Natural Resources Management

Streamflow forecast model for Ebro river and dam impact analysis

Students:

Ghodrat Rezaei 952128

Luca Acquati 920545

Mostafa Baghdadi 952057

Thomas Manzoni 920635

1.Streamflow forecast model for Ebro river

In this section, different data-driven models for streamflow forecast are computed in order to find the one that gives best performances. Dataset consists of 27 (1990-2016) years timeseries of streamflow, precipitation and temperature in the city of Tudela (Spain), boarded by Ebro river.
Since Ebro streamflow is the target variable to forecast, precipitations and temperature are treated as exogenous signals.

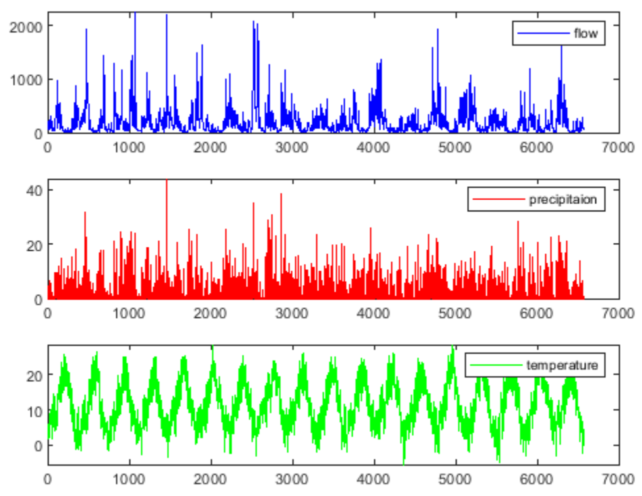
1.1 Loading data in the workspace

27 years dataset have been divided according to the following choice: 18 years (1990 to 2007) are used for training and remaining 9 years (2008 to 2016) are used for validation. Txt data file has 6 columns: Year, month, day, precipitation, streamflow and temperature respectively.

```
load -ascii data.txt
year_training=18;

prec_training=data(1:year_training*365,4);
flow_training=data(1:year_training*365,5);
temp_training=data(1:year_training*365,6);
prec_val=data(year_training*365+1:end,4);
flow_val=data(year_training*365+1:end,5);
temp_val=data(year_training*365+1:end,6);
```

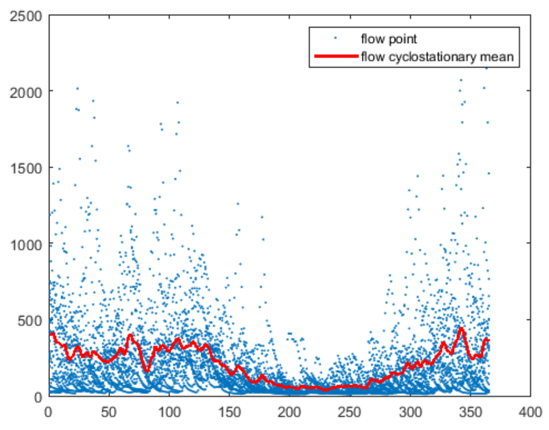
1.2 Data analysis

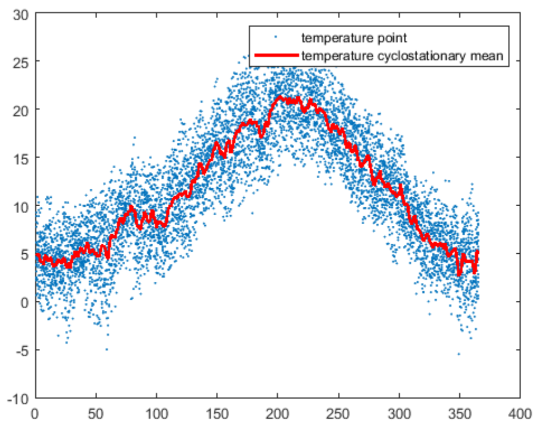
```
% data analysis based on training ones
figure;
subplot(3,1,1);
plot(flow_training,'b');
legend('flow');
subplot(3,1,2);
plot(prec_training,'r');
legend('precipitaion');
subplot(3,1,3);
plot(temp_training,'g');
legend('temperature');
```

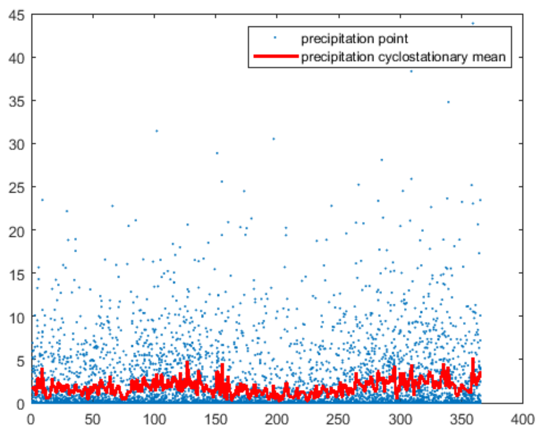


Data of the three variables of interested have been plotted and results show a cyclostationary behaviour, as expected. In order to perform model identification in a simpler way, this trend has to be deleted. To detrend data, cyclostationary mean has been computed. First, simple mean has been taken into account. Obtaining it is quite easy. Reshape function is been used to transform target vector into a matrix with 365 rows. In this way, every raw of the obtained matrix contains every year values of the interest variable. Mean function take the average on every raw and thus, on every day. Transposing it is mandatory because mean function performs the task on columns and not on rows.

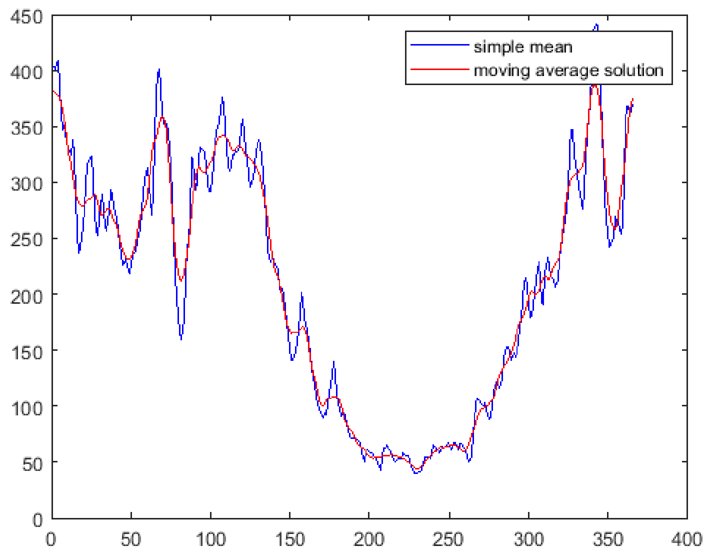
```
% tt is a repetead vector of time
tt = repmat([1:365]',year_training, 1);

figure;
plot(tt, flow_training,'.')
hold on;
CmF=mean((reshape(flow_training, 365, year_training))');
plot(CmF, 'r', 'LineWidth', 2)
legend('flow point', 'flow cyclostationary mean');

figure;
plot(tt, prec_training,'.')
hold on;
CmP=mean((reshape(prec_training, 365, year_training))');
plot(CmP, 'r', 'LineWidth', 2)
legend('precipitation point', 'precipitation cyclostationary mean');

figure;
plot(tt, temp_training,'.')
hold on;
CmT=mean((reshape(temp_training, 365, year_training))');
plot(mean((reshape(temp_training, 365, year_training))'), 'r', 'LineWidth', 2)
legend('temperature point', 'temperature cyclostationary mean');
```







Plots above shows the cyclostationary mean of the three signals (in red) with respect to all the measurements of 18 years of training. For the purpose of making the cyclostationary mean smoother, the given moving_average function is used. The higher the third parameter of the function, the longer the windows and so the smoother the output. Comparison between simple mean and moving average of streamflow has been plotted below as example. Computation of variances as moving average are also performed so that times series can be normalized without cyclostationary mean and cyclostationary standard deviations.



```
[f, F]=moving_average(flow_training, 365, 5);

figure;
plot(CmF, 'b');
hold on;
plot(f, 'r');
legend('simple mean', 'moving average solution');

[p, P]=moving_average(prec_training, 365, 5);
[t, Tree]=moving_average(temp_training, 365, 5);
[f_v, F_v]=moving_average(flow_val, 365, 5);
[p_v, P_v]=moving_average(prec_val, 365, 5);
[t_v, T_v]=moving_average(temp_val, 365, 5);

[sf, Sf]=moving_average((flow_training-F).^2,365,5);
sigmaF=sqrt(Sf);
[sp, Sp]=moving_average((prec_training-P).^2,365,5);
sigmaP=sqrt(Sp);
[st, St]=moving_average((temp_training-Tree).^2,365,5);
sigmaT=sqrt(St);
[sf_v, Sf_v]=moving_average((flow_val-F_v).^2,365,5);
sigmaF_v=sqrt(Sf_v);
[sp_v, Sp_v]=moving_average((prec_val-P_v).^2,365,5);
sigmaP_v=sqrt(Sp_v);
[st_v, St_v]=moving_average((temp_val-T_v).^2,365,5);
sigmaT_v=sqrt(St_v);
```

1.3 Detrendization

Once that cyclostationary means and variances have been computed, all signals (both of training and validation) can be normalized according to the expression

$$x = \frac{y - m}{\delta}$$

```
Fx = (flow_training-F)./sigmaF ;
Px = (prec_training-P)./sigmaP ;
Tx = (temp_training-Tree)./sigmaT ;
Fx_v = (flow_val-F_v)./sigmaF_v ;
Px_v = (prec_val-P_v)./sigmaP_v ;
Tx_v = (temp_val-T_v)./sigmaT_v ;
```

1.4 Correlation computations

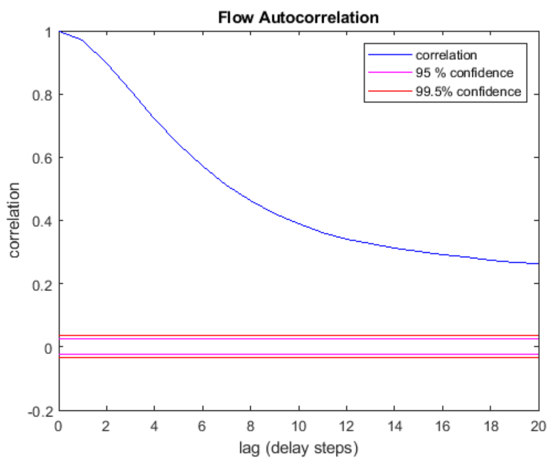
Thanks to the given correlogram function, the streamflow autocorrelation and the correlation between streamflow and the other two signals have been computed. These procedures help the designer understanding wheter the available signals are useful for model building.
Third parameter of the given function represent the maximum value of delay while computing correlations.

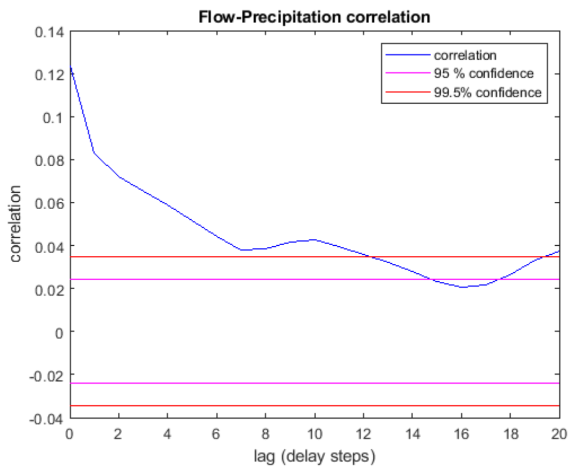
```
figure;
correlogram(flow_training,flow_training,20);
title('Flow Autocorrelation');
figure;
correlogram(flow_training,prec_training,20);
title('Flow-Precipitation correlation');
figure;
correlogram(flow_training,temp_training,20);
title('Flow-Temperature correlation');
```

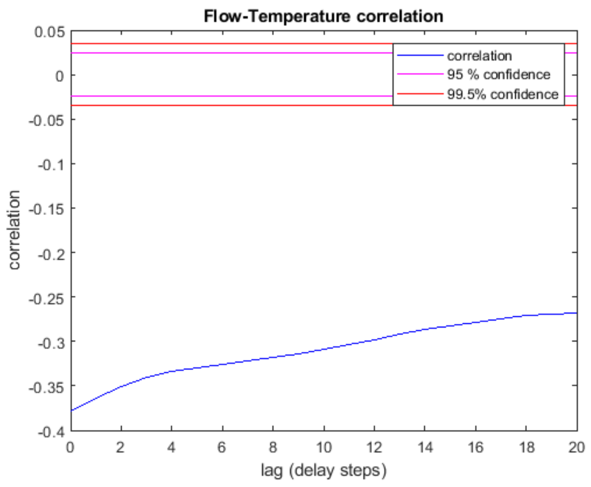
As plotted below it is possible to affirm that:

- River streamflow is highly autocorrelated (linear AR is a good option).
- Precipitations signal is a weak information for predicting river streamflow but still useful.
- Temperature has a good endless correlation with streamflow.

Correlation temperature streamflow is negative because increasing temperature streamflow decreases. Once can find justification in the Ebro river origins. Since this river hasn't glacial origins indeed, when temperature increases, there aren't enough melted glaciers to compensate the strong impact of evaporation. This phenomenon can also be read on average year streamflow, where it is clear that in the hottest months of the year, the streamflow has a minimum peak.







Correlations results show that both exogenous signals could be useful for streamflow forecast modelling.

1.5 ARX_solver function

First kind of models to have been analyzed were linear ones. Because of the presence of two exogenous signal modelling choice has fallen on ARX models. These models consist of an auto regressive part (past values of streamflow) and past values of exogenous signals. General ARX one day ahead forecast model for streamflow is then:

$$F(t+1) = \alpha_1 F(t) + \cdots + \alpha_n F(t-n+1) + \beta_1 P(t) + \cdots + \beta_{m_1} P(t-m_1+1) + \gamma_1 T(t) + \cdots + \gamma_{m_2} T\,(t-m_2+1)$$

where n, m1 and m2 are orders for autoregressive part, precipitation part and temperature part respectively, while F, P, and T are streamflow, precipitation and temperature signals. When m1 or m2 are zero, the correspondent exogenous part is deleted from model equation.
ARX parameters can be easily solved with least-square approach thanks to the linearity of the model. It is not easy to decide a priori the order of the model and so experiments have been set to find out the best one.

To evaluate the model performances, R square index has been used:

$$R^2 = 1 - \frac{\sum_i (y_i - \hat{y}_i)^2}{\sum_i (y_i - \overline{y})^2}$$

This index practically computes how far the estimated output is different from the simplest model ever, i.e. the basic average of all outputs.
For the purpose of computing many ARX linear model to find out the best one, a dedicated function has been created.
This function performs the computation of ARX model of desired order for river streamflow one day ahead forecast, given time series of the three signals.
Required inputs are:

- flow_val, F_v and sigmaF_v are real time series of the process, cyclostationary mean and variance respectively, used for rebuilding correct signal and for R squared computation.
- Fx, Px, Tx, Fx_v, Px_v, Tx_v are detrendized signals (both training and validation are needed).
- n, m1, m2: orders of the ARX; n represent the auto regressive part, while m1 and m2 represent order of the exogenous signals.

Outputs are:

- y: estimated time series of streamflow prediction.
- R2: value of R squared index of the model

Harder part of the function construction has been building the correct general M and Y matrices given any possible set of signal orders. Limit of the function is that m1 and m2 orders cannot be greater than n.

```
function [y, R2] = ARX_solver(flow_val,F_v,sigmaF_v,Fx,Fx_v,Px,Px_v,Tx,Tx_v,n,m1,m2)

% TRAINING

Y=Fx(1+n:end);
M=Fx(1:end-n);
if(n>1)
for i=2:n
    M=[Fx(i:end-(n-i)-1), M];
end
end
if(m1>0)
    for i=1:m1
        M = [M, Px(i:end-(m1-i)-(n+1-m1))];
    end
end
if(m2>0)
    for i=1:m2
        M=[M, Tx(i:end-(m2-i)-(n+1-m2))];
    end
end

%LEAST SQUARE SOLUTION

theta = M\Y;
```

```
%VALIDATION

M_v=Fx_v(1:end-n);
if(n>1)
for i=2:n
    M_v=[Fx_v(i:end-(n-i)-1), M_v];
end
end
if(m1>0)
    for i=1:m1
        M_v=[M_v, Px_v(i:end-(m1-i)-(n+1-m1))];
    end
end
if(m2>0)
    for i=1:m2
        M_v=[M_v, Tx_v(i:end-(m2-i)-(n+1-m2))];
    end
end

%OBTAINED TIME SERIES

x = M_v*theta;

%RECONSTRUCTION OF THE SIGNAL

x = [Fx_v(1:n); x];
y = x.*sigmaF_v+F_v;

%PERFORMANCE INDEX COMPUTATION (R SQUARE)

R2 = 1-(sum((flow_val(1+n:end)-y(1+n:end)).^2)/sum((flow_val(1+n:end)-F_v(1+n:end)).^2));
end
```

1.6 Best proper linear model computation

Once that ARX_solver has been defined, experiments have been set:
A big nestled for loop has been created to compute every possible (and feasible for the function) ARX model. Orders that give the best model are memorized in dedicated variable, while the big solutions matrix tracked all the possible values. Maximum order of the model has been set to 5 for avoiding overcomplicated models. At the end, first empty row of the solution matrix has been deleted.

```
maxOrder=5;
R2_best=0;
best_order= [0 0 0];
solutions= zeros(1,4);
for i=1:maxOrder
    for j=0:maxOrder
        for h=0:maxOrder
            if(i>=j-1 && i>=h-1)
             [y_temp,R2_temp] =
                          ARX_solver(flow_val,F_v,sigmaF_v,Fx,Fx_v,Px,Px_v,Tx,Tx_v,i,j,h);
             solutions=[solutions;
                         R2_temp, i, j, h,];
                if(R2_temp>R2_best) R2_best=R2_temp;
                    y_best=y_temp;
                    best_order=[i j h];
                end
            end
        end
    end
end
```

```
solutions=solutions(2:end,:);
```

ARX (4,5,5) has been the best model so far, with an extraordinary R square value of 0.9801.

1.7 Best improper linear model computation

So far, only proper models have been considered, i.e. models where, at time t+1, available signals are those of time t, t-1,t-2…Nowadays, since rain and temperature forecasts are very accurate, one can build ARX models based on the future values of rain and temperature. This means that at time t, precipitation and temperature signals of time t+1 are available. By simply shifting the exogenous signals by one position, (and adding the last one value two time, to have vector length consistency) improper models can be built by using same procedure as before:

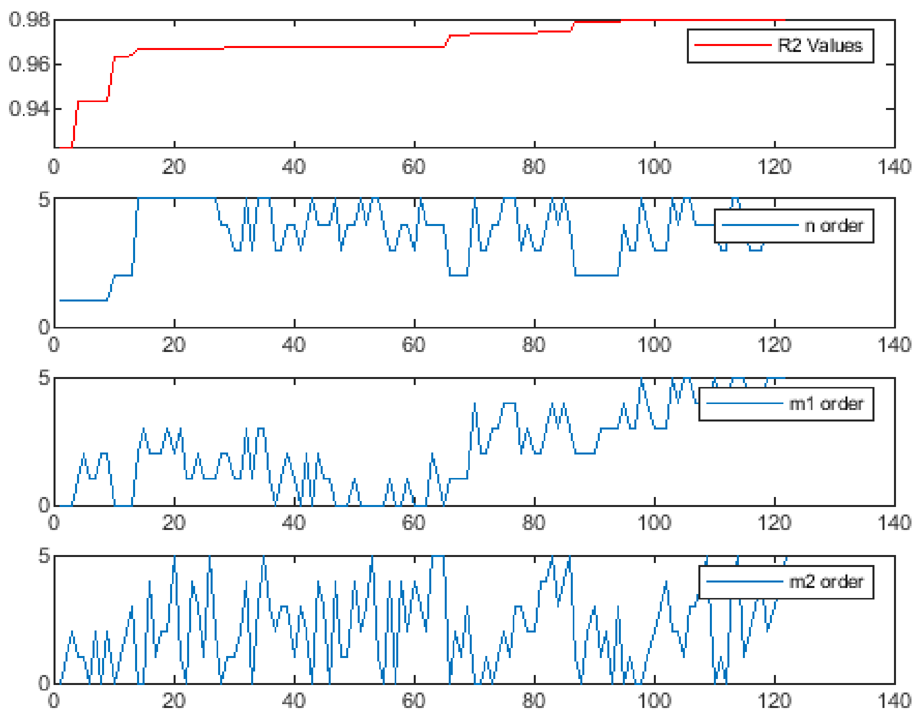
```
Pximp=[Px(2:end);Px(end)];
Pximp_v=[Px_v(2:end);Px_v(end)];
Tximp=[Tx(2:end);Tx(end)];
Tximp_v=[Tx_v(2:end);Tx_v(end)];
R2_best_imp=0;
best_order_imp= [0 0 0];
solutions_imp = zeros(1,4);

for i=1:maxOrder
    for j=0:maxOrder
        for h=0:maxOrder
            if(i>=j-1 && i>=h-1)
                 [y_temp, R2_temp] =
ARX_solver(flow_val,F_v,sigmaF_v,Fx,Fx_v,Pximp,Pximp_v,Tximp,Tximp_v,i,j,h);
                  solutions_imp=[solutions_imp;
                                 R2_temp, i, j, h,];
                if(R2_temp>R2_best_imp) R2_best_imp=R2_temp;
                    y_best_imp=y_temp;
                    best_order_imp=[i j h];
                end
            end
        end
    end
end
```

Results shows that best improper model is ARX_imp (3,3,3), with a R square value of 0.9802

1.9 Linear models result

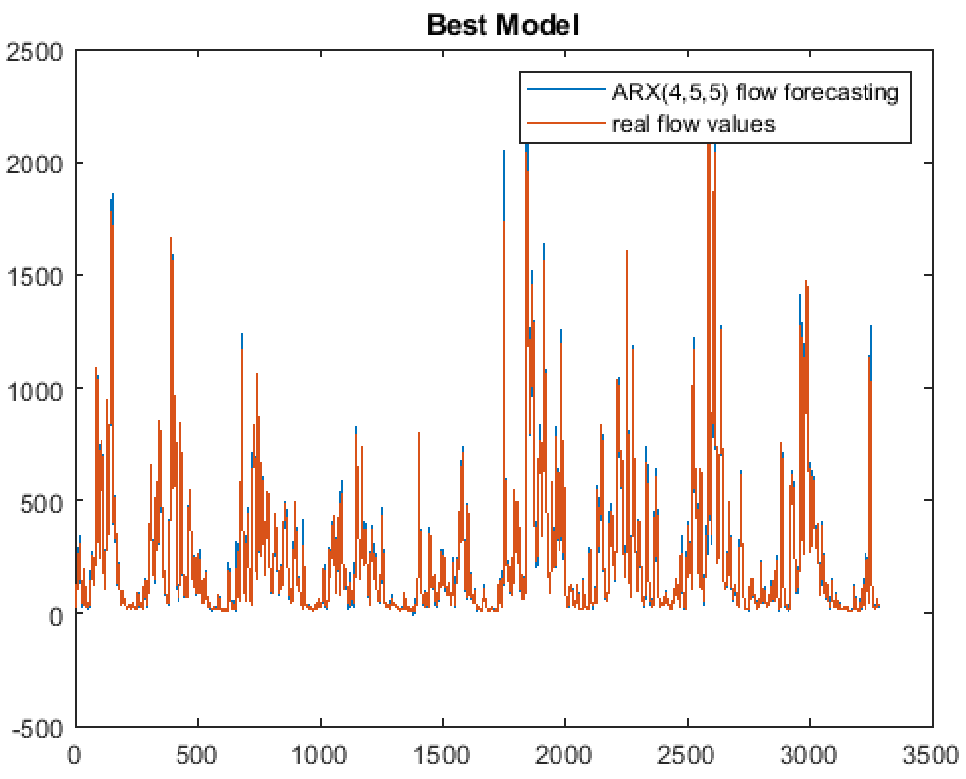
By looking at results, it is clear that the very slight improvement in using improper models don't justify the fact that weather forecasts cannot always have a 100% reliable prediction due to some unpredictable weather events. For this reasons, only proper models will be considered from now on.
Solutions matrix has been reordered in ascending values of R2 and, as shown in the plot below, performances increase in increasing n and m1, while m2 doesn’t affect so much the results. This fact is coherent with the analysis of correlations. Since temperature is highly long term correlated with streamflow, it doesn’t really matter the order of temperature, it is anyway good.

```
[~,idx] = sort(solutions(:,1));
solutions_reordered = solutions(idx,:);

figure;
subplot(4,1,1);
plot(solutions_reordered(:,1),'r');
legend('R2 Values');

subplot(4,1,2);
plot(solutions_reordered(:,2));
legend('n order');

subplot(4,1,3);
plot(solutions_reordered(:,3));
legend('m1 order');

subplot(4,1,4);
plot(solutions_reordered(:,4));
legend('m2 order');
```



Best model has been plotted compared to the real validation data.

```
figure;
string = sprintf('ARX(%d,%d,%d) flow
forecasting',best_order(1),best_order(2),best_order(3));
plot([y_best flow_val]);
title('Best Model');
legend(string,'real flow values');
```



1.10 Non linear modelling: Artificial Neural Networks

Also non-linear models have been considered. First, ANN_solver, i.e. an Artificial Neural Networks model solver has been built.
Input of the function are basically the same of ARX_solver, with only difference that n is now the number of layers in the ANN. Since Deep Learning toolbox works with rows and not column vectors, all matrices must be first transposed. The Function performs net training and then simulation on validation data. As ARX_solver, ANN_solver ouputs are simulated time series and R square value.

```
function [y, R2] = ANN_solver(flow_val,F_v,sigmaF_v,Fx,Fx_v,Px,Px_v,Tx,Tx_v,n)

X = [Fx(1:end-1), Px(1:end-1), Tx(1:end-1)]';
Y = Fx(2:end)';
X_v =[ Fx_v(1:end-1), Px_v(1:end-1), Tx_v(1:end-1)]';

net = newff(X,Y,n);
net = train(net,X,Y);
Y_v = sim(net, X_v );
Y_ann = [Fx_v(1); Y_v'];
y = Y_ann .*sigmaF_v + F_v;
R2 = 1 - sum( (flow_val( 2 : end ) - y( 2 : end )).^2 ) / sum( (flow_val( 2 : end )-
F_v(2:end) ).^2 );
end
```

Simulation based on the training net is quite a random process and so 10 proper models and 10 improper models have been computed to approach the theoretical best ANN model. Since there are 2 exogenous signal, optimal number of nodes is 4.

```
R2_ann_best=0;
for i = 1:10
    [y_temp, R2_temp] = ANN_solver(flow_val,F_v,sigmaF_v,Fx,Fx_v,Px,Px_v,Tx,Tx_v,4);
    if (R2_temp > R2_ann_best)
        R2_ann_best = R2_temp;
        y_ann_best= y_temp;
    end
end

for i = 1:10
    [y_temp, R2_temp] =
ANN_solver(flow_val,F_v,sigmaF_v,Fx,Fx_v,Pximp,Pximp_v,Tximp,Tximp_v,4);
    if (R2_temp > R2_ann_best)
        R2_ann_best = R2_temp;
        y_ann_best= y_temp;
    end
end
```

Best ANN model to be computed had R square value of 0.9448, quite lower than linear models value.

1.11 Non linear modelling: Classification And Regression Trees

Also CART models have been considered: first the general CART model is computed, then the results of optimal CART model are given. Optimal means with optimal minimum number of elements in every leaf.

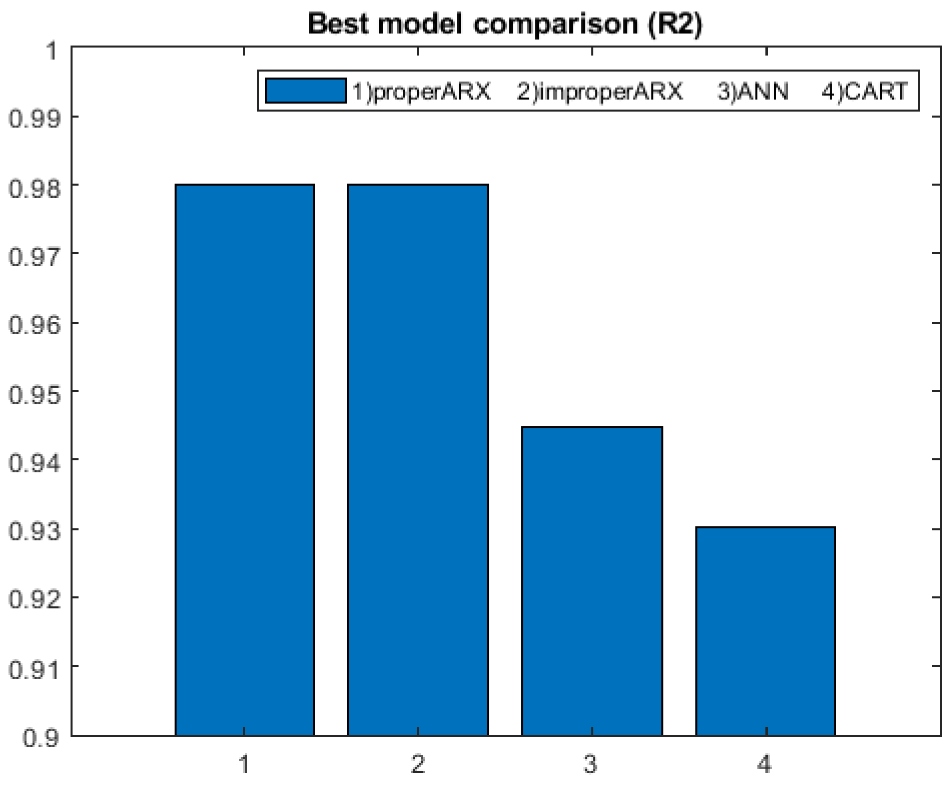
```
Ycart = Fx(2:end);
Xcart = [ Fx(1:end-1), Px(1:end-1), Tx(1:end-1)];
Xcart_v = [ Fx_v(1:end-1), Px_v(1:end-1), Tx_v(1:end-1)];

Tree = fitrtree(Xcart,Ycart);
Ycart_v = predict(Tree,Xcart_v);

Ycart_v = [Fx_v(1); Ycart_v];
y_cart = Ycart_v.*sigmaF_v + F_v;
R2_cart = 1 - sum( (flow_val( 2 : end ) - y_cart( 2 : end )).^2 ) / sum( (flow_val( 2 :
end )-F_v(2:end) ).^2 )


Tree_opt = fitrtree(Xcart, Ycart, 'OptimizeHyperparameter', 'auto' );
Ycart_v_opt=predict(Tree_opt,Xcart_v);
Ycart_v_opt = [Fx_v(1); Ycart_v_opt];
y_best_cart = Ycart_v_opt.*sigmaF_v + F_v;
R2_cart_best = 1 - sum( (flow_val( 2 : end ) - y_best_cart( 2 : end )).^2 ) / sum(
(flow_val( 2 : end )-F_v(2:end) ).^2 )
```

Also CART is not as good as a linear model. Optimal model gives indeed a R square of 0.9301

1.12 Final results

It is now clear that linear models perform the best. By putting an high number (40) in "maxOrder" variable, best absolute model has been ARX (18,19,19) with R square of 0.9804. In order to not overcomplicate the model "maxOrder" variable has been reduced (5) and best model ARX (4,5,5) gives an extraordinary near R square of 0.9801. Finally models are compared.

```
figure;
bar([R2_best, R2_best_imp, R2_ann_best, R2_cart_best]);
title('Best model comparison (R2)');
axis([0 5 0.9 1]);
legend('1)properARX    2)improperARX     3)ANN     4)CART');
```


Best model comparison (R2)
1)properARX 2)improperARX 3)ANN 4)CART