NATURAL RESOURCES MANAGEMENT PROJECT

🗺️ Doubs river, Barrage du Chatelot, Lac de Moron (France)

🖼️ 981.405000 Km^2

ABSTRACT

In this case study it is required to build a black-box one-day-ahead forecast for the inflow to the Lac de Moron, by exploring "linear models", "artificial neural networks" and "classification and regression trees". Moreover, the possibility of building a dam has to be discussed, alternatives have to be presented, tested and compared with the virtual no-dam alternative and the alternative currently in use at the Barrage du Chatelot by taking into account many stakeholders interests. A static management policy is sought through theoretically-driven considerations as well as a multi-objective, density-aware evolutionary algorithm.

Author:

Matteo Leccardi (mat. 945122)

Table of Contents

Problem statement

The objectives of this project are the following:

- Part 1: Build a one day ahead forecast model of the river streamflow.
- Part 2: Discuss the impacts of damming up the river.

The available data are historical time series of average daily streamflow [m3/s], average daily precipitations [mm] and daily average temperature [°C] starting from Jan. 1, 1990 up to Dec. 31, 2009.
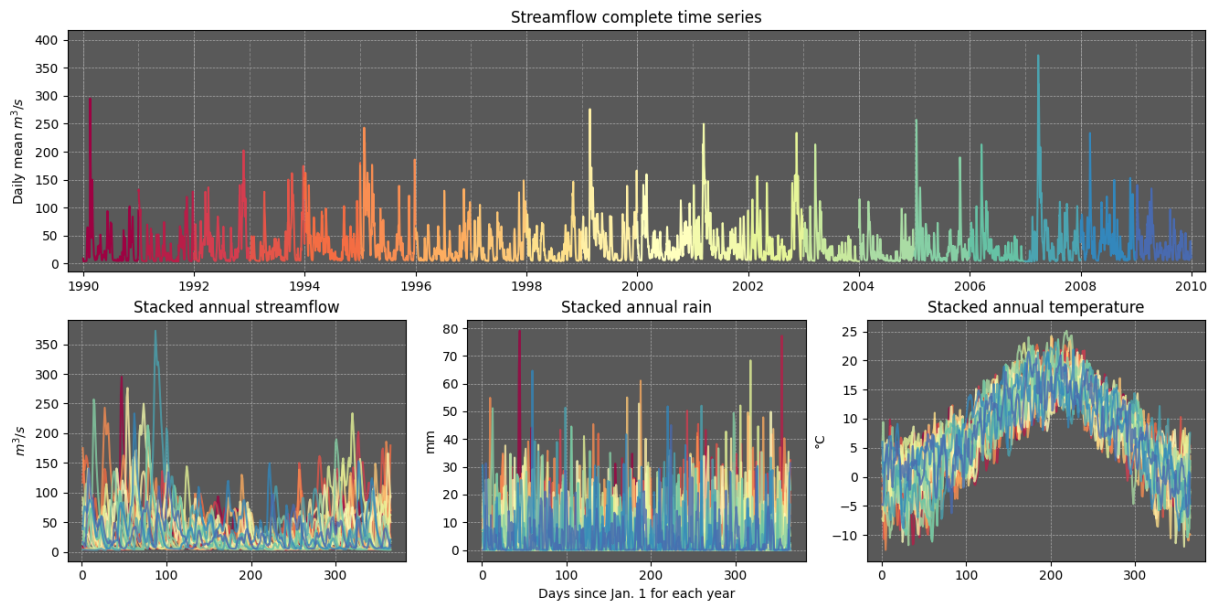
Some useful quantitative information about the time series quantities (expressed as daily averages) are reported in the following table:

Statistics over 20 years	Streamflow [m3/s]	Rainfall [mm/day]	Temperature [°C]
Minimum	3.05	0.0	-12.5942
Maximum	372.81	79.07	25.135
Range	369.75	79.07	37.73
Mean	32.91	4.37	7.59
(Variance, Std. dev.)	(1308.17 , 36.16)	(59.86 , 7.73)	(50.93 , 7.13)
Average #days/year with high observations ($> mean + 2stdev$)	20 days/year ($> 96 m^3/s$ on average)	21 days/year ($> 19mm$ on average)	/

A useful statistic to better understand the type of environment in which we are working are the average number of days per year with high observations for streamflow and rainfall, reported in the table.

Part 1: finding the best data-driven one-day-ahead forecast model

The best data-driven model is sought by considering three main families of models: models linear in the parameter's space such as autoregressive (pure, with exogenous inputs, proper and improper), artificial neural networks (both shallow and deep), and binary trees.

Before discussing the results obtained by the model identification procedures, some ground considerations have to be considered first.

Premises

The same cost function should be kept to evaluate the accuracy of the model throughout the whole decision-making process. Results obtained from testing different models would thus be directly comparable. We selected as the main cost function the mean square error (MSE) between the set of forecasted streamflow and the set of observed streamflow in the validation set.

This cost function is the preferred choice because model identification and calibration based on the minimization of the MSE cost function implicitly favors parametrizations and model structures that perform better in high values of the output (where it is easier to have higher values for the prediction error) rather than low values. This means that this cost function is well suited for problems related to wet ecosystems such as the one we are considering, at the border between Swiss and France with temperatures seldomly exceeding 20°C, mean daily precipitations of 4mm with peaks exceeding 70mm, and rivers with high peak flows during the rainy season between October and the end of April (at which it can peak to more than 330 m^3/s – around ten times more than the average streamflow over the recorded twenty years). The choice of the cost function is driven also by the following motivations: being this river the inflow of a (possible future) dam system, large prediction errors associated with larger streamflow values must be minimized as much as possible; it is much more useful to predict accurately high, sudden values of streamflow rather than average flows with slower dynamics. In that regard, a mean fourth-grade error would be suitable, however it doesn't share the same computational properties of the MSE.

In fact, the MSE cost function also has handy computational properties which allow to obtain a unique solution for the optimization problem during model calibration, however formulated (well suited for linear models as well as for artificial neural networks). Other indicators are considered in choosing the best model: together with the MSE cost function, other two indicators are useful, which are the MSE just for above-average observations (values greater than the average + 2 * standard deviation for that period) and for normal/below-average flows.

The validation sets and validation methodology was kept as consistent as possible. The developed algorithm cyclically calibrates the model under inspection on 19 out of 20 available years, and then use the remaining year as the validation dataset. Twenty separate metrics are obtained: one for each year. This cross-validation cycle seems reasonable for the following reasons: it allows to perform thorough cross-validation, the final cost function will be considered as the median of the twenty resulting cost functions, and this procedure allows to validate the proposed model on a whole year so that every season is considered. The median is chosen because if the 20 costs are symmetrically-enough distributed it would be quite the same as taking the mean, while if some outlier is present among the 20 years, the median would be more robust with respect to any skewness in the distribution.

The general procedure is defined as follows: the calibration step consists in the cyclical calibration of one model, the validation step is the subsequent evaluation of its accuracy using the validation set, through the mean square error between the real-world flow observations at time *t+1* and the one-day-ahead

forecast performed at time *t*. The final metric of evaluation is the median of the thus obtained 20 square roots of the MSE values, called J_{FINAL} (RMSE). J_{FINAL} is considered for three kinds of evaluations: all situations (J_{FINAL}), high-flow situations (J_{FINAL}^{H}) and medium-low-flow situations (J_{FINAL}^{L}). The best model is selected by considering all three situations, where J_{FINAL}^{H} has doube the weight in the final decision with respect to the other two situations. Also, models with lower order are preferred.

The R^2 performance index was used in the process, however just as a reverence and not as the main performance index, because although it offers an absolute reference frame, it has no physical meaning, and we deemed more useful working with results expressed in real physical quantities (such as m^3/s for *RMSE*).

The autocorrelation of the prediction error is used as a metric to verify the whiteness of the time series of the prediction errors and the minimum-complexity parametrization needed to obtain a white noise.

Linear parametric models

In model identification it is always better to start testing the simplest model possible to then, only if needed, increase the complexity. Here we start exploring linear models to perform the one day ahead forecast.

Following the Box and Jenkins method, the first step in the data-driven linear model identification is to remove any trend from the time series.

The histograms of streamflow and rainfall shows that their distribution is not normal and highly skewed to the left, with a hard limit at zero. Lognormal and Pearson III should be the best distributions to fit streamflow data distributions (Philip Kibet Langat, 2019) while for rain data the exponential distribution seems a good fit (except for very low values of rain). Knowing that these distributions should represent a good fit and finding the best parametrizations algorithmically though the maximum likelihood method, we can use the distributions to normalize and subsequently standardize the data. However, there are two problems: the fit are never very accurate for low values of flows and rain; but the most relevant issue is that, when using the preprocessed data to calibrate and then evaluate the model and we use that model to make predictions, these predictions have to be transformed back into the original distribution. This transformation is responsible for very high residual errors between the observed and predicted data. Any other combination of detrending, normalization and standardization presented the same issue. Let's take for example a normalization procedure consisting in transforming the real-world data into normalized data through a natural logarithm (and other fine-tuning parameters). The model calibration happens on normalized data, so the forecasting model works on normalized data, and the forecast has to be transformed back to the real-world space through the inverse transformation, an exponential. This means that a small error on the prediction for the normalized data transforms to an exponentially high error in the real-world data.

Instead, direct detrending and standardization proved more effective in the end. Since the normalized streamflow still expresses an evident cyclo-stationary behavior both in the mean trajectories and in the infra-seasonal variability, these cyclo-stationary trends are extracted. To do so, the mean and variance over the twenty years for each day (Jan. 1, 2, …, Dec 31) is computed, and then a moving average is performed over the year with particular attention not to obtain unwanted boundary effects. The window of the moving average over the daily means is chosen empirically to be of 13 days, as it yields the best end results.
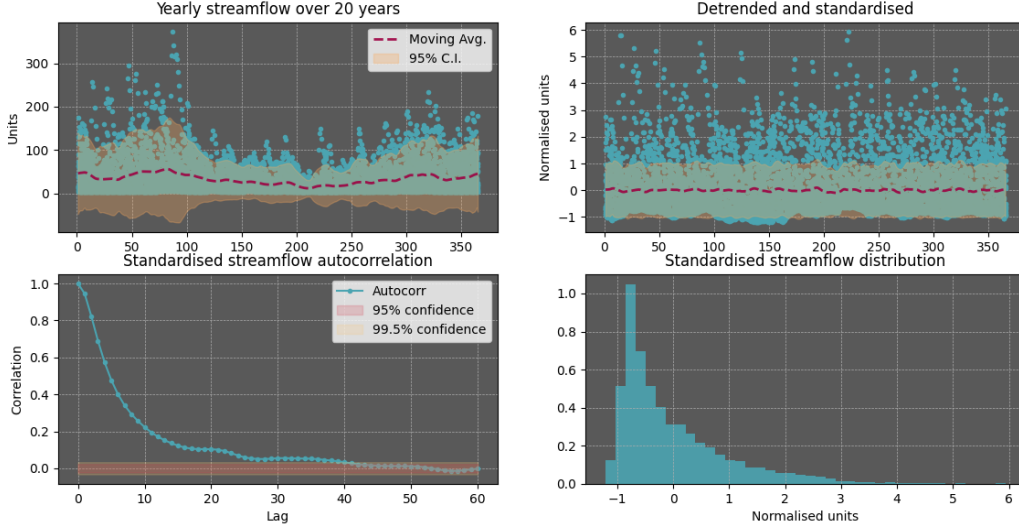
The autocorrelation clearly shows that the resulting signal (streamflow, but also rainfall and temperatures) is still not a white noise, so a model of greater complexity is sought.

The training and validation procedure as well as the cost function are chosen as discussed in the previous section (*Premises*). By defining N as the order of the autoregressive part, P as the order of the proper exogenous part given by rainfall, T as the order of the proper exogenous part given by temperature, and $\vec{p}$ the vector of parameters, the general prediction structure is the following:

$$x_t = \frac{F_t - \mu_k}{\sigma_k}, \qquad \widehat{F}_{t+1} = \sigma_{k+1}\hat{x}_{t+1} + \mu_{t+1}$$

where F_t is the streamflow at time t, x_t is the standardized and detrended streamflow, μ_k and σ_k are the yearly cyclo-stationary mean and variance (day k of the year related to day t), and $\widehat{}$ refers to a forecasted quantity.

We consider the following model structures (this list will be referenced to later in the document):

0. Yearly moving average: $\widehat{F}_{t+1} = \mu_{t+1}$. This naïve model will serve as a baseline for comparisons.
1. Linear AR(N) models: (size($\vec{p}$) = N+1)
$$\hat{x}_{t+1} = p_0 x_t + p_1 x_{t-1} + \cdots + p_{N-1} x_{t-N+1} + p_N$$
2. Polynomial AR(N) models: (size($\vec{p}$) = N+1)
$$\hat{x}_{t+1} = p_0 x_t^N + p_1 x_{t-1}^{N-1} + \cdots + p_{N-1} x_{t-N+1} + p_N$$
3. Linear proper ARX(N, P) models: (size($\vec{p}$) = N+P+1)
$$\hat{x}_{t+1} = p_0 x_t + \cdots + p_{N-1} x_{t-N+1} + p_N + p_{N+1} u_t + \cdots + p_{N+P} u_{t-P+1}$$
4. Linear proper ARX(N, T) models: (size($\vec{p}$) = N+T+1)
$$\hat{x}_{t+1} = p_0 x_t + \cdots + p_{N-1} x_{t-N+1} + p_N + p_{N+1}\tau_t + \cdots + p_{N+P}\tau_{t-P+1}$$
5. Linear improper[1] ARX(N, P+1) models: (size($\vec{p}$) = N+P+2)
$$\hat{x}_{t+1} = p_0 x_t + \cdots + p_{N-1} x_{t-N+1} + p_N + p_{N+1} u_{t+1} + \cdots + p_{N+P+1} u_{t-P+1}$$

---

[1] The improper inputs, u_{t+1} of rainfall and τ_{t+1} for temperature, were considered as outsourced: these are not prediction performed by another calibrated model, but they are considered as if they were observations; this is feasible because complex but accurate models exist, in general, for rainfall and temperature estimation. Autoregressive modelling for one-day-ahead forecasting was attempted for both inputs, however the rainfall model showed no sign of dependency from the model order. The temperature model showed adequate "learning" behavior (best model order AR(17)), however prediction errors were high enough as to influence the streamflow prediction negatively.

6. Linear improper ARX(N, T+1) models: (size($\vec{p}$) = N+T+2)
$$\hat{x}_{t+1} = p_0 x_t + \cdots + p_{N-1} x_{t-N+1} + p_N + p_{N+1}\tau_{t+1} + \cdots + p_{N+P+1}\tau_{t-P+1}$$

7. Linear proper ARX(N, P, T) models: (size($\vec{p}$) = N+P+T+1)
$$\hat{x}_{t+1} = p_0 x_t + \cdots + p_{N-1} x_{t-N+1} + p_N +$$
$$+p_{N+1}u_t + \cdots + p_{N+P}u_{t-P+1} +$$
$$+p_{N+P+1}\tau_t + \cdots + p_{N+P+T}\tau_{t-T+1}$$

8. Linear mixed ARX(N, P+1, T) models (improper in rainfall input, proper in temperature input): (size($\vec{p}$) = N+P+T+2)
$$\hat{x}_{t+1} = p_0 x_t + \cdots + p_{N-1} x_{t-N+1} + p_N +$$
$$+p_{N+1}u_{t+1} + \cdots + p_{N+P+1}u_{t-P+1} +$$
$$+p_{N+P+2}\tau_t + \cdots + p_{N+P+T+1}\tau_{t-T+1}$$

9. Linear improper ARX(N, P+1, T+1) models (improper both in rainfall and temperature inputs): (size($\vec{p}$) = N+P+T+3)
$$\hat{x}_{t+1} = p_0 x_t + \cdots + p_{N-1} x_{t-N+1} + p_N +$$
$$+p_{N+1}u_{t+1} + \cdots + p_{N+P+1}u_{t-P+1} +$$
$$+p_{N+P+2}\tau_{t+1} + \cdots + p_{N+P+T+2}\tau_{t-T+1}$$

The following table show the results for each model in its best parametrization and order:

Model ID (see previous list)	Best model order			Metrics (m³/s)			% improvement w.r.t latest best		
	N	P	T	J_{FINAL}	J_{FINAL}^H	J_{FINAL}^L	J_{FINAL}	J_{FINAL}^H	J_{FINAL}^L
0	/	/	/	33.25	98.61	26.05	/	/	/
1	4	/	/	6.63	17.71	5.59	- 80.0%	- 82.0%	- 78.54%
2	1	/	/	9.69	25.20	8.54	+46.1%	+ 42.3%	+ 52.8%
3	4	3	/	5.59	12.57	4.91	- 15.7%	- 29.0%	- 12.2%
4	4	/	15	6.33	16.30	5.48	+ 13.2%	+ 29.7%	+ 11.6%
5	4	3	/	5.25	11.96	4.47	- 6.1%	- 4.9%	- 8.9%
6	6	/	15	6.32	16.36	5.43	+ 20.4%	+ 36.8%	+ 21.5%
7	4	3	15	5.40	11.68	4.74	+ 2.9%	-2.34%	+ 6.0%
8	4	2	15	5.22	11.46	4.50	- 0.57%	-1.9%	+ 0.7%
9	4	2	15	5.25	11.46	4.49	+ 0.57%	0.0%	+ 0.4%
9	4	3	15	5.12	11.91	4.45	-1.9%	+3.92%	-0.44
Best model ID 8	4	2	15	5.22	11.46	4.50	% improvement w.r.t model 0		
							- 84.3%	- 88.4%	- 82.7%

The best linear model is chosen not just by comparing J_{FINAL} values, but it is chosen with particular interest to both model complexity, where lower complexity is to be preferred is little is to be gained, and J_{FINAL}^H values, where it is preferred to be lowest possible even if this means sacrificing a little in the other two cost functions (*Premises*). The two best choices would be model 5, as it shows very good improvements in performance with respect to simpler models, and model 8; model 8 shows improvements with respect to model 5 in J_{FINAL} and J_{FINAL}^H and a negligible loss in J_{FINAL}^L, which is the less important metric. Also model 9 of lower order shows similar characteristics, however it is not preferred as model 9 contains an improper component for the temperature. Model 9 of higher order shows a noticeable improvement in J_{FINAL}, however even more noticeable is the loss in performance for J_{FINAL}^H, at a cost of having a temperature improper component.
It should be noted that, for all the models listed in the table, the correlogram of the prediction errors time series showed that they were all white noises.

Non-linear parametric models: artificial neural networks

Artificial neural networks represent a very valuable tool in modelling nonlinear models if a considerable amount of data is available for calibrating the model. Their advantage consists mainly in the Principle of Universal Approximation, the abundance of frameworks to build and conceive virtually any model, and the versatility in modelling nonlinear models and phenomena such as the one we are interested in. However, the degrees of freedom granted by ANN impose a toll on the number of parameters the designer of the network has to tune. Specifically, the designer has to choose whether to preprocess data and if so the best preprocessing, learning rate, learning rate exponential decrease, number of inputs and type of inputs with related order, properness or improperness of input, loss function, optimization algorithm, and most important the neural network structure itself. In the recent literature there is no lack of papers tackling this problems, however they propose state-of-the-art solution often involving very advanced model structures (such as long-short term memory neural networks).

Many of these parameters were chosen empirically, such as preprocessing (same normalization procedure used for linear models), learning rate and exponential decrease ([100e-3÷50e-3], coefficient 0.95), loss function (mean squared error), optimization algorithm (stochastic gradient descent, as it is well suited for regression problems). These parameters have been chosen after many trials in different configurations.

The structure of the neural network is the key role in this problem. An iterative process considering different model structures resulted in the choice of four main models to be tested and validated:

a) Shallow linear NN: it is composed of a linear input layer, a rectified linear unit function (ReLU), and a linear output layer.
b) Shallow non-linear NN: it is composed of a linear input layer, a sigmoid layer followed by a rectified linear unit function, and a linear output layer.
c) Deep linear NN: it has the same layer stack of (1) but repeated three times (linear-ReLU x 3, linear output). It was observed empirically that for models with low number of inputs such as our case more layers are not beneficial.
d) Deep non-linear NN: a linear-Tanh-ELU stack followed by a linear-Sigmoid-ELU stack.
e) Very deep non-linear NN: a stack of 5 hidden layers with various non-linear activation functions such as tanh, ReLU, sigmoid, ELU.

Since training is performed on normalized data (both inputs and observed values were standardized), there is an intrinsic limit to the fitness of the models. Data get normalized by extracting the moving mean and standard deviation, they are fed into the network which returns a forecast value which has to be de-standardized to obtain the final forecast value in the appropriate units (m^3/s). This implies that, being the network perfect at its task of forecasting on normalized data (zero error forecasting), there would still be a ground error given by the de-standardization part of the model. However, this "intrinsic" error resulted insignificant, in the order of 10^{-5} m^3/s. This magnitude is the lower limit for any prediction error in this analysis.

The number of inputs and specifically what inputs to chose and the order of the model are chosen in the same way as for linear models (list 1.-9., page 4).
If we were to test all possible combinations of network structures and models and model orders, this would mean to have an unfeasible number of possibilities to test.
The search for the best model structure, inputs and order does not have to be blind, in fact we chose the model order ranges to be tested by considering the best results obtained in the linear case.

We test the four NN models (a-d) for the simplest cases and, from there, the best-performing NN structures are chosen. From there, input complexity is increased as we test situations 3. through 9.

For each, all the most likely-to-succeed permutations are tested. The best input order is chosen to be the best one and, if similar to other orders, the smallest one is chosen.

NN model	Input configuration	Best input order			Loss (m³/s)		Improvement w.r.t.:			
		N	P	T	J_{FINAL}	J^H_{FINAL}	Similar linear model		Best linear model	
a	1	4	/	/	6.52	/	-1.6%		+18.1%	
b	1	4	/	/	7.13	/	+7.5%		29.2%	
c	1	5	/	/	9.55	/	+44.0%		+73.0%	
d	1	4	/	/	6.51	/	-1.8%		+17.9%	
e	1	4	/	/	6.55	/	-1.2%		+18.6%	
a	3	4	3	/	5.34	/	-4.5%		+2.3%	
b	3	3	3	/	5.63	/	+0.7%		+7.8%	
d	3	4	5	/	5.29	/	-5.3%		+1.3%	
e	3	3	8	/	5.30	/	-5.2%		+1.5%	
a	5	3	3	/	4.86	/	-7.4%		-6.9%	
d	5	5	3	/	4.66	/	-11.2%		-10.7%	
e	5	5	4	/	5.02	/	-4.4%		-3.8%	
a	4	3	/	3	6.14	/	-3.0%		+17.6%	
a	6	3	/	5	6.01	/	-4.9%		+15.1%	
a	7	2	3	5	5.20	9.16	-3.7%	-21.6%	-0.4%	-20.0%
b	7	2	4	3	5.49	9.94	-1.7%	-14.9%	+5.2%	-13.2%
c	7	5	4	6	5.33	8.98	-1.3%	-30.0%	+2.0%	-27.6%
d	7	3	4	5	5.08	8.65	-5.9%	-25.9%	-2.7%	-24.5%
e	7	3	4	4	5.22	9.05	-3.3%	-22.5%	=	-21.0%
d	8	4	3	3	4.80	7.87	-8.0%	-31.3%	-8.0%	-31.3%

Overall, the best neural network model is chosen to be the one highlighted in greed in the table. Its order is reasonable, as well as the model structure and its parameters. Moreover, with respect to the linear exogenous models, it shows significant improvement in both low-flow performance and high flow performance. It is clear that model "d8" has better overall qualities, however it implies the use of improper information, of which we do not rightly dispose if not in a hypothetic situation; it was decided that the inputs of the models shall be just the ones known up to time t.

It should be reported that even if the best input order varies from model to model, the significance of the difference between one model order and the other is not very much significant for N>3, P>3, T>3.

Also, neural network training is not a deterministic algorithm: the same network trained on the same data can lead to slightly different performances in validation. These oscillations play a role in choosing the best model.

Some conclusions can be drawn with enough certainty tough, both considering the linear modelling, the neural network modelling shown in the table and other not-reported trials:

1. In this system, exogenous inputs play a very important role, where the rainfall contributes more than the temperature. All three contributions should be considered.
2. An improper system performs generally better than a proper system for forecasting.
3. The best model order for forecasting in this system are: $3 \leq N \leq 5, \ 3 \leq P \leq 4, \ 3 \leq T \leq 6$.
4. Considering the day of the year as input (converted as a sine-cosine tuple) does not show noticeable improvement in predictions.

5. Data preprocessing is necessary to better train the network. Data rescaling-only associated with day of the year does not show improvements in prediction.

The selected network has overall good qualities, however it is prone to underestimation when very high inflow values are involved.
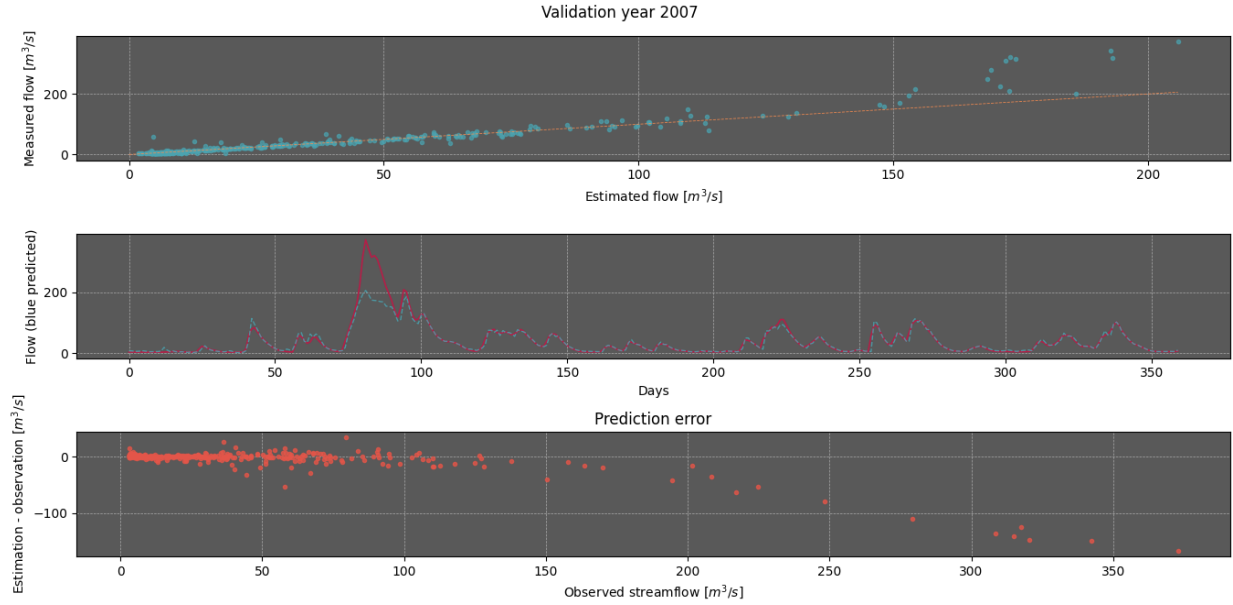
This issue has been mitigated by coupling networks "c" and "d" by means of a last shared bilinear layer output. Although the performance of the resulting network is comparable with the related tabulated results ("d7" model order, $J_{FINAL} = 5.57$, $J_{FINAL}^{H} = 9.77$), it solves the underestimation problem for high values. Also the correlation between the validation performance in low and high flow increases during training. The following represent the performance of the final network in two validation years, the easiest year (2004) and the harshest year (2007):
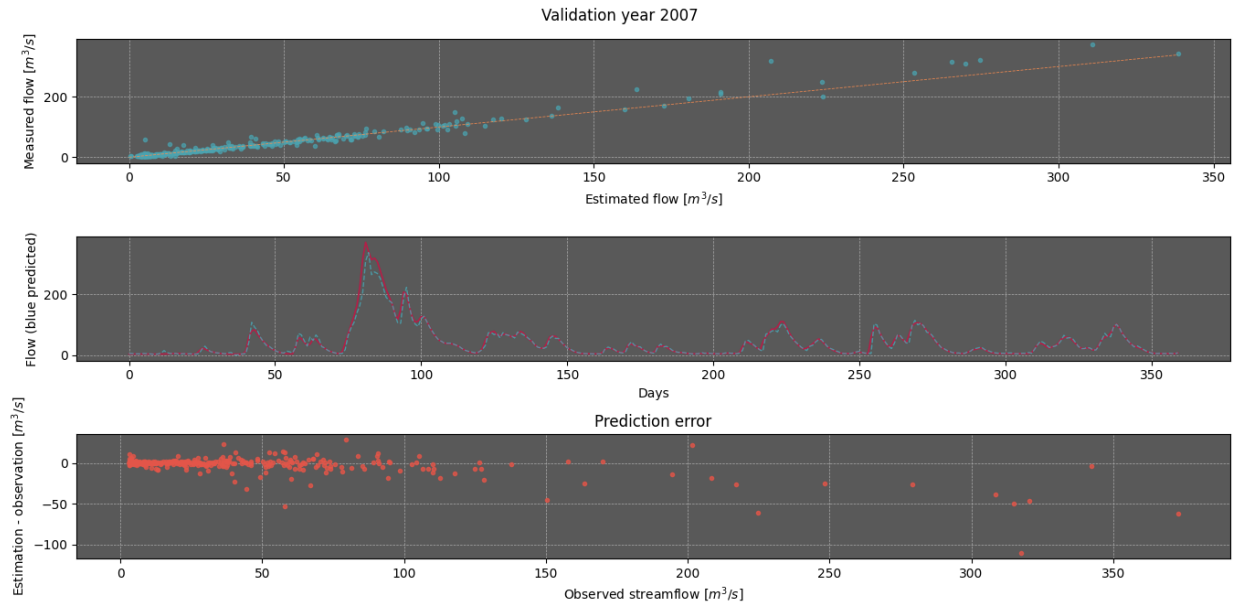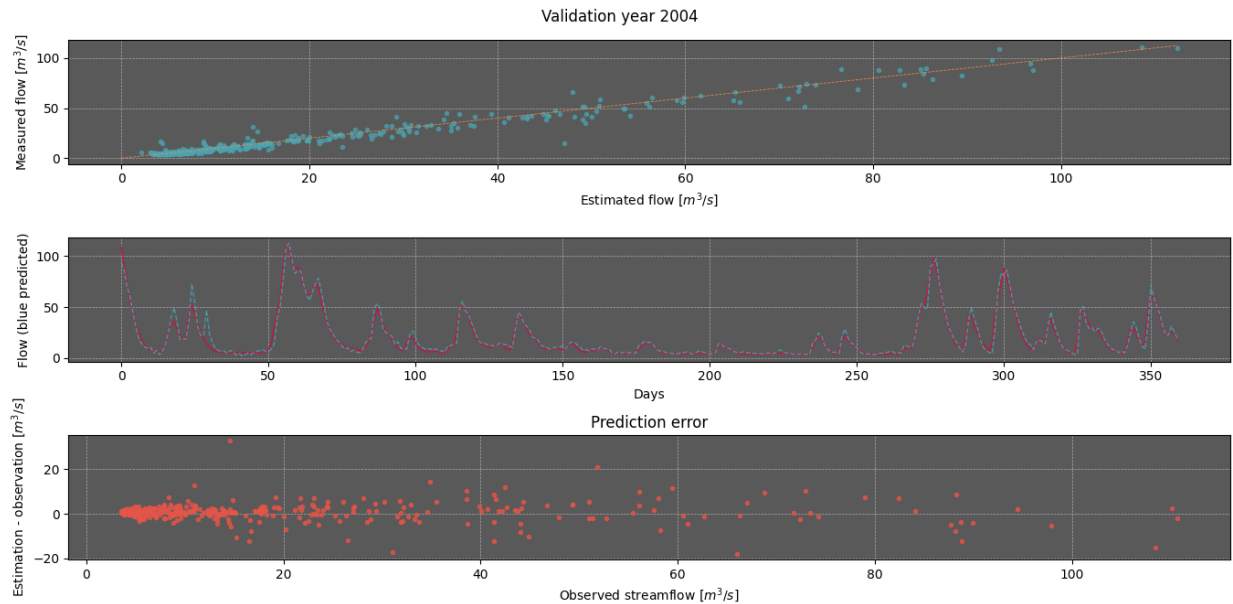
Non-linear parametric models: regression trees and random forests

Classification and regression trees (CART) represent a somewhat simpler approach to modelling than neural networks. The algorithms were developed by using the "sklearn" python library as backbone.

Although being simpler than neural networks, some parameters have to be chosen optimally to obtain the best results without overfitting, a very easy condition to incur to when working with CARTs. The parameters to be chosen optimally are:

1. Optimal maximum tree depth: the optimal interval is between 7 and 10, with 7 favoring performance for low flows, 10 favoring performance for medium and high flows. More than 10 is unnecessary and can lead to overfitting.
2. Optimal minimum number of samples required to split an internal note: between 3 and 7, where 3 and 4 favors performance in medium and high-flow predictions, while 7 favors low-flow predictions.
3. Optimal minimum number of samples required to be on a leaf node: found to be 9 favoring low-flows predictions and 4 favoring medium and high-flow predictions. This parameter has the effect of smoothing the model in regression.
4. Maximum number of leaf nodes: limiting the number of leaf nodes does not bring any benefit. The performance increases up to abut 250 leaves, after which it simply stops improving.

Optimal parameters are chosen by cycling through a set of possible parameters values, cycled 40 times to obtain statistically meaningful results (tree building can vary a lot from one run to another, thus a smoothing action is needed to compare statistical result), and then cycled through the usual cross-validation process over the 20 years. These parameters were evaluated with respect to three ranges: high flow conditions, low flow conditions, all data without distinction.

Model order identification is performed with the same 20 years cross-validation procedure performed on the other models.

Modelling start considering all three inputs in their proper form since previous modelling attempts showed that the best results are obtained when considering flow as well as rainfall and temperature data. The first model is a simple regression tree, splitting through recursive binary splitting. The first noticeable feature is that both temperature and rainfall contributions seem to worsen prediction performances. The best model order in this case is *N=2, P=1, T=1*, with $J_{FINAL} = 7.30$, $J_{FINAL}^H = 16.15$. The worst effect is given by temperature contributions, while rainfall contribution, while decreasing the performance, are not as relevant as temperature. To validate this result, a streamflow-only model was cross-validated, and again it appears that the best model order is N=2, however in this case $J_{FINAL} = 8.29$, $J_{FINAL}^H = 17.06$. The same pattern represents itself when using both streamflow and rainfall past data, and once again the best model order is *N=2, P=1*, which results in $J_{FINAL} = 7.34$, $J_{FINAL}^H = 15.84$.

The model performs decently – even if not remotely close to linear models and ANN - for low and medium flows, however for some validation years (2007) this simple model is not able to forecast the very high peaks of inflow. This is easily explained by considering that in 2007 is registered the highest, unprecedented inflow among the 20 years. Since the benefit of adding temperature contribution is minimal if none, we will proceed exploring other models by using the *N=2, P=1* model order.

Bagged trees are created as an ensemble of the simple tree model: each tree is trained on a random subset of the training set, and the one-day-ahead-prediction is computed as the average of all the trees outputs. The multiple training makes this model more computationally expensive, however performances benefit from the randomization. About 80% of the training set is bootstrapped for each tree.

Number of trees	Best model order	J_{FINAL} (m³/s)	J_{FINAL}^H (m³/s)
10	N=2, P=1	6.58	14.30
30	N=2, P=1	6.50	14.09
100	N=2, P=1	6.43	13.89
200	N=2, P=1	6.42	13.88

Random forests tests showed that, once again, the best model order is *N=2, P=1*. For every tree, 70% of the training set is bootstrapped for training. Since the input variables are three (x_t, u_t, x_{t-1}) the random trees are allowed to randomly choose one of the three variables to perform splits. Random forests show a somewhat weak correlation between the number of trees and the quality of the predictions, which becomes insignificant for more than 100 trees, at which $J_{FINAL} = 6.52$, $J_{FINAL}^H = 14.25$. The actual minimum is at 300 trees ($J_{FINAL} = 6.50$, $J_{FINAL}^H = 14.22$), however the knee is at 100 trees and it is very close to the performances at 300 trees, with the addition of being much faster.

Nonetheless, tree-based models remain in this case inferior to artificial neural networks.

Part 2: the dam

The dam design and its operations are generally subjected to many different and conflicting objectives. In the case of the Barrage du Chatelot the actual dam was built to operate as an hydroelectric power plant. By exploring the nearby attractions and its downstream path, the river passes inside and near some small villages on the border between Suisse and France, where the stream sometimes opens up to some beautiful small lakes. Further on, the stream passes through the Doubs Natural Regional Park, to then enter France completely and pass through some camping areas and over-river restaurants and other structures. From the dam up to 10km downstream, the river streams at the bottom of a quite steep, v-shaped valley with no structures inside it.

No record was found which states that the river is used as primary irrigation source for the area.

Given these considerations, the main stakeholders are ranked here in order of importance:

1. Hydroelectric power production: as of today, power production is the main noticeable trait of this river and its primary source of interest.
2. Environment preservation: since the river passes into a regional national park, preserving the habitat downstream is an important aspect of the design, both for environmental reasons and for touristic reasons.
3. Flood prevention of downstream river: the river passes through many attractions and small towns.
4. Dam structural integrity: water should not overtop the dam, as its overtopping is an hazard both for its structure and for flood prevention downstream.
5. Irrigation: it was found no proof of the use of this river for irrigation purposes.
6. Flood prevention on the lake shores: on the whole lake shore there is just one man-made structure, a kayak club called Départ kayak.
7. Tourism: the Départ kayak club is a touristic attraction which needs the lake levels to not fall too much.

Many of these goals are somewhat compatible with each other, however never entirely. For example, the Départ kayak club would want to have high lake levels to appeal its customers which is compatible with stakeholder 1, however not too high as for scaring customers away or making the club facilities inaccessible, which is in contrast with stakeholder 1.

Also the total filling time of the dam up to its nominal level should be considered as a valuable metric.

Decision variables. As for what concerns the dam design, the decision variables are the following: the *height of the dead storage*, the *height of the active storage* which is also considered the nominal operational height of the dam, and the *total height of the dam*; these quantities are strictly related to the capacity of the storage capacity.

Another decision variable should concern the maximum possible release, or release capacity, via the spillways and what happens in the extreme situation of the water level threatening to overtop the dam. We assume for this exercise that the *total maximum cross-section of the main spillways* is the decision variable. For simplicity we assume n spillways of fixed diameter 0.56 meters (1 spillway = 0.2463 m^2 typical value for a 100kW turbine).

When the water is overtopping the dam the water discharges from the top by design choice, at a rate of $r_{overtop} = 15(h - h_{dam})^2 \ m^3/s$.

The management variable will be the fraction u of the maximum release capacity, $u \in [0,1]$, and it will be considered a continuous variable.

Alternatives. Many alternatives can be considered.

- A0: The basic, no-dam alternative will be the low-end benchmark against which to compare the indices.
- A1: This alternative will be the opposite of A0: given the topological constraints of the area nearby the existing dam, the elevation of the lowermost point is 680m, while the valley could topologically store water up to a height of 800m, allowing for a 120m jump instead of the 40m of the actual dam. Since more space generally means better performance for every index, this will serve as a high-end benchmark.
- A2: An alternative closely corresponding to the existing dam design.
- A5: A dam with characteristics extrapolated from the reservoir design procedure.

Alternative	Dead storage height [m]	Active storage height [m]	Total dam height [m]	Number of spillways
A0	/	/	/	/
A1	8	110	120	12
A2	5	70	74	5, 8, 12
A5	5	5+68.47+2	80	11 (Min:2)

Reservoir model. To model the reservoir, some simplifying assumption are made: it is assumed a cylindrical reservoir with constant surface area, negligible infiltration and temperature-independent evaporation rate. The rain contributes to the net inflow to the lake. The outflow was modelled following a simplified version of the Bernoulli principle to find the speed of the outflow, which multiplied by the available cross-section defines the outflow:

$$\frac{v^2}{2} + gz + \frac{P}{\rho} = const. \implies g(h - h_{min}) \cong \frac{v^2}{2} \implies$$
$$\implies v \cong \sqrt{2g(h - h_{min})} \implies Q_{out} = v \cdot A_{cross-section} \ [m^3/s]$$

This allows a simple yet more accurate physical simulation of the system and also allows to define the energy generated from the power plant on a daily basis as:

$$P_{out} = \eta \cdot P, \qquad P = \frac{1}{2}\rho A_{cross-section} v^3, \qquad \eta = 90\%$$

The model is set to stop producing energy if the water overtops the dam.

Indicators. Indicators should be developed for each interest of the stakeholders. Some indicators can represent multiple stakeholders' objectives. With reference to the introduction to this chapter, the followings are the indicators that will be implemented:

- Power production *yearly average*, *reliability*, *vulnerability*, *resilience* as defined in (Hashimoto, 1982). These indicators should accommodate stakeholder 1 and partially 6 and also, in a way, also 3 and mainly 7, as downstream floods would be caused mainly by overtopping of the dam, which is very heavily penalized in the *vulnerability* and *resilience* indices.
- *Low* and *High Pulses* indicator for environmental impact (stakeholder 2). The high pulses indicator is also useful as a downstream flood indicator, accommodating stakeholder 3. The low pulses

indicator can be seen as a not-so-far proxy for an irrigation indicator, since irrigation is not one of the most important stakeholders in our case.

- Assuming that, whatever the dam alternative, only the Départ kayak club is build near the shore, the shore flood indicator is expressed as the annual mean number of days with flow events hitting the club structure in a damaging way (active storage height + 3m).

Together with these indicators, it should be presented another indicator, which corresponds to the amount of days needed to make up for the economic (electric power) loss given by the filling time of the dam. This "indicator" only concerns to the first transient of the life of the dam. Since it cannot be optimized very much by the management strategy of our design, we will just mention it and not go any further.

Scenarios. To design and test the reservoir and the dam we will use the 20-years time series of the inflow and rain.

Reservoir storage capacity design

Storage capacity will be designed mainly considering hydroelectric power production.
First, we have to define a reasonable power target. To do so, we have to define a release quantity such that, were the full dam to constantly release this quantity without missing a day, on the long run the reservoir would increase its level: this is a safe quantity to spill every day which does not empty the reservoir on the medium-long run. This quantity is set to the 45th percentile of the streamflow time series data:

$$\overline{Q_{out}} = Q_{45} = 16.59 [m^3/s]$$

This flow is such that, if the storage is enough, then it is possible to have $Q_{out} > \overline{Q_{out}}$, while if the storage is running low, then $Q_{out} = \overline{Q_{out}}$. The target power should be set in relation to this quantity, since the powerplant should be able to surpass the target power most of the days. For a modest dam with a jump of 50mt the output power is 7.3 MW. Since the topological conformation of the territory should theoretically allow for dams high up to 120 mt, we will take a reasonable power value as the stakeholder's target power to be $\overline{P_{out}} = 10$ MW. We will also consider seasonality, meaning that the power requirement is higher in winter and lower in summer by 10% of the nominal power.

Sequent peak analysis is used, together with considerations regarding the expected height of the water level at equilibrium. By considering $\overline{Q_{out}}$ as the target release that should always be possible to actuate, sequent peak analysis suggests that the dam should have a capacity of at least 0.109 km³, corresponding to a minimum dam height of 0.111 m. This means that, for the sole purpose of storing enough water to be able to provide a constant release of $\overline{Q_{out}}$, a dam 0.111 meters high would be sufficient.

The real design choice for the height of the dam relies entirely on power production.
Consider the two relationships:

$$\begin{cases} \overline{Q_{out}} = A_{cross-section} \cdot n_{spillways,min} \cdot \sqrt{2g\Delta h} \\ \overline{P_{out}} \leq \eta \frac{1}{2} \rho \overline{Q_{out}} \cdot 2g\Delta h \end{cases}$$

From the second equation we find the sufficient height necessary to deliver a power equal or greater than $\overline{P_{out}}$ and keeping a constant $Q_{out} = \overline{Q_{out}}$, while the first equation takes care of keeping the release flow

constant by leveraging the number of spillways.
The two solutions are:

$$\begin{cases} n_{spillways,min} = 1.83 \approx 2 \\ \quad \Delta h \geq 68.47 \ [m] \end{cases}$$

The maximum number of spillways should be designed so that the dam can release as much water as the maximum recorded inflow value when the water level is at its nominal operational point of 68.47 meters above the dead storage. This would however be a hazard to the downstream communities and environment; some stakeholders could however be more inclined at sacrificing the environment in case of an extreme event rather than risking damages to the structure due to overtopping and stopping power production.

By considering Q_{max} to be equal to the 95th percentile of the inflow time series, $Q_{95} = 102 \ m^3/s$. This value is quite higher than the maximum value allowed by environmental constraints, however simulations proved that a high value is necessary to not overtop the dam. Otherwise, were $Q_{max} = Q_{75}$, the total dam heigh should be much higher to accommodate any extra water that would accumulate over time due to insufficient release:

$$Q_{max} = Q_{95} = 102 \ m^3/s = A_{cross-section} \cdot n_{spillways,max} \cdot \sqrt{2g\Delta h}$$
$$\Longrightarrow n_{spillways,max} = 11.29 \approx 11$$

Alternative A5 results very similar to the design in use today for what concerns the height of the dam. Alternative A5 will have a dead storage height of 5m, an active storage height of 5m+68.47m+2m=75.47m, a total dam height of 80 m, and 11 spillways.

Standard policy design

The standard policy is designed following the standard operating policy scheme. We assume that, from regulations, $h_{min} = h_{dead-storage} + 2 \ [m]$ and $h_{max} = h_{active-storage} + 1 \ [m]$. These values were arbitrary chosen and will simulate an external imposition by part of laws and regulations. This means that, regarding power production, h_{max} will be the point which maximizes power production, however it is the point which maximizes the release as well. The control variable is the fraction of the total maximum release, mimicking the action of a continuous valve. This is necessary as the release of the dam is controlled by the Bernoulli equation, and no pumps are considered in the system, so that no active (forceful) control of the release is assumed.

The policy regulating the reservoir release is sought in the family of piecewise-linear, time-invariant functions with three control parameters. More parameters could be set; however, three parameters allow the operating policy to be smooth, without jumps near the h_{min} and h_{max} limits which can make release and power production highly discontinuous, a feature which is not much appreciated when considering almost any system.

Training the operating policies and testing the alternatives

The goal is to optimize a control strategy with respect to multiple conflicting objectives (by simulating different stakeholders' interests) starting from a standard operating policy consisting of a piecewise linear function. Thus, it will not be possible to define one "best" performance, as any stakeholders may consider the same solution acceptable or not acceptable. Thus, I will limit myself at finding the best set of solutions through the following methods, without superimposing any judgement.

The developed optimization procedure could be considered as a kind of Evolutionary Multi-Objective Direct Policy Search (EMODPS), however no particular already-existing algorithm or library is used, rather I preferred to build a simple version from scratch myself.

The algorithm starts from a population of alternatives, in this case it initialize the same dam model but different, randomly initiated operating policies. Each individual is tested by simulating the system for a long enough time period, after which the performance indicators are computed. The algorithm is built in such a way that any number of indicators can be used; the only constraint is that the indicators should be built so to return lower values as performances increase. As the number of considered indicators goes up, also the population should increase in its numbers accordingly.

All the alternatives are ranked in order of performance, where all the alternatives which belong to the pareto frontier come first, and all the others come after. Here either all or just half of the population will be considered for mating (considering all the population without any cut-off proved to better preserve diversity and preserves dominant solutions across generations).

Mating can happen between 2 or more parents. Each alternative of the pareto front is chosen to be a parent by drawing from a uniform distribution, and each have higher probability to be chosen than the other non-dominant alternatives. The non-dominant alternatives get drawn from the end-tail of a geometric distribution.

When mating between parents occur the "DNA", represented by the three parameters of the operating policy, gets recombined between the parents, where every parent contributes to the child DNA of a random amount drawn from a uniform distribution.

Mutations can be either punctual of global, they happen rarely, but commonly enough so to thoroughly explore the parameter space. A mutation can completely reset a child's DNA randomly, or it can mutate its parameters by a random quantity drawn from a gaussian distribution with appropriate specified variance.

After each cycle (generation) there is the same number of individuals as there were at the beginning.

To be effective, the algorithm should run for many generations and with a high enough number of individuals. A high number of individuals is crucial to preserve diversity among the solutions and have better chances of finding better and better solutions.

Results

The program "NRMproject02" will automatically display the results after the optimization procedure in an interactive way.

Simulations show that all the proposed alternatives, exception made for A0 (undammed river) can grant the minimum power requirement (which admittedly was set a little too low to be a real challenge for the considered alternatives). As for the objectives, power requirement (objective 1) and environmental high pulses (objective 2) were considered, as they are the most representative of all the stakeholders (only two objectives were chosen for visualization purposes, however the developed algorithm can take any number of objective).

As expected, alternative A1 can reach the highest peaks in power production and release flow, with power production peaks at 100MW and release flow peaks at 120 m^3/s.

Alternative A2 shows that 5 spillways are not enough to handle the reservoir inflow, and it ends up overtopping. However, 8 spillways are already enough to handle the inflow. Maximum power peaks and release peaks top at 60MW and release flow peaks at 100 m^3/s, respectively, with no tangible distinction

between 8 and 12 spillways.
Alternative A5 has performances almost identical to alternative A2 with 12 spillways, being the height and structure of the dam very similar.

For what concerns the release (considering A5 as an example), no policy is able to keep the release under the 75 percentile limit: even with the release valves fully open to keep the water level as low as possible the system stabilizes at around 80 m^3/s with an output power of 32 MW on average.

Being the power threshold as previously defined, the best operating policy to minimize release is to keep the valve fully open at all times (in the limits provided by regulations), while to maximize the produced electric power the best operating policy is to keep the valves closed to maximize water height, and then open them just before the water approaches the regulations higher limits. Optimal solutions are found in between, and the choice of the "optimal" solution should fall on the negotiation process.
For what concerns flood control, the alternatives A1, A2[8], A2[12] and A5 have no trouble at preventing floods (at the cost of a high release, up to 120 m^3/s) and act as a low-pass filter, cutting off any high frequency disturbance. This means that, while the inflow can peak at up to 370 m^3/s, the release will never see these peaks and the lake level, thanks to the release capacity of the solutions, will never rise much above to the predefined limit (maximum +1.5m). Of course, flood prevention is better performed by dams with a higher number of spillways, so that any anomalous inflow or forecasted inflow can be dealt with.
By considering the irrigation, the proposed A5 solution can deliver, at equilibrium, between 80 m^3/s and 100 m^3/s very reliably (value depends on the operating policy), with as little deviations as ±2 m^3/s at regime.

The EMODPS procedure is able to deliver a set of solutions from which stakeholders can start to discuss and choose the best compromise. However, the developed procedure often fails at considering the "limit" solutions at the boundaries of the decision space (as they are very rare to be drawn from a random set), and this is the reason underlying the previous, handmade considerations about the operating policies.
In the following I report the results of the policy optimization procedure of the A5 alternative, which shows the evolution of the performances generation after generation.

The followings are the evolution of the policies and the indices space if mating and mutation are performed on all the individuals. It can be observed that the solutions tend to cluster as no dominant solution survives from one generation to the following. This behavior allows to freely explore the parameters space with all the individuals, but at the cost of make the solutions converge towards a "drifting center solution", as shown in the two animations.

The solution to this problem is to preserve the dominant policies from one generation to the next one, and fill the remaining places with individuals built from the mating-mutation process.
In this way, at each generation, the dominant policies compete against the newly formed policies, and the results are far more comprehensive, as it is shown in the policies animation and the performance space animation (70 generations, 90 individuals) and here (policies, performance space with 50 generations, 120 individuals. The issue with this kind of search is that, after a few generations, all the points tend to settle into the dominant front, and the algorithm would altogether stop if not for a "safe" number of solutions which always gets mated and mutated, thus allowing the continued exploration of the parameters space. However, this solution is not efficient: it is clearly visible that many points are clustered into the same region of high density, while other regions are much less dense.

To mitigate this problem, mutations were allowed to be performed even on the dominant set of solutions. However, to not lose precious information, not all the dominant solutions should have the same probability of mutating: the isolated solutions should not mutate, while mutations should be allowed for some solutions which are very closely packed together, alas they are "denser".

The "mutation set" contains all the offspring produced from the mating step, and also a certain percentage of the offspring belonging to the densest regions of the dominant solutions (the densest 20%). The rationale behind this is that, given enough generations, all the points in the performance space will converge to some position into the dominant set, thus threatening to block the advancement of evolution. If the convergence happens more towards a specific zone rather than homogeneously, this "density-aware mutation probability" will redistribute the points so that no clumps are created, which means that a broader region of the pareto frontier can be explored, even if more coarsely, at the price of loosing some dominant solutions which are however well approximated by their neighbors.

The result of these considerations are visible in the distribution of points in the performance space and in the policies, as the generations advance. It is clearly visible that, after some generations, the points in the densest regions of the pareto frontier start re-distributing.

The algorithm also detects when the distribution of points is uniform enough, in which case it does not include any dominant solution into the mutation set.

References

Ghimire, S. Y. (2021). Streamflow prediction using an integrated methodology based on convolutional neural network and long short-term memory networks. *Sci Rep - Nature*. doi:https://doi.org/10.1038/s41598-021-96751-4

Kilinc, & Haznedar. (2022, 1). A Hybrid Model for Streamflow Forecasting in the Basin of Euphrates. *Water*. doi:https://doi.org/10.3390/w14010080

Philip Kibet Langat, L. K. (2019, 11). Identification of the Most Suitable Probability Distribution Models for Maximum, Minimum, and Mean Streamflow. *Water*. doi: doi:10.3390/w11040734

https://energyeducation.ca/encyclopedia/Energy_from_water#cite_note-RE1-2

https://structurae.net/en/structures/chatelot-dam

The code and some figures can be found at this link:
https://github.com/Matteoleccardi/2020NaturalresourcesManagement

Python was preferred to Matlab as it provides broader documentation and tools, especially for neural networks through the package "PyTorch".