamma1.*(x-c1)))
Y=X1
f=y(end,:)*LSTAR_1(end,:)
for p=1:i+1
h(:,p)=LSTAR_1.*Y(:,p)
end


elseif model==3              % TAHN_STAR (tangent hyperbolic)

TSTAR_1=(2./(1+exp(-2*gamma1.*(x-c1)))-1)
Y=X1
f=y(end,:)*TSTAR_1(end,:)
for p=1:i+1
h(:,p)=TSTAR_1.*Y(:,p)
end

elseif model==4              % Generalized Bell function

GBELL_STAR_1=(1./(1+abs((x-c1)/gamma1).^2*be))
Y=X1
f=y(end,:)*GBELL_STAR_1(end,:)
for p=1:i+1
h(:,p)=GBELL_STAR_1.*Y(:,p)
end

elseif model==5              % Gaussian
```

```
GAUSS_STAR_1=exp(-(x - c1).^2/(2*gamma1^2))
Y=X1
f=y(end,:)*GAUSS_STAR_1(end,:)
for p=1:i+1
h(:,p)=GAUSS_STAR_1.*Y(:,p)
end

end
X=[Y h]
yhat=[X(end,1) y(end,:) X(end,3) f ]*bols2
data1=[data1;yhat]
save('can', 'data1')
end
X3=[X1 h]
predict=X3*bols2
se=sqrt(diag(COVB));              % Get coefficient standard errors
tstudent=sol./se;
nforecast=4
load test_file.mat
test_y=test_data(end-nforecast+1:end,:)

yfore=train_data(end-nforecast+1:end,:)


figure, plot(y,'-r'); hold on; plot(predict,'-b');
xlabel('Periods')
ylabel('Values')
%title('In_sample forecasts')
h1 = legend('Actual','forecasts',1);
figure, plot(test_y,'-r'); hold on; plot(yfore,'-b');
xlabel('Periods')
ylabel('Values')
%title('Out_of_sample forecasts')
h = legend('Actual','forecasts',1);
```

MATLAB routine 3

Nonlinear squares with grid search and one-step ahead forecasts

```
clear all;
% Load input data
load file.mat       % load the file with data
y=data
g=mean(y)

t=length(y)
% Tset up the delay
d=1;
x=lagmatrix(y,d)
% Set up the lag order
i=1;
% Some initial values for c, gamma and be
c=0
gamma=1
model=4
be=2
stdev=std(y)

for p=1:i
ylag(:,p)=lagmatrix(y,p)
end

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

X1=[ones(size(y)) ylag]
if model==1                % ESTAR (exponential)
ESTAR_1= 1-exp(-gamma.*((x-c).^2))
Y=X1

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


elseif model==2                % LSTAR (logistic)
LSTAR_1=1./(1+exp(-gamma.*(x-c)))
Y=X1
for p=1:i+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
for p=1:i+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
for p=1:i+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
for p=1:i+1
h(:,p)=GAUSS_STAR_1.*Y(:,p)
end

end
X=[X1 h]
X2=[X1 x]
% Estimate if you want with OLS to take initial values for estimated
% coefficients
bols=inv(X'*X)*X'*y


grid_gamma = 1:0.5:3;
grid_c=-0.03:0.01:0.03
grid_b=0.5:0.1:1
gamma_f = zeros(size(grid_gamma));
c_f = zeros(size(grid_c));
if model==1
for i = 1:length(grid_gamma)
    for j = 1:length(grid_c)
bols1=[bols(1);bols(2);bols(3);bols(4);grid_gamma(i);grid_c(j)]

[sol,res,J,COVB] = nlinfit(X2,y,@ESTAR,bols1);

error(i,j)=(y-X*sol(1:end-2,:))'*(y-X*sol(1:end-2,:))

    end
end



elseif model==2
for i = 1:length(grid_gamma)
    for j = 1:length(grid_c)
bols1=[bols(1);bols(2);bols(3);bols(4);grid_gamma(i);grid_c(j)]

[sol,res,J,COVB] = nlinfit(X2,y,@LSTAR,bols1);

error(i,j)=(y-X*sol(1:end-2,:))'*(y-X*sol(1:end-2,:))

    end
end

elseif model==3
for i = 1:length(grid_gamma)
```

```
    for j = 1:length(grid_c)
bols1=[bols(1);bols(2);bols(3);bols(4);grid_gamma(i);grid_c(j)]

[sol,res,J,COVB] = nlinfit(X2,y,@TSTAR,bols1);

error(i,j)=(y-X*sol(1:end-2,:))'*(y-X*sol(1:end-2,:))

    end
end

elseif model==4
    for i = 1:length(grid_gamma)
    for j = 1:length(grid_c)
        for r = 1:length(grid_b)
bols1=[bols(1);bols(2);bols(3);bols(4);grid_c(j);grid_gamma(i);grid_b
(r)]

[sol,res,J,COVB] = nlinfit(X2,y,@GBELL_STAR,bols1);

error.field(i,j,r)=(y-X*sol(1:end-3,:))'*(y-X*sol(1:end-3,:))
        end
    end
    end

elseif model==5

for i = 1:length(grid_gamma)
    for j = 1:length(grid_c)
bols1=[bols(1);bols(2);bols(3);bols(4);grid_c(j);grid_gamma(i)]

[sol,res,J,COVB] = nlinfit(X2,y,@GAUSS_STAR,bols1);

error(i,j)=(y-X*sol(1:end-2,:))'*(y-X*sol(1:end-2,:))

    end
end
end



nforecast=20
load file.mat
bols2=sol(1:end-2,:)
for ii=nforecast:-1:1
y=data(end-100:end-ii,1)

d=1;
x=lagmatrix(y,d)
i=1;
c=0.01
gamma=1.5
model=1
be=2

clear ylag
for p=1:i
ylag(:,p)=lagmatrix(y,p)
end
```

```
te=length(ylag)
t=length(y)

ylag=ylag(i+d+1:t,:)
y=y(i+d+1:t,:)
x=x(i+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:i+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:i+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:i+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:i+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:i+1
h(:,p)=GAUSS_STAR_1.*Y(:,p)
end
```

```
end

X=[Y h]

yhat(ii,:)=[X(end,1) y(end,:)  X(end,3) f ]*bols2

end

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