RCP209

APPRENTISSAGE, RÉSEAUX DE NEURONES ET MODÈLES GRAPHIQUES

Enseigné par Fouad Badran, Michel Crucianu et Meziane Yacoub

PROJET n°4

« Prédiction du débit entrant d'un lac »

Auditeur : Franck Dernoncourt
<*franck.dernoncourt@gmail.com*>

SOMMAIRE

ILLUSTRATIONS

1. INTRODUCTION

1.1. Objectifs

Le but de ce projet est de réussir à prédire le débit entrant d'un lac, le Lac St-Jean, en fonction de l'évolution du débit entrant d'un lac à partir de l'historique de ce débit, de la fonte des neiges et des précipitations dans le bassin hydrographique. Les données permettant ce travail sont déjà recueillies : il s'agira donc de les traiter, les analyser et les utiliser pour construire un modèle capable de prévoir efficacement le débit entrant dans le lac.

Dans une première partie, nous effectuerons une étude liminaire des données à fin d'en extraire les informations générales. En seconde partie, nous tenterons d'établir une classification des données afin de voir les tendances principales. Enfin, en troisième partie, nous construirons plusieurs modèles de prédiction et les évaluerons à l'aide de mesure de qualité.

L'intérêt du projet de mettre en pratique les différentes techniques et théories vues lors du cours et des séances de TP de *RCP209 : Apprentissage, Réseaux de neurones et Modèles graphiques*. Elles seront mises en oeuvre par des scripts MATLAB, en faisant appel à plusieurs toolboxes open source et gratuites. Tous les fichiers utilisés pour élaborer ce rapport sont disponibles à l'adresse : http://files.wiki4games.com/cnam/rcp209/projet.zip

1.2.Fichiers joints à ce rapport

Les trois scripts MATLAB principaux sont :

- *main.m* --> Comme son nom l'indique, c'est le script principal réalisant les calculs et les graphiques de tout les chapitre 2 et 4. Il utilise les toolboxes *utils* et *tsa*, et fait appel aux scripts :
  - *showRandomVariableInformation.m* : pour afficher les données principales d'une variable aléatoire.
  - *corrmap.m* : pour afficher des matrices de corrélation.
  - *mlpWrapper.m* : pour faciliter les appels à *mlpLearn.m* , elle-même faisant appel à la toolbox *Netlab*, tout en faisant des boucles, des calculs et des graphiques supplémentaires.
- *mainKmeans.m* --> Ce script est utilisé dans le chapitre 3 traitant de la classification. Il permet de dérouler les algorithmes de classification hiérarchique et de K-moyennes en faisant uniquement à des fonctions natives de MATLAB.
- *mainKohonen.m* --> Ce script est utilisé dans le chapitre 3 et est issue en grande partie des TP de RCP209. Il implémente l'algorithme des cartes auto organisatrices, dites de Kohonen, en faisant appel à la toolbox *somtoolbox*.

Une attention particulière a été donnée pour commenter tous les scripts de façon détaillée, afin de rendre plus aisée leur lecture est de permettre une réutilisation rapide dans le futur. Pour faciliter leur exécution, les librairies sont également inclus dans l'archive des fichiers fournie avec ce rapport. A noter que la librairie Netlab a été très légèrement modifiée pour mieux convenir à nos besoins.

2. ETUDE LIMINAIRE DES DONNEES

Dans un premier temps, nous allons analyser les données afin de bien comprendre ce que nous manipulerons dans les parties suivantes.

2.1. Le Lac Saint-Jean

Figure 1 - Le lac Saint-Jean

Source : http://fr.wikipedia.org/wiki/Lac_Saint-Jean

2.2. Source des données

The University of Western Ontario Les données sont issues du *Department of Statistical & Actuarial Sciences*, de l'University of Western Ontario. Ce département a une base de données, le *Hipel-McLeod Time Series Datasets Collection,* utilisée dans un cadre académique pour servir de base d'études au livre "*Time Series Modelling of Water Resources and Environmental Systems*" écrits par K.W. Hipel et A.I. McLeod en 1994 [1]. Elles sont disponibles à cette adresse:

http://www.stats.uwo.ca/faculty/aim/epubs/mhsets/thompsto/ [4]

Trois fichiers de données sont disponibles :

1. *lacstjin.1* --> Lac St-Jean, débit entrant (inflows) dans le lac par quart de mois, entre 1953 et 1982
2. *lacstjra.1* --> Lac St-Jean, pluviométrie (rainfall) par quart de mois, entre 1953 et 1982
3. *lacstjsn.1* --> Fonte de neige (snowmelt) par quart de mois, entre 1953 et 1982, Bassin Daval

2.3. Informations de base sur les données

Chacune des trois variables de notre série statistique Inflows, Rainfall et Snowmelt comprend 1440 éléments, car 1440 = 4x12x(1982-1953+1)

****** Inflows summary *****
Number of elements: 1440
Minimum: 5.502170e+001
Maximum: 7.419486e+003
Median: 7.889873e+002
Mean: 1.089712e+003
STD: 1.040589e+003

****** Rainfall summary (*lacstjra.1*) ******
Number of elements: 1440
Minimum: 0
Maximum: 1.083000e+001
Median: 1.085000e+000
Mean: 1.767333e+000
STD: 1.983131e+000

****** Snowmelt summary (*lacstjsn.1*) ******
Number of elements: 1440
Minimum: 0
Maximum: 1.794590e+001
Median: 0
Mean: 6.972892e-001
STD: 1.969771e+000

Nous pouvons constater que les valeurs de la pluviométrie et de la fonte des neiges sont à peu près du même ordre de grandeur (moyenne et écart-type presque égaux) tandis que celles du débit entrant sont bien plus importantes. Il faudrait pour certains modèles de prédiction centrer et réduire les données. Cependant, nous remarquons que si l'on divise simplement les débits entrant par 1000, nous obtenons trois séries statistiques ayant presque la même moyenne et les mêmes écarts-types. Mais ceci n'est peut-être pas optimal car comme nous le montrerons plus tard, centrer et réduire toutes les données donnera de meilleurs résultats que la simple division par 1000 du débit entrant.

En outre, il est intéressant de noter que la médiane pour la fonte des neiges est 0 : cela signifie que pour plus de la moitié des mois, aucune fonte des neiges n'est constatée.

Nous allons à présent dessiner les courbes de ces trois séries statistiques ainsi que les box plots pour mieux visuellement appréhender les données, notamment concernant leur évolution dans le temps et la répartition.

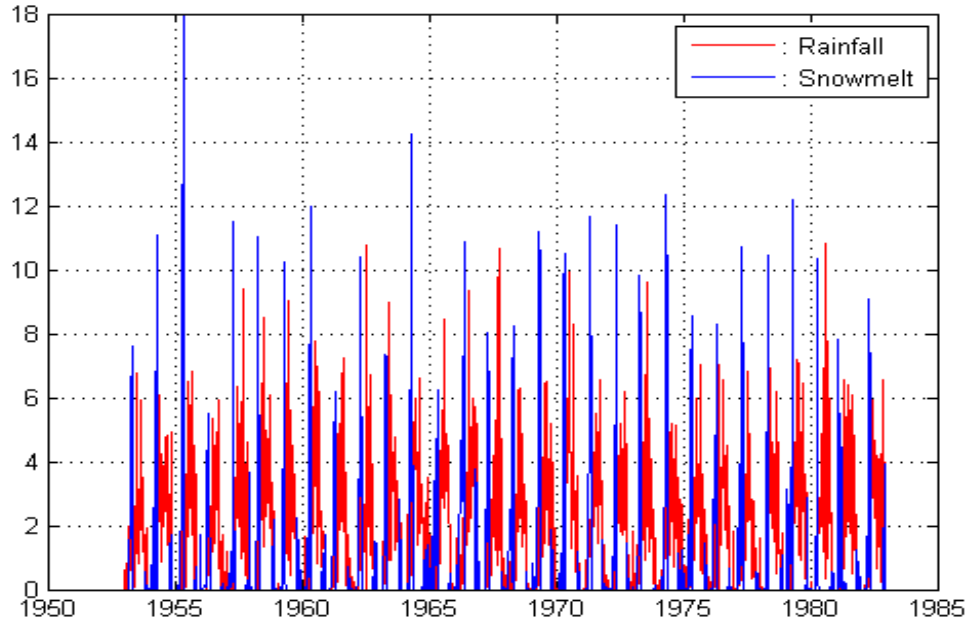


Figure 2 - Rainfall and snowmelt curves

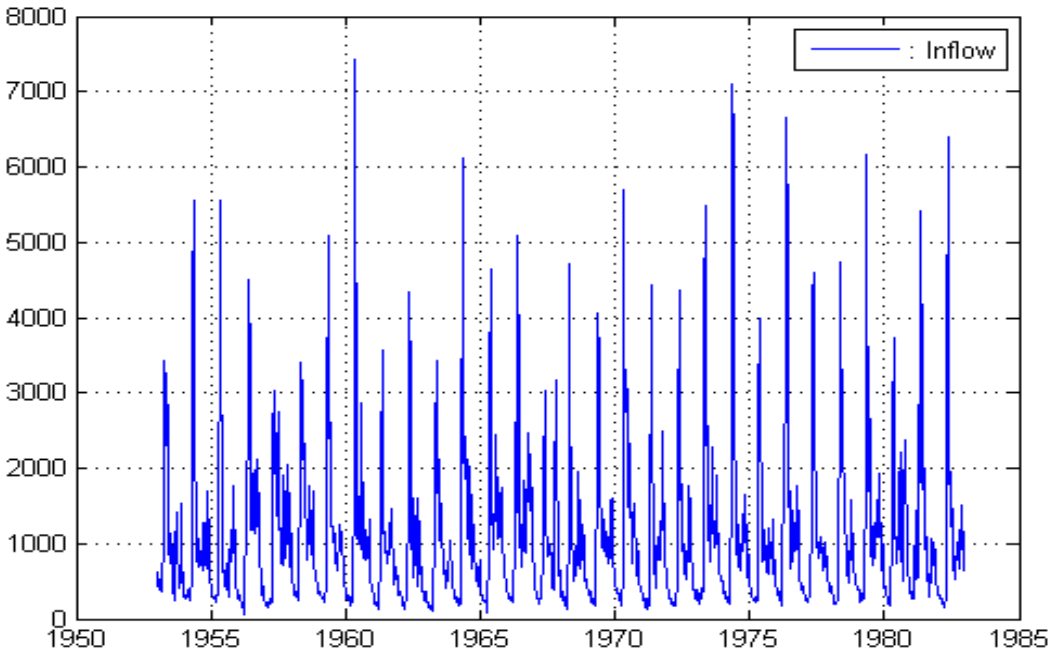


Figure 3 - Inflow curve

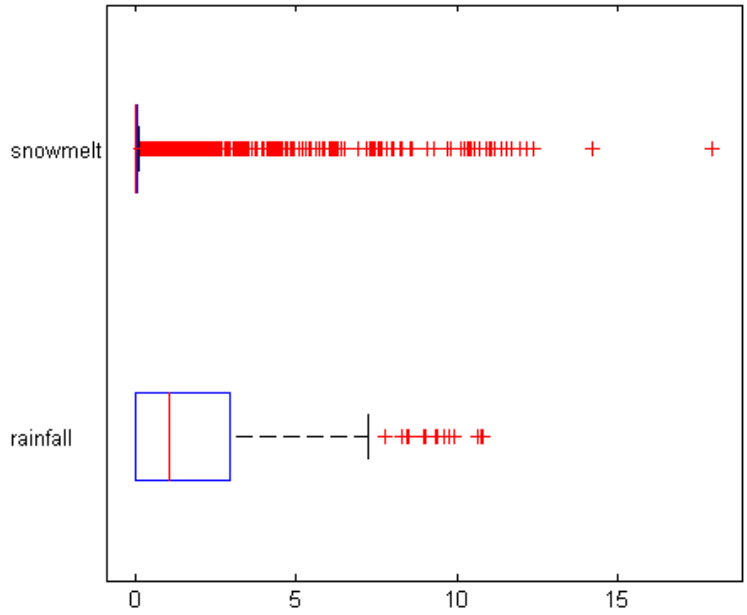


Figure 4 - Snowmelt and rainfall box plots

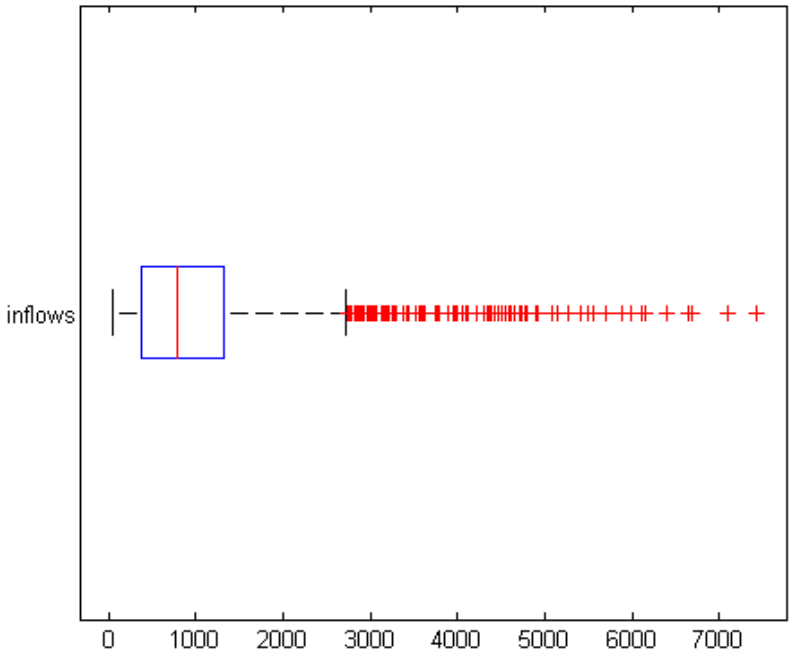


Figure 5 - Inflow box plot

2.4. Matrice de corrélation

À présent, nous allons voir s'il est utile de conserver les trois variables pour le reste de notre étude. Garder une ou plusieurs variables inutiles pourraient mettre en péril l'efficacité de notre modèle futur de prédiction pour le débit entrant dans le lac.

Nous allons nous pencher sur les corrélations entre les trois variables de notre série statistique. À cet effet, nous étudions la matrice de corrélation, grâce à la librairie *utils* de http://home.online.no/~pjacklam/matlab/software/util/index.html [3] ainsi que le script *corrmap.m* du site http://www.eigenvector.com/MATLAB/corrmap.html [5].

Voici les résultats :

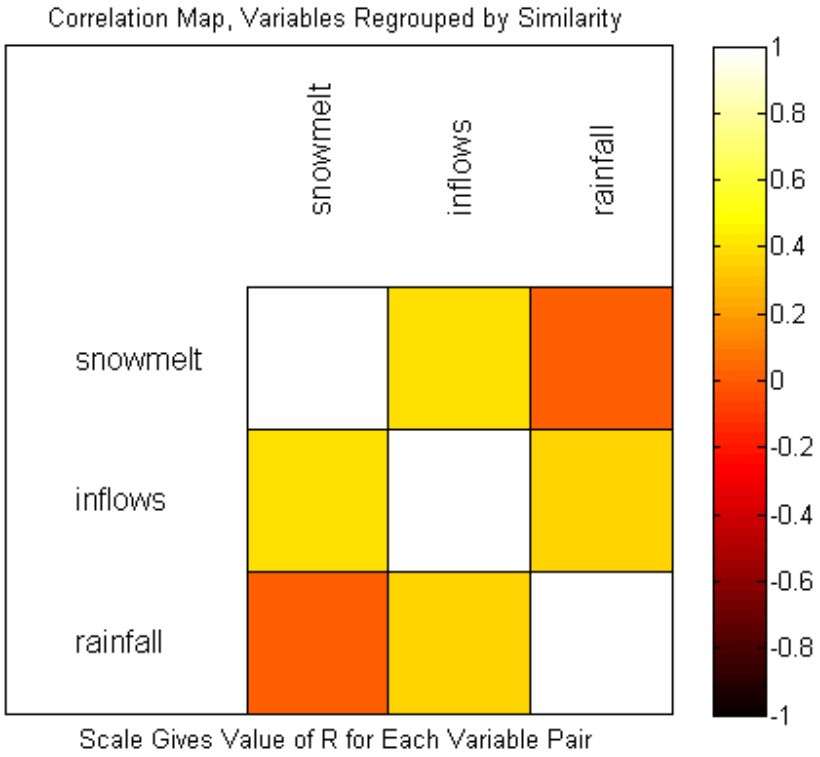


Figure 6 - Matrice de corrélation entre inflow, rainfall et snowmelt

Les valeurs précises sont :

1.0000	0.3616	0.0128
0.3616	1.0000	0.3899
0.0128	0.3899	1.0000

Ces résultats correspondent à l'intuition que l'on pourrait avoir eue au moment de la prise de connaissance de la situation à étudier. La pluviométrie et la fonte des neiges sont corrélées positivement avec le débit entrant. La très faible corrélation entre pluviométrie et fonte des neiges signifie qu'il n'y a pas plus de probabilités qu'il pleuve lors d'un jour de neige que lors d'un jour sans neige. Nous pouvons d'ailleurs le voir sur le graphique du sous-chapitre précédent où sont superposées les courbes de pluviométrie (rouge) et de fonte de neige (bleu).

Comme aucune corrélation est proche de 1 ou -1, aucune variable peut-être presque totalement déduite par un autre. Cependant, nous ne pouvons pas écarter l'hypothèse d'une variable correspondant à peu près à la combinaison linéaire des deux autres. Afin de confirmer ou d'infirmer celle-ci, nous devons faire une Analyse en Composantes Principales.

2.5. Analyse en Composantes Principales (ACP)

Etant donné un ensemble d'observations décrites par des variables exclusivement numériques $\{x_1, x_2, ..., x_p\}$, l'Analyse en Composantes Principales (ACP) a pour objectif de décrire ce même ensemble de données par de nouvelles variables en nombre réduit. Ces nouvelles variables seront des combinaisons linéaires des variables originales, et porteront le nom de Composantes Principales (CP).

Sauf exception, la réduction du nombre de variables utilisées pour décrire un ensemble de données provoque une perte d'information. L'ACP procède de façon à ce que

cette perte d'information soit la plus faible possible, selon un sens précis et naturel que l'on donnera au mot "information". L'ACP peut donc être vue comme une technique de réduction de dimensionnalité.

Faire une ACP nous permettra ainsi de voir, encore mieux que la matrice de corrélation étudiée précédemment, s'il est utile de conserver les trois variables ; autrement dit, si aucune variable ne peut être presque totalement déduite des deux autres.

Voici les résultats :

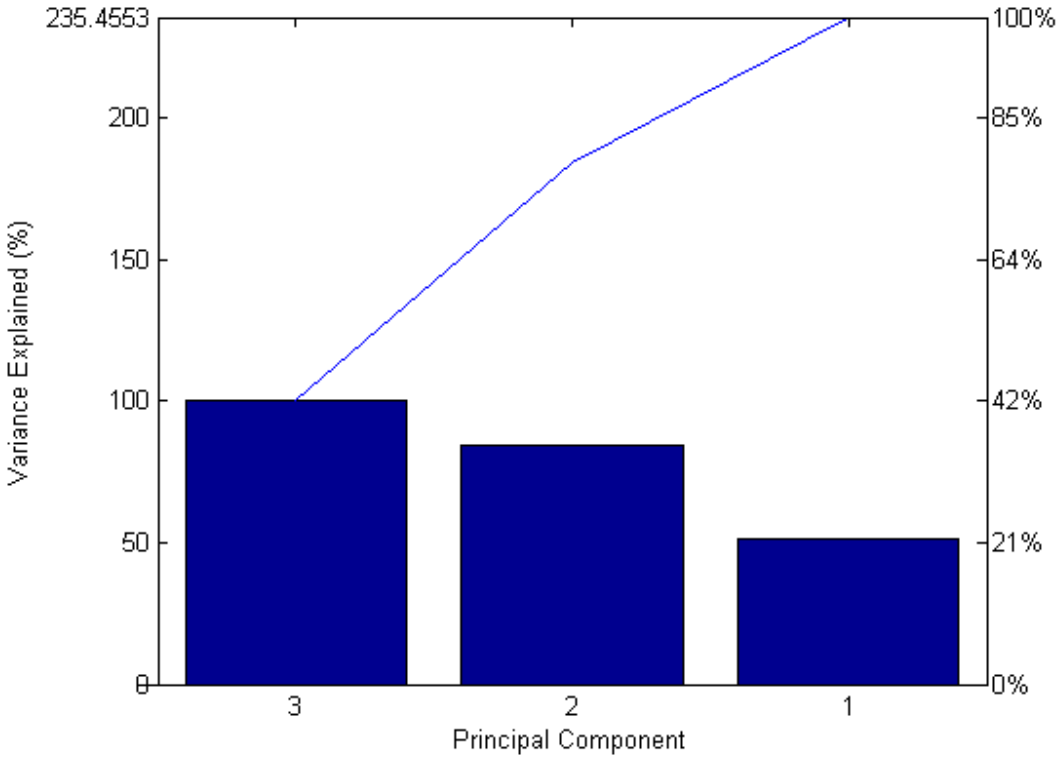


Figure 7 - Proportion de la variance expliquée par Composante Principale

Les valeurs précises sont :
51.2742 %
84.1811 %
100.0000%

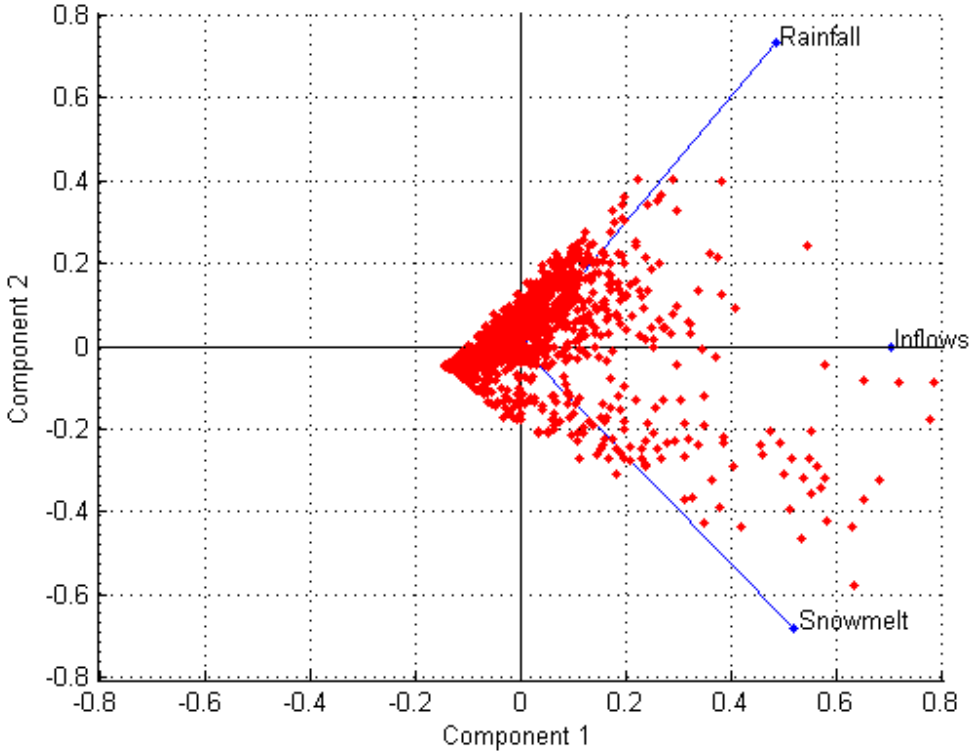


Figure 8 - Projection des données sur les 2 premières Composantes Principales

Ces résultats nous montrent qu'il existe aucune combinaison linéaire de deux variables permettant d'en déduire très bien la troisième, même s'il est vrai que la troisième composante principale n'explique qu'à peine plus de 15 % (= 100.000 - 84.181) de la variance. Néanmoins, comme la dimension est déjà très faible, nous garderons les trois variables inflow, rainfall et snowmelt pour le reste de notre étude. Nous pouvons également remarquer que la variable inflow correspond à la première composante principale, ce qui est un fait notable et qui sera important par la suite.

3. CLASSIFICATION

A présent que nous avons étudié les principales caractéristiques des trois variables de notre série statistique, nous allons voir s'il est possible d'établir une classification afin d'en extraire les tendances générales.

3.1. Classification hiérarchique

La classification hiérarchique est un algorithme simple publié en 1967 par S.C. Johnson. Il effectue une classification en établissant une hiérarchie entre les différents groupes en formant progressivement un dendrogramme : les racines des arbres représente la classe contenant l'ensemble des observations et les feuilles représentent les différents individus de la série statistique. Lorsque l'arbre est entièrement construit, il faut alors décider du nombre de classes que nous gardons. Cette décision se base sur la distance calculée entre les différentes classes en fonction de la métrique utilisée.

Nous allons utiliser ici la fonction `clusterdata` disponible nativement sous MATLAB, que nous appelons dans le script `mainKmeans.m`. Les variables seront centrées et réduites, et les vecteurs ainsi obtenus normalisées, afin d'obtenir des résultats non biaisés par les différences d'ordre de grandeur constatées dans le chapitre précédent, d'où la forme sphérique des données sur les graphiques.

Les résultats obtenus pour 12, 48 et 96 classes se trouvent sur les deux pages suivantes.

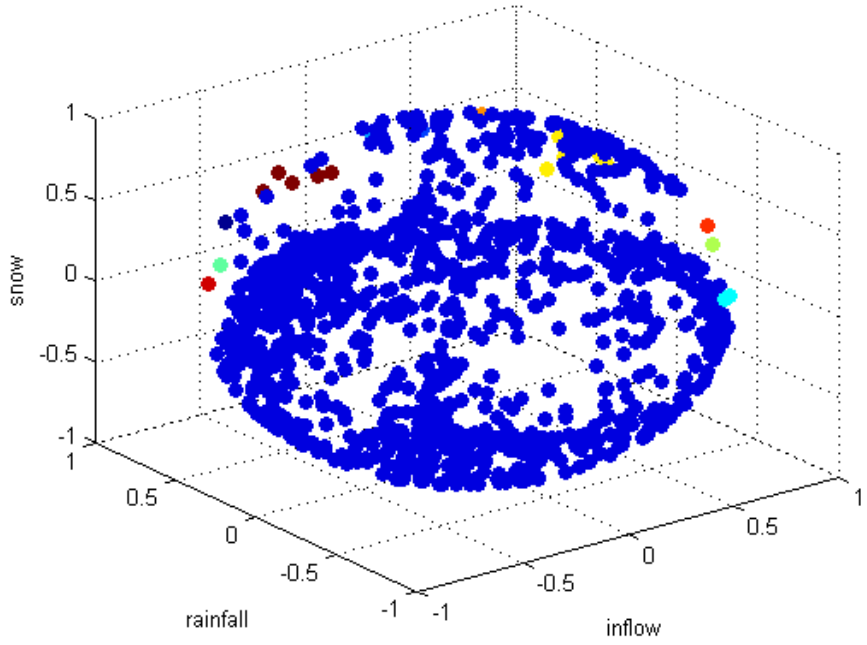


Figure 9 - Classification hiérarchique avec 12 classes

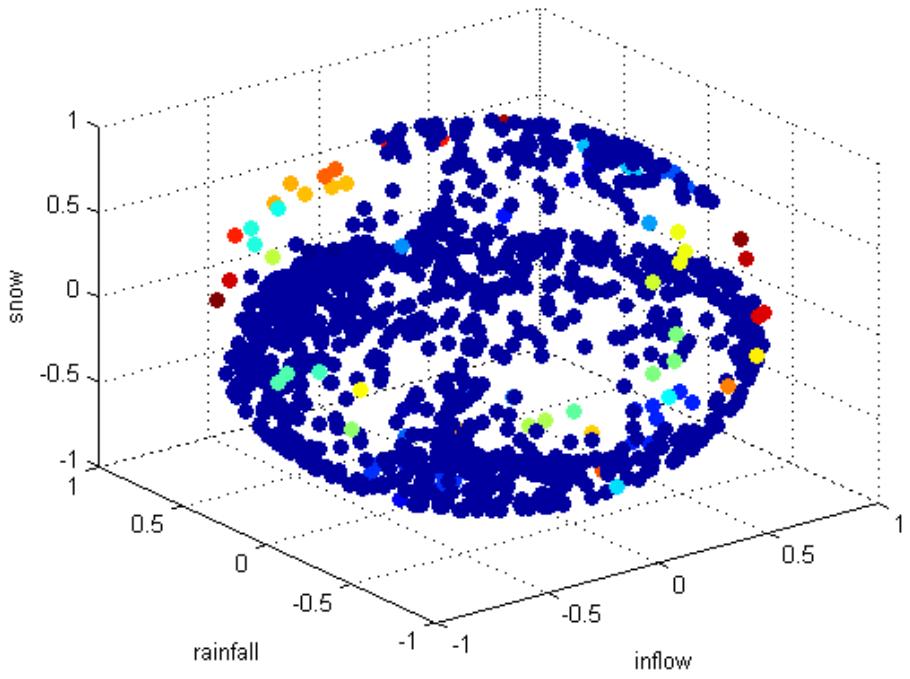


Figure 10 - Classification hiérarchique avec 48 classes

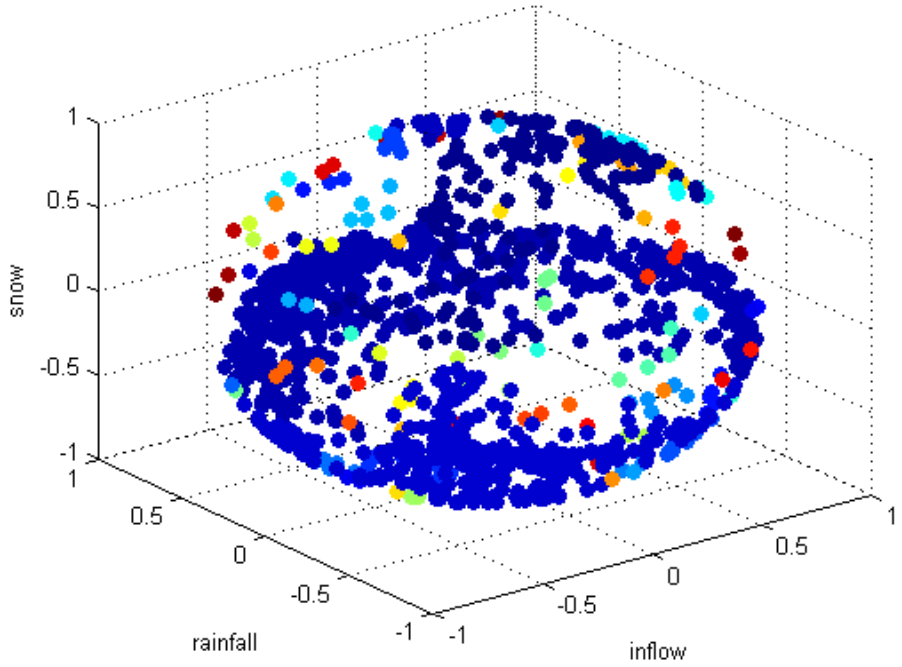


Figure 11 - Classification hiérarchique avec 96 classes

Nous voyons que la classification hiérarchique ne donne pas de bons résultats. Dans les cas avec 12 et 48 classes, nous constatons que la plus grande classe regroupent à elle seule la très grande majorité des observations. Dans le cas avec 96 classes, ce sont les 2 plus grandes classes qui sont excessivement grandes par rapport aux 94 autres classes.

Ces mauvais résultats s'expliquent que par la très grande proximité entre la majorité des vecteurs, tandis qu'il existe également des points isolés. Or, la fonction `clusterdata` appelle à son tour la fonction `linkage`, qui utilise par défaut le regroupement par la plus courte distance (*Single-linkage clustering*), méthode favorisant le regroupement « à la chaîne » entre les points situés à proximité, au détriment des points éloignés.

En outre, la fonction `clusterdata` appelle aussi la fonction `plist`, qui utilise par défaut la distance euclidienne pour calculer la distance entre deux classes. Choisir la distance de Mahalanobis donne en général de meilleurs résultats.

Voici ce que l'on obtient en utilisant le regroupement de Ward ainsi que la distance de Mahalanobis, via la commande

```
T = clusterdata(D,'maxclust',12 , 'distance', 'mahalanobis',
'linkage', 'ward'); :
```

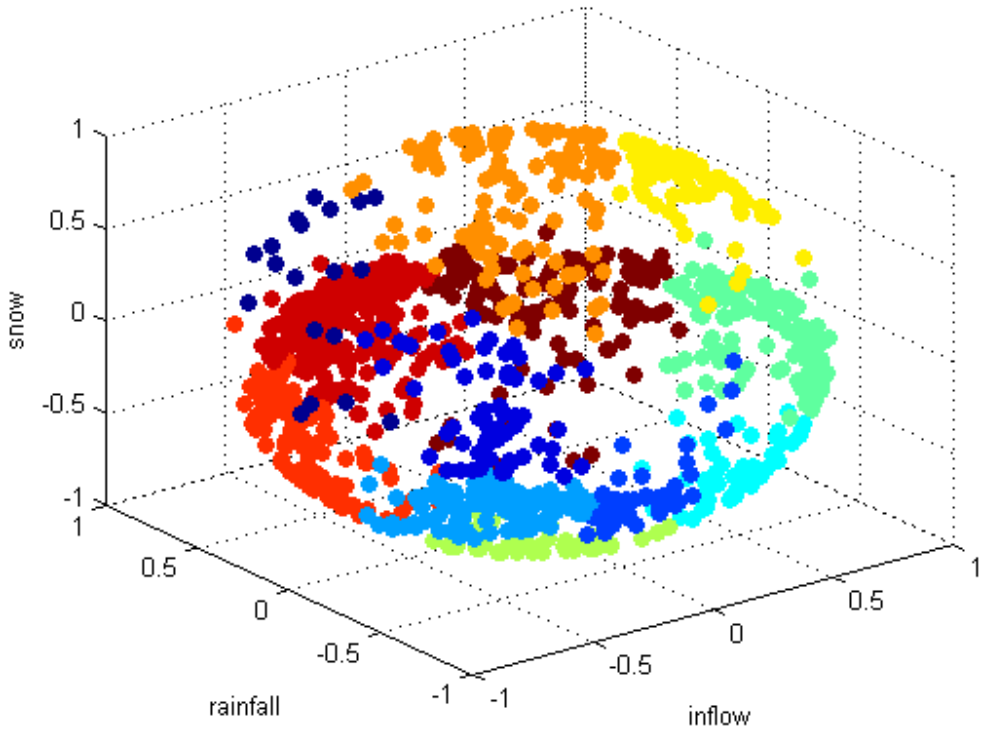


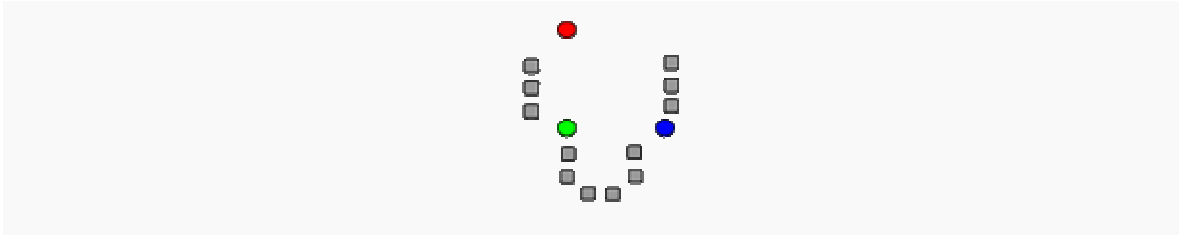
Figure 12 - Classification hiérarchique avec 12 classes, Ward et Mahalanobis.

Les résultats sont bien meilleurs : les classes obtenues sont de taille à peu près égale et il semble raisonnable de penser que chaque classe correspond environ à un mois.

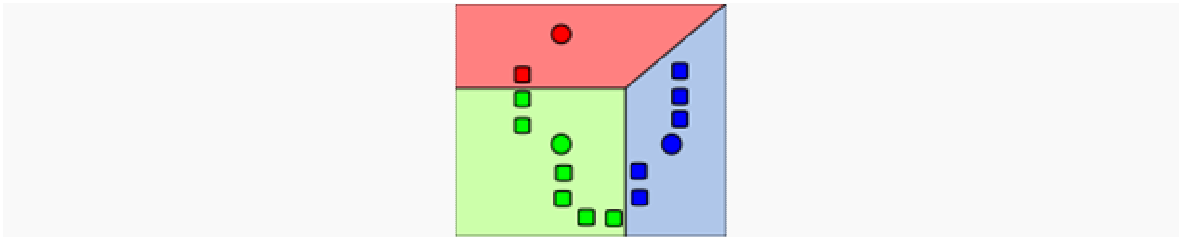
3.2.Méthode des K-moyennes

La méthode des K-moyennes est également un algorithme simple mais efficace publié en 1967 par MacQueen. Voici le principe :

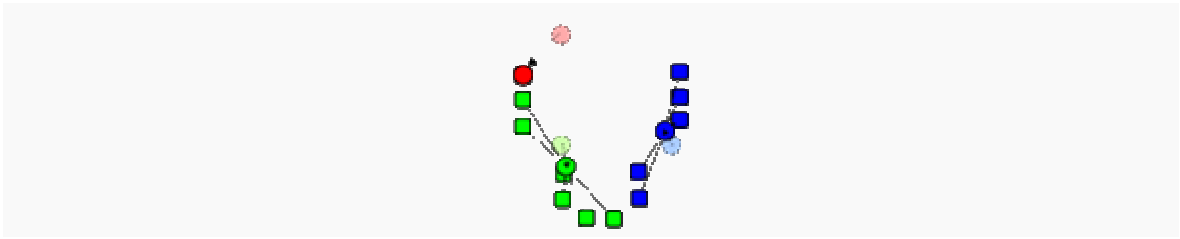
1) *k* centres initiaux sont sélectionnés aléatoirement parmi l'ensemble des données (ici k=3)



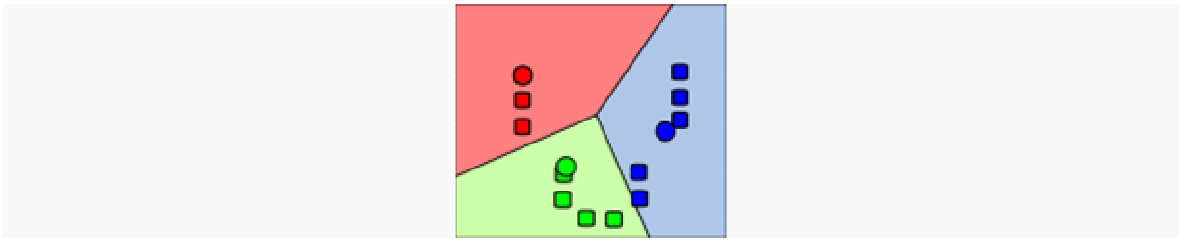
2) *k* groupes sont créés en associant chaque individu au centre le plus proche.



3) Les centroïdes de chaque groupe deviennent les nouveaux centres.



4) Les étapes 2 et 3 sont répétées aussi longtemps que les centres ne sont pas stabilisés.



Cette méthode donne ici de bons résultats, comparables à ceux obtenus avec la classification hiérarchique en utilisant le regroupement de Ward associé à la distance de Mahalanobis. Voici ce qu'on obtient en utilisant la fonction `clusterdata` disponible nativement sous MATLAB, que nous appelons dans le script `mainKmeans.m` :

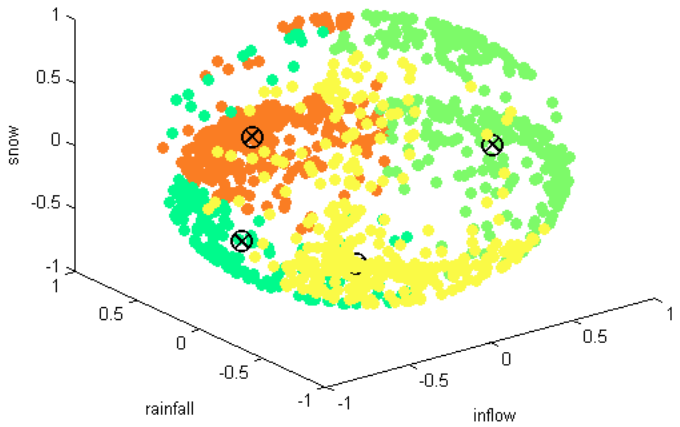


Figure 13 - Méthode des K-moyennes avec 4 classes

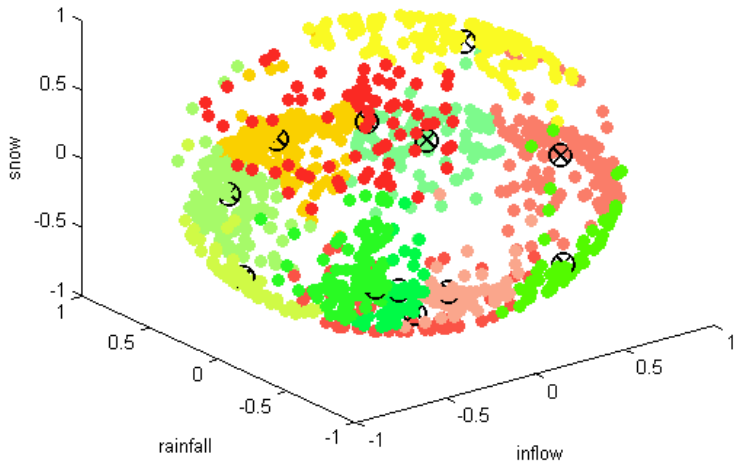


Figure 14 - Méthode des K-moyennes avec 12 classes

3.3.Cartes auto organisatrices (Kohonen)

Nous allons à présent essayer de voir à quel point il est possible de qualifier les différents mois en fonction des trois variables que sont le débit entrant, la fonte de neige et la pluviométrie. Pour cela, nous allons utiliser les cartes auto organisatrices dites cartes de Kohonen, en utilisant la SOM toolbox se trouvant sur http://www.cis.hut.fi/somtoolbox/ , permettant d'établir des cartes auto organisatrices, via le script `mainKohonen.m` .

L'étude se fera cette fois-ci en deux dimensions, par souci de clarté. Nous allons donc construire trois graphes. Les données utilisées sont centrées réduites, et chaque vecteur a été normalisé, d'où la forme en cercle sur les deux graphiques suivants où l'on a mis le débit entrant en abscisse et la pluviométrie en ordonnée.

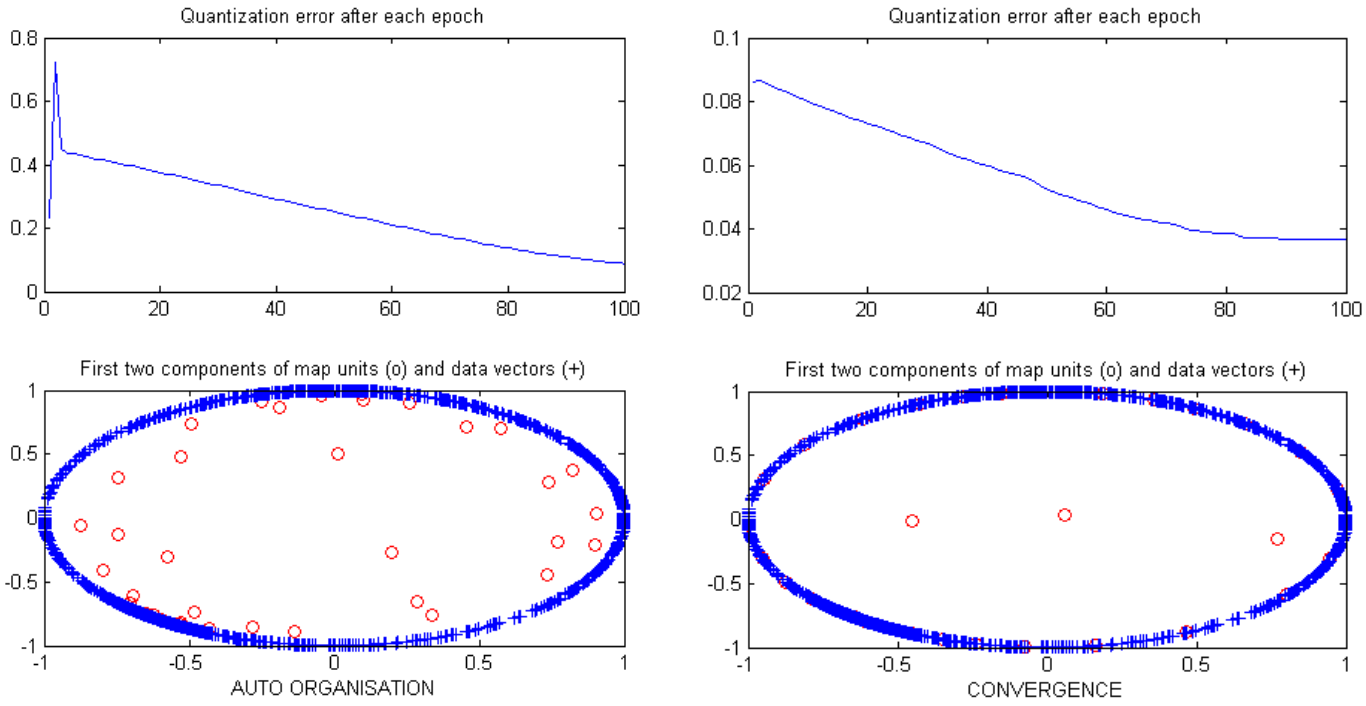


Figure 15 - Apprentissage de la carte auto organisatrice - X is inflow, Y is rainfall

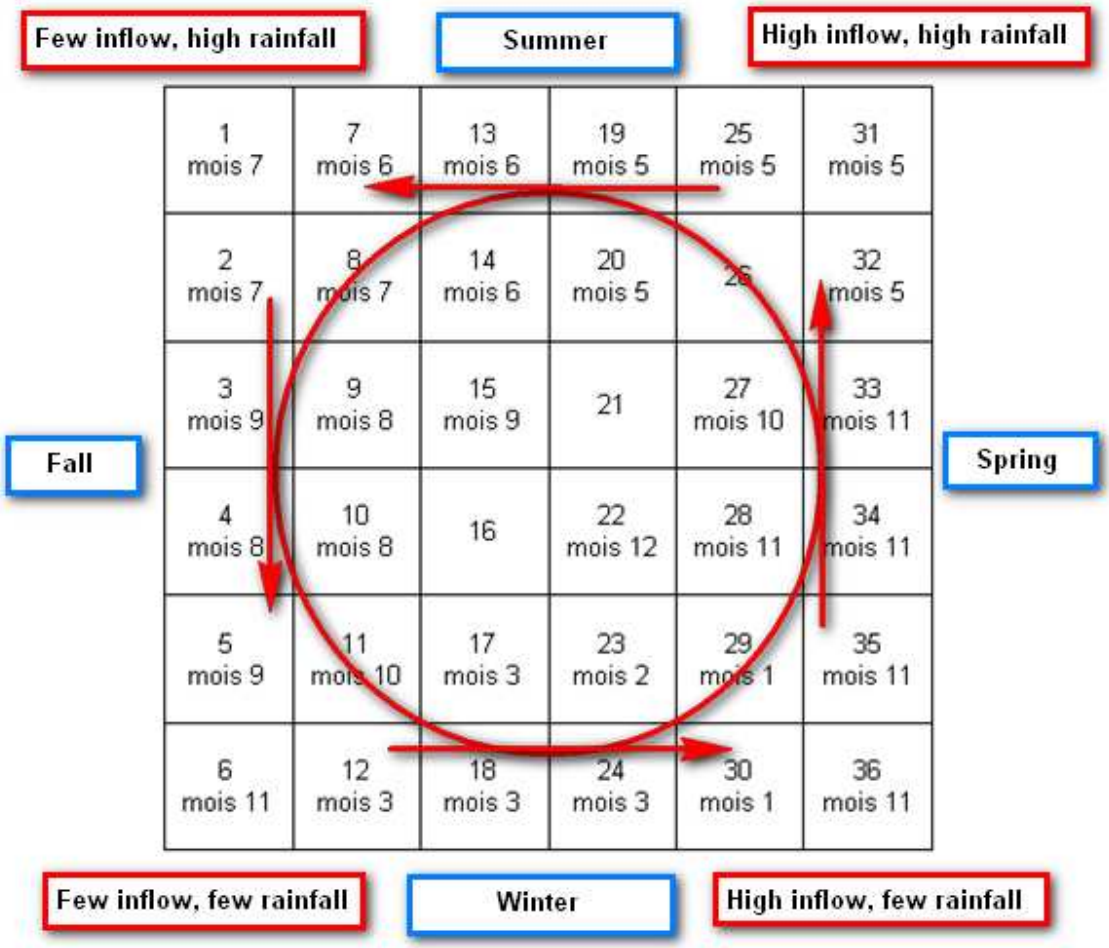


Figure 16 - Carte auto organisatrice - X is inflow, Y is rainfall

Nous voyons que la carte obtenue corrobore le graphe des variations moyennes constatées sur une année pendant la période étudiée vue dans la partie de ce rapport étudiant l'autocorrélation. Les annotations que nous avons mises en rouge et en bleu montrent une tendance observée pour un cycle annuel. Cependant, certains mois sont beaucoup plus représentés que d'autres, à l'instar du mois 11 (novembre) qui apparaît six fois, alors que le mois 2 et 12 n'apparaissent chacun qu'une seule fois. La carte se lit ainsi : si une mesure se situe en haut à gauche de cette carte, cela signifie qu'il y a de fortes chances qu'elle ait été effectuée en juin ou juillet.

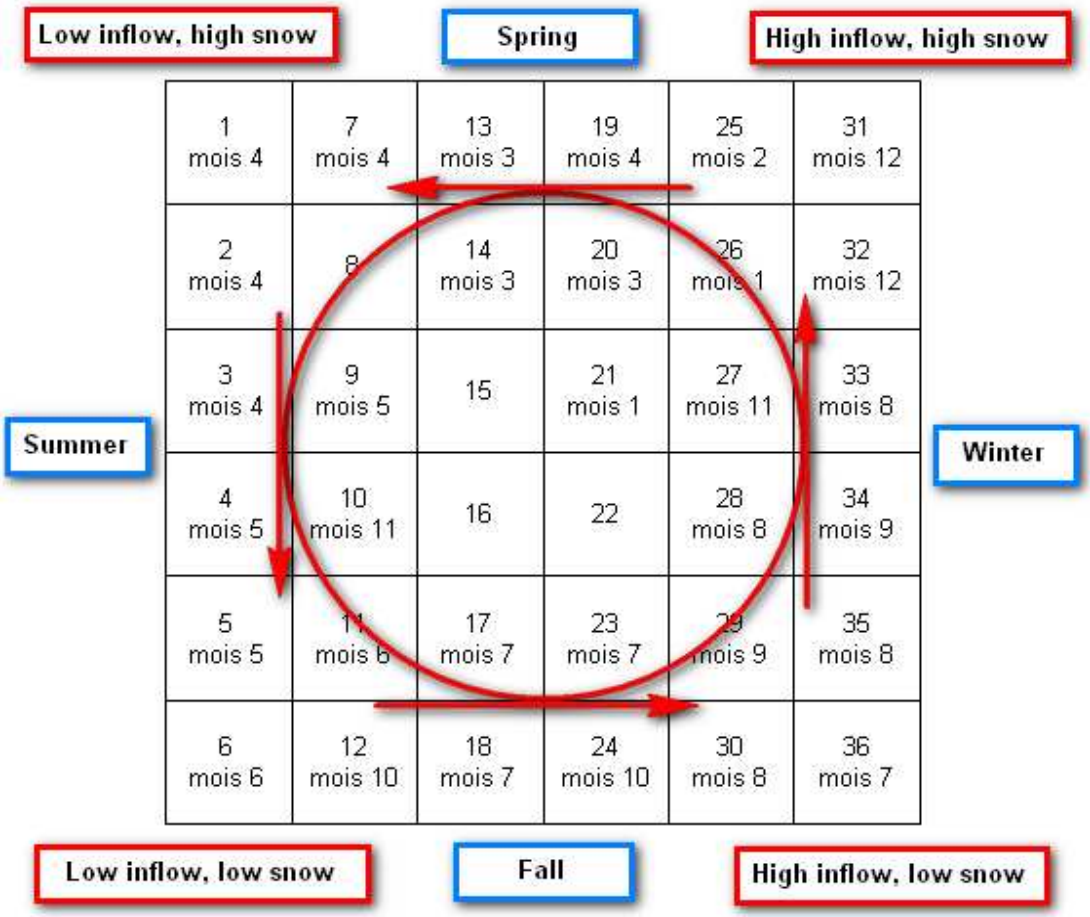


Figure 17 - Carte auto organisatrice - X is inflow, Y is snowmelt

Ici encore, nous pouvons voir un cycle annuel. Cependant, nous remarquons également que certaines saisons ne présentent pas exactement les mêmes propriétés que sur le cycle précédent. Par exemple, nous constations précédemment que l'automne était une période où le débit entrant était moyen, tandis que sur cette nouvelle carte nous pourrions en déduire que l'automne est une période où le débit entrant est moyen. Une telle interprétation est en fait l'erronée car elle ne tient pas en compte le fait que tous les vecteurs ont été normalisés, et que les données ont été préalablement centrées es réduites : sur cette carte, les mois situés en haut à gauche sont caractérisés par une fonte

des neiges bien plus importants que le débit entrant, ce qui ne signifie pas forcément que le débit entrant soit faible dans l'absolu.

Voici à présent la troisième et la dernière carte où se trouvent en abscisse la fonte des neiges et en ordonnée la pluviométrie :

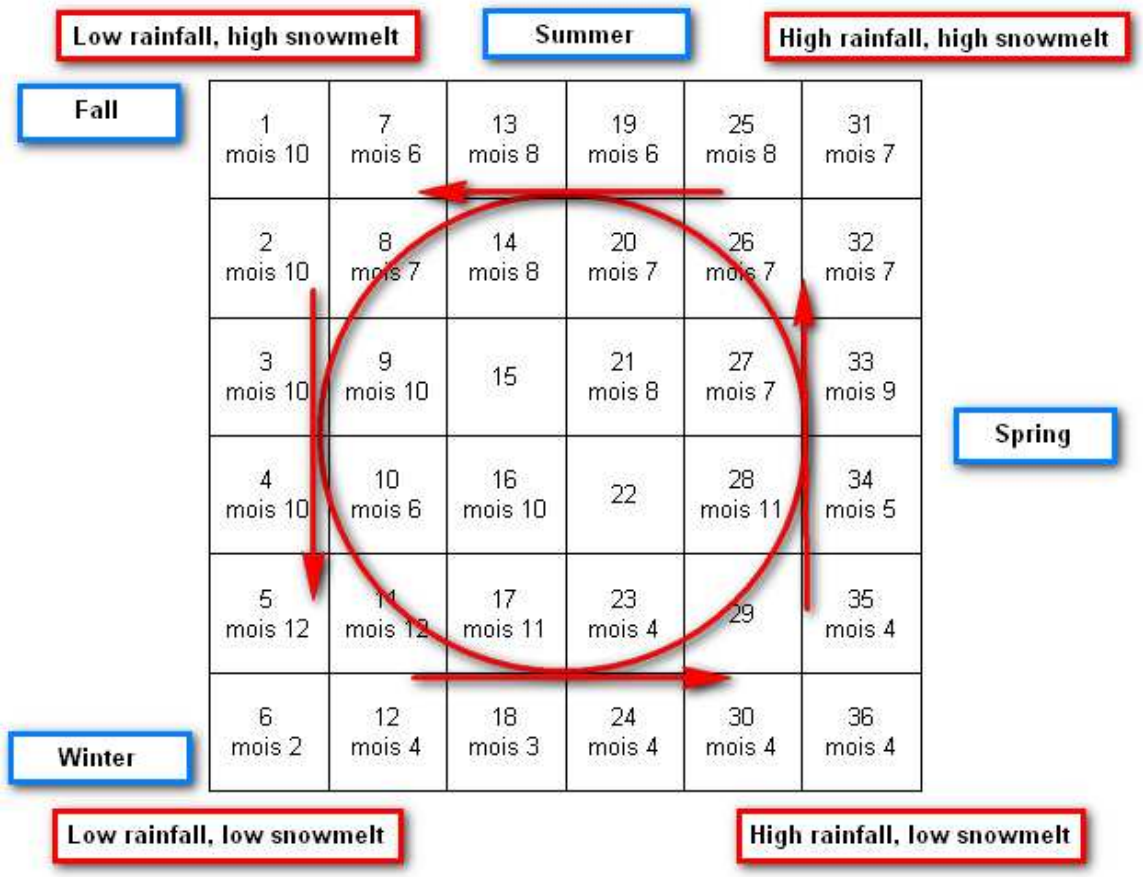


Figure 18 - Carte auto organisatrice - X is rainfall, Y is snowmelt

Comme dans les deux graphes précédents, nous constatons dans cette carte un cycle annuel. Nous retrouvons certaines caractéristiques que nous avions vu précédemment en construisant le graphique de la pluviométrie et la fonte des neiges moyennes sur un an : par exemple, la fonte des neiges et la pluviométrie sont toutes les deux également faibles en hiver.

4. CONSTRUCTION D'UN MODELE DE PREDICTION

A présent que nous nous sommes familiarisés avec les données à étudier et que nous avons regardé dans quelle mesure il était possible de les classifier et quels étaient les tendances générales, nous allons dans ce chapitre essayer de construire un modèle pouvant prédire le débit entrant du lac.

4.1. Autocorrélation

La façon la plus intuitive de prédire le débit entrant dans le lac est d'utiliser le caractère cyclique des saisons. L'idée est donc de simplement prédire pour l'instant t le débit entrant constaté 1 an auparavant.

Pour tester la valeur de cette prédiction, nous allons calculer l'autocorrélation pour le débit entrant, à l'aide de la TSA toolbox disponible sur le site http://biosig-consulting.com/matlab/tsa/ [8]. L'autocorrélation correspond à la corrélation croisée d'une série statistique par elle-même. Elle permet ainsi de détecter des régularités, des profils répétés dans une série statistique, même si elle contient beaucoup de bruit

Comme dans notre série statistique chaque année comprend 48 mesures, toute périodicité annuelle se remarquerait donc par une forte corrélation lorsque le décalage testé (appelé *lag* en anglais) est un multiple de 48.

Voici les résultats obtenus :

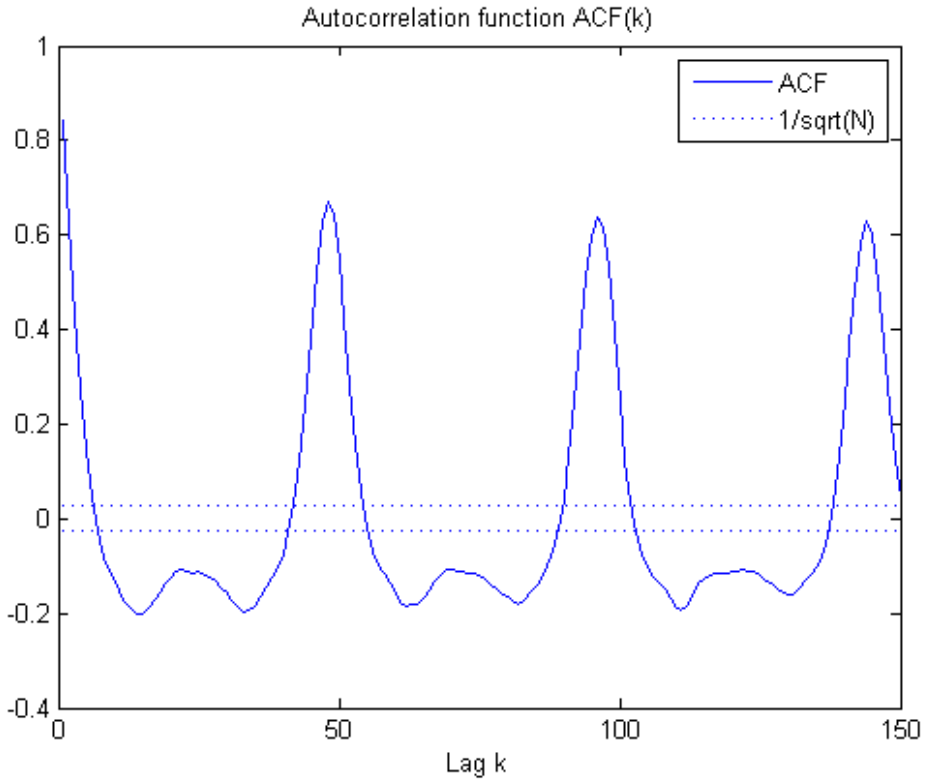


Figure 19 - Autocorrélation du débit entrant

Nous voyons que l'autocorrélation est légèrement supérieure à 0,6 lorsque le décalage est un multiple de 48, soit un an. Cela indique qu'une telle prédiction est d'une qualité médiocre.

Nous allons à présent calculer l'autocorrélation pour la pluviométrie et la fonte des neiges. Les résultats sont sur la page suivante. Il est intéressant de noter la différence entre les deux courbes : elle suggère que les variations de pluviométrie lors des différents relevés sont plus « douces » que celles de la fonte des neiges. Cela s'explique par la médiane de la fonte des neiges qui est égale à zéro et très éloignée de la moyenne (6.972892e-001)

comme nous l'avons vu précédemment, contrairement à la pluviométrie dont la médiane (1.085000e+000) en est beaucoup plus proche (la moyenne est de 1.767333e+000).

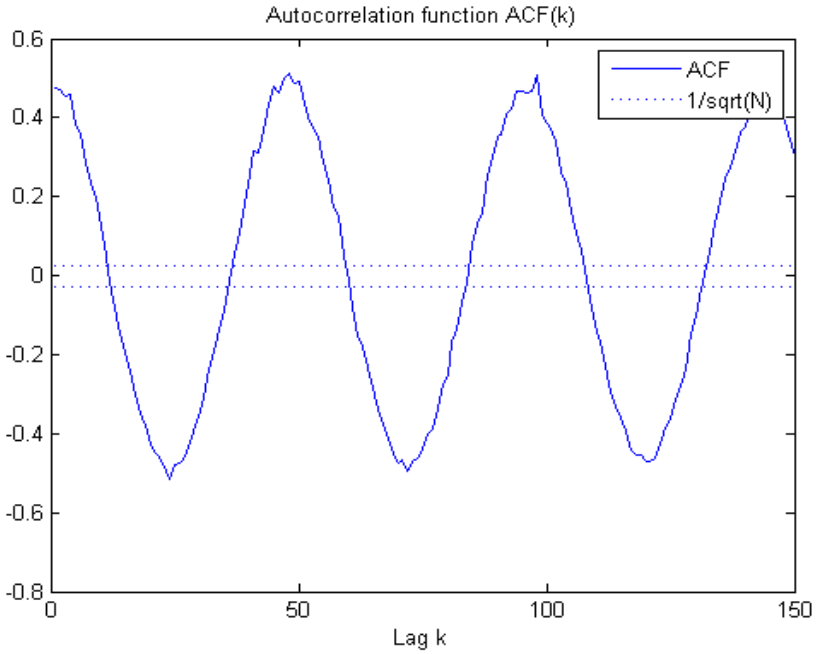


Figure 20 - Autocorrélation de la pluviométrie

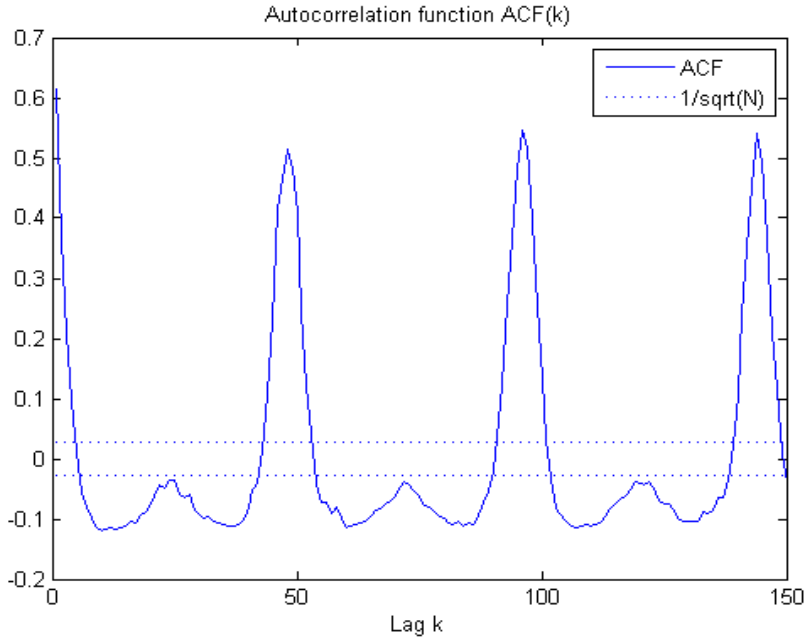


Figure 21 - Autocorrélation de la fonte des neiges

Ces résultats d'autocorrélation peuvent être rapprochés avec les box plots vus dans le premier chapitre.

Pour calculer plus précisément la qualité d'une telle prédiction basée sur l'autocorrélation et pour le comparer aux autres modèles de ce rapport, nous utiliserons ici comme mesure de qualité du modèle l'*Average Relative Variance* (ARV), qui établit le rapport entre l'erreur quadratique moyenne du modèle et la variance des données :

$$arv(D) = \frac{\frac{1}{|D|}\sum_{i \in D}(y^i - f(x^i))^2}{\frac{1}{|S|}\sum_{i \in S}(y^i - \mu_S)^2}$$

où

- D est l'ensemble des données test ;
- S et l'ensemble de la série (données d'apprentissage + données de validation + données test) ;
- x^i est la valeur en entrée de la série à l'instant i ;
- y^i est la valeur en sortie de la série à l'instant i ;
- $f(x^i)$ est la sortie du réseau à l'instant i (dont l'objectif est d'être le plus proche de y^i) ;
- μ_S la moyenne de la valeur désirée dans S.

Plus l'ARV est proche de 0, plus le modèle évalué est efficace. Si l'ARV est supérieur à 1, cela signifie que le modèle est moins efficace que prédire simplement en faisant la moyenne des valeurs passées, autrement dit que le modèle est inutile.

En calculant l'ARV en prédisant la valeur du débit entrant simplement en prenant celle constatée l'année précédente, nous obtenons **0.6109**, ce qui est un résultat tout à fait honorable.

Nous pouvons améliorer ce modèle en prenant la moyenne des années précédentes. Voici les variations moyennes constatées sur une année pendant la période étudiée (1952-1983), les tendances mensuelles sont claires :

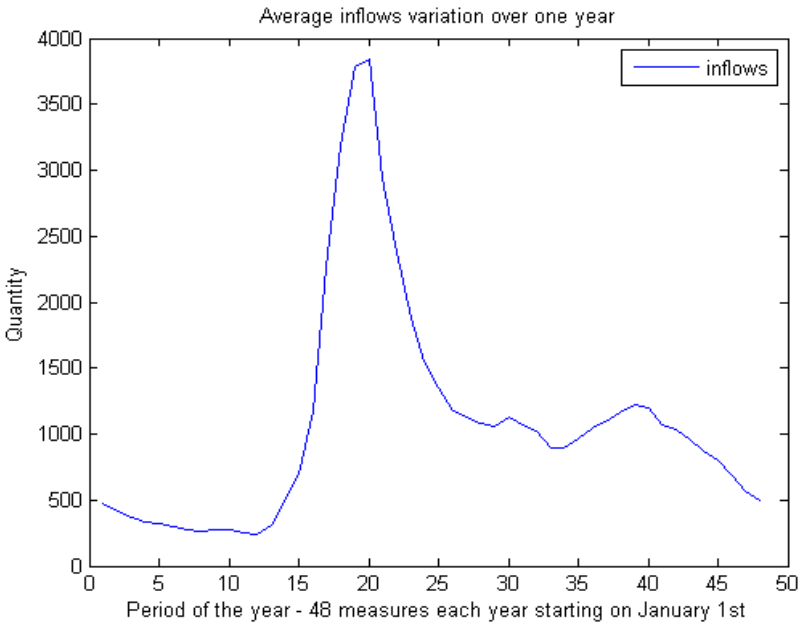


Figure 22 - Débit entrant moyen sur un an

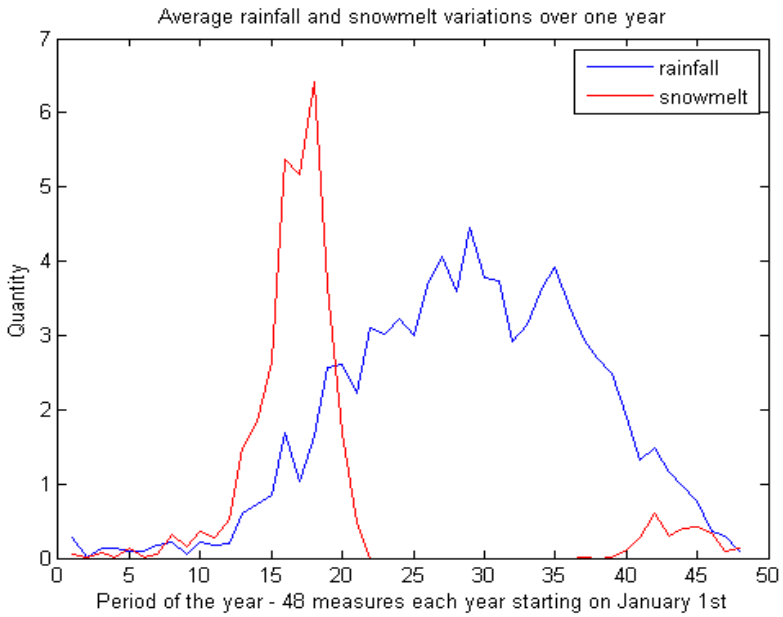


Figure 23 - Pluviométrie et fonte des neiges moyennes sur un an

Voici la courbe de la moyenne du débit entrant constaté sur les années précédant la valeur à prédire. Par exemple, la valeur de ce graphe correspondant à la première mesure de l'année 1960 correspond à la moyenne de la première mesure de chacune des années précédentes, en l'occurrence de 1953 à 1959.

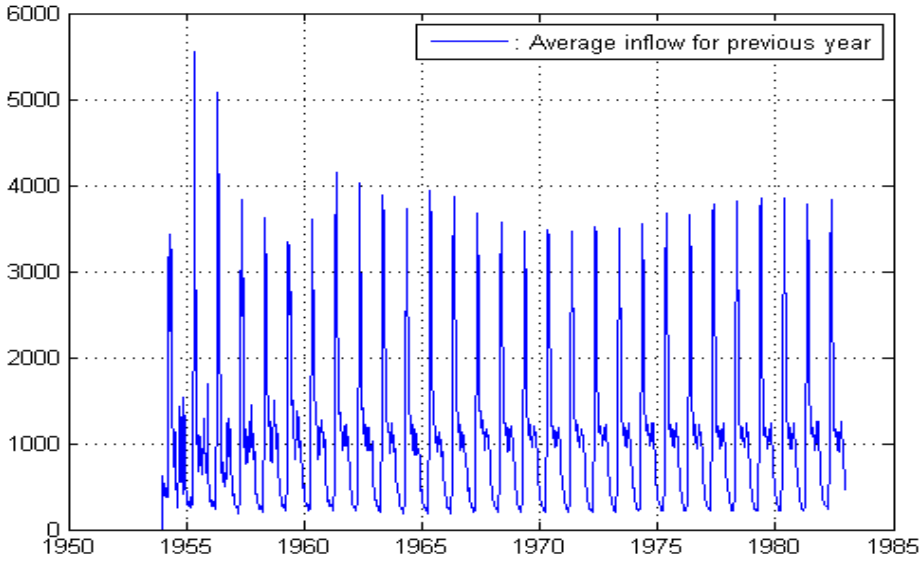


Figure 24 - Moyenne du débit entrant constatée sur les années précédentes

En prenant la moyenne des années précédentes constatée pour le débit entrant, nous obtenons un ARV de **0.3551**, ce qui est nettement mieux que le résultat obtenu précédemment (0.6109).

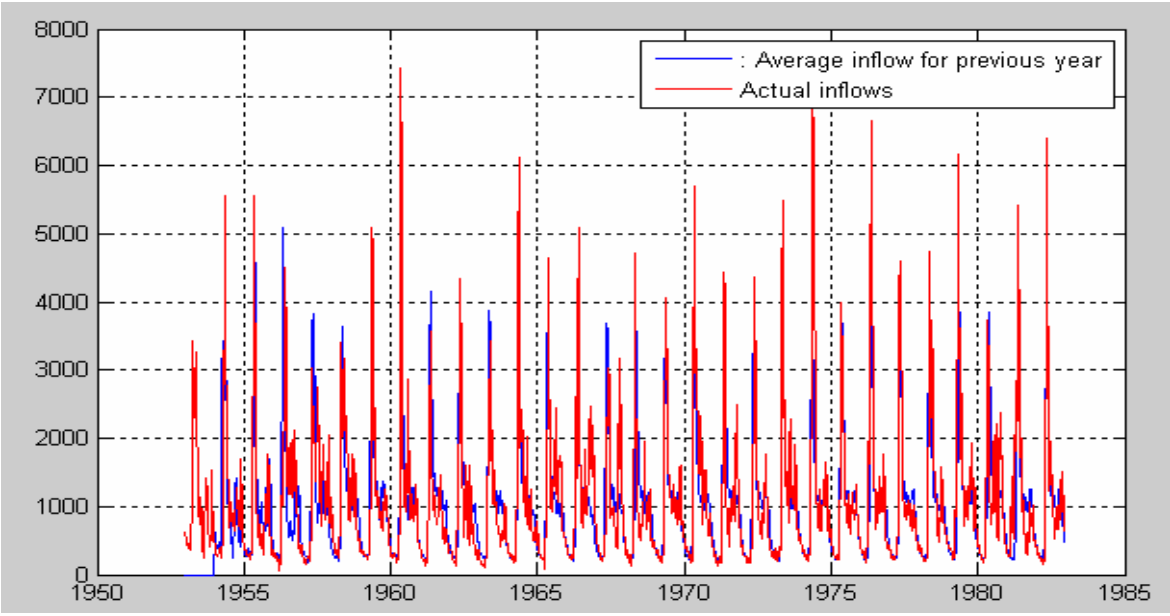


Figure 25 - Moyenne des débits passés vs débit constaté

4.2.Perceptron multicouches (MLP)

4.2.1. Définitions et structure

Un perceptron multicouches est un réseau de neurones sans cycle. Nous présentons à la couche d'entrée (*input layer*) un vecteur et le réseau nous retourne un vecteur résultat dans la couche de sortie (*output layer*). Entre ces deux couches, les éléments du vecteur d'entrée sont pondérés par le poids des connexions et mélangés dans les neurones cachés qui se trouvent dans les couches cachées (*hidden layer*).

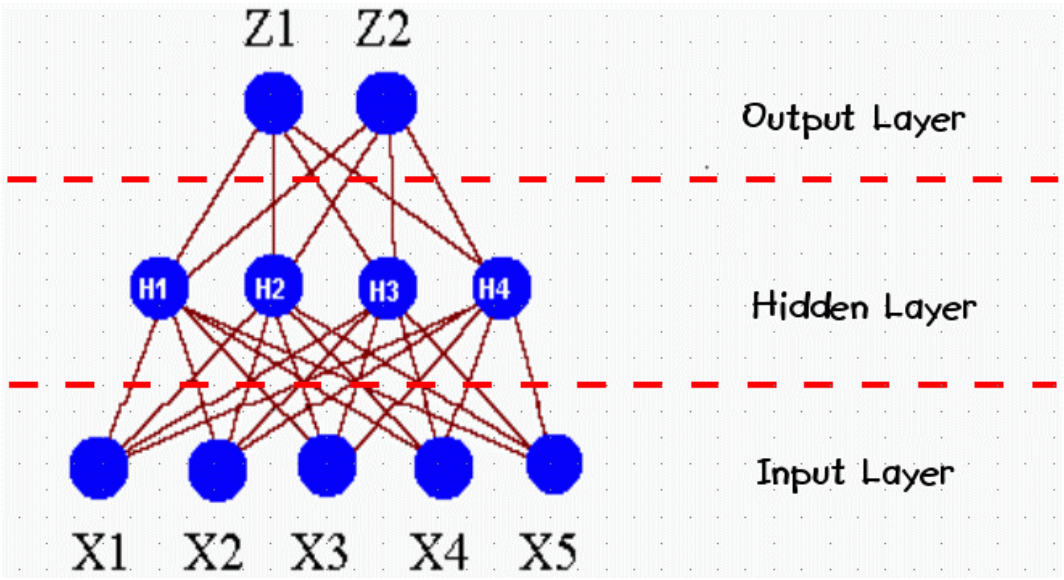


Figure 26 - Exemple d'un réseau de neurones feedforward

Plusieurs fonctions d'activation pour la couche de sortie sont couramment utilisées, telles les fonctions linéaires, logistique ou encore softmax. De même, il existe plusieurs algorithmes de rétropropagation des erreurs permettant d'optimiser l'apprentissage des poids à partir des erreurs faites entre les valeurs calculées par le réseau et les valeurs

réelles : méthode du gradient conjugué (Conjugate gradients optimization), Scaled Conjugate Gradient, Quasi-Newton optimization, etc.

L'ensemble des données à étudier sera partitionné en trois sous-ensembles de données. Ces trois sous-ensembles doivent être distincts afin d'assurer le bon apprentissage du réseau et son évaluation non biaisée.

1. L'*ensemble d'apprentissage*, utilisé dans la phase d'apprentissage afin de régler les poids du réseau pour que les sorties du réseau se rapprochent le plus possible des valeurs attendues ;
2. L'*ensemble de validation*, permettant de déterminer l'arrêt de la phase d'apprentissage afin d'éviter tout sur-apprentissage qui nuirait à l'application du modèle pour d'autres données ;
3. L'*ensemble de test*, que nous utiliserons pour évaluer les performances du réseau.

En général, ces trois sous-ensembles sont choisis de telle sorte à ce que leur cardinalité soit à peu près égal. Nous étudierons plus loin l'impact de leur choix de taille.

4.2.2. Analyse de l'impact du prétraitement des données

Nous allons vérifier si centrer et réduire les données initiales nous donnent de meilleurs résultats.

Tout d'abord, regardons les résultats obtenus lorsque les données ne sont pas traitées. La variable `Predict_size` indique le nombre de résultats que nous allons prendre en compte. Par exemple, si `Predict_size = 6`, cela signifie que le réseau prend en entrée les valeurs de pluviométrie, fonte des neiges et débit entrant lors des six dernières mesures, lesquelles sont prises tous les quarts de mois. `Predict_size = 6`

Signifie donc que l'on prend en compte seulement le mois et demi écoulé précédent la valeur de débit entrant que l'on veut prédire.

Voici les résultats obtenus en utilisant la toolbox Netlab [7] :

```
=======================================
 === ARV table with Predict_size = 6 ===
=======================================
 ARV with 3 hidden neurons = 1.272048e+000
 ARV with 6 hidden neurons = 9.575360e-001
ARV with 12 hidden neurons = 1.102687e+000
ARV with 24 hidden neurons = 8.796483e-001
ARV with 48 hidden neurons = 8.795889e-001
ARV with 96 hidden neurons = 4.346010e-001
ARV with 144 hidden neurons = 1.082678e+000

=======================================
 === ARV table with Predict_size = 12 ===
=======================================
 ARV with 3 hidden neurons = 1.045543e+000
 ARV with 6 hidden neurons = 2.462280e+000
ARV with 12 hidden neurons = 1.086966e+000
ARV with 24 hidden neurons = 9.555312e-001
ARV with 48 hidden neurons = 9.817822e-001
ARV with 96 hidden neurons = 8.559246e-001
ARV with 144 hidden neurons = 5.900990e-001

=======================================
 === ARV table with Predict_size = 48 ===
=======================================
 ARV with 3 hidden neurons = 8.335742e-001
 ARV with 6 hidden neurons = 8.136201e-001
ARV with 12 hidden neurons = 8.420245e-001
ARV with 24 hidden neurons = 1.971012e+000
ARV with 48 hidden neurons = 6.617589e-001
ARV with 96 hidden neurons = 7.596775e-001
ARV with 144 hidden neurons = 1.972292e+000
```

Nous voyons que la plupart des ARV sont supérieurs à 1, ce qui signifie que le modèle n'est pas efficace, comme nous le confirme le graphique :

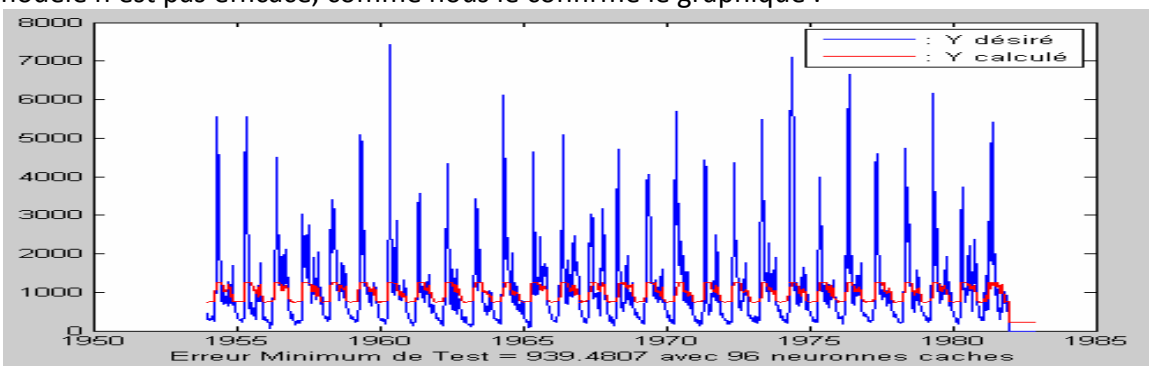


Figure 27 - Prédiction du MLP avec données non traitées

A présent, essayons de diviser simplement toutes les valeurs de débit entrant dans le lac par 1000, comme nous le suggérions dans le premier chapitre. Nous obtenons des résultats bien meilleurs. Cette fois-ci, le modèle est efficace :

```
=======================================
 === ARV table with Predict_size = 6 ===
=======================================
ARV with 3 hidden neurons = 3.615301e-001
ARV with 6 hidden neurons = 3.648960e-001
ARV with 12 hidden neurons = 3.804038e-001
ARV with 24 hidden neurons = 3.671627e-001
ARV with 48 hidden neurons = 4.014003e-001
ARV with 96 hidden neurons = 4.229980e-001
ARV with 144 hidden neurons = 4.231062e-001
=======================================
 === ARV table with Predict_size = 12 ===
=======================================
ARV with 3 hidden neurons = 3.776689e-001
ARV with 6 hidden neurons = 4.684528e-001
ARV with 12 hidden neurons = 3.889060e-001
ARV with 24 hidden neurons = 3.734274e-001
ARV with 48 hidden neurons = 4.715908e-001
ARV with 96 hidden neurons = 4.472611e-001
ARV with 144 hidden neurons = 4.650412e-001
=======================================
 === ARV table with Predict_size = 48 ===
=======================================
ARV with 3 hidden neurons = 5.211806e-001
ARV with 6 hidden neurons = 3.768322e-001
ARV with 12 hidden neurons = 3.628320e-001
ARV with 24 hidden neurons = 4.588404e-001
ARV with 48 hidden neurons = 3.592023e-001
ARV with 96 hidden neurons = 4.267923e-001
ARV with 144 hidden neurons = 3.827214e-001
```

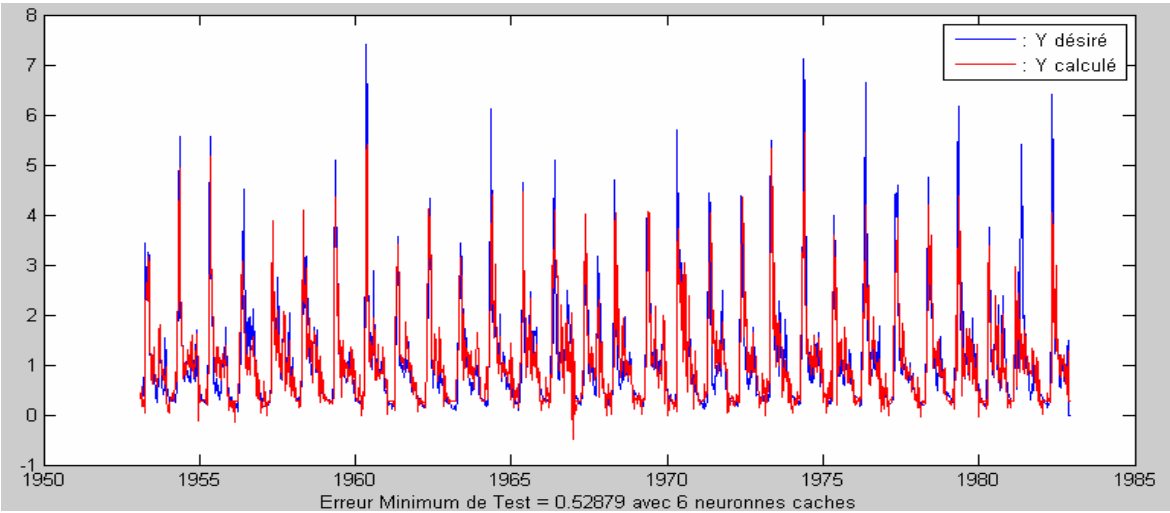


Figure 28 - Prédiction du MLP avec débit entrant divisé par 1000

Maintenant, essayons de centrer et réduire toutes nos données. Nous obtenons des résultats légèrement meilleurs à ceux obtenus précédemment, ce qui montre que notre traitement simple précédent était une bonne approximation :

```
=======================================
 === ARV table with Predict_size = 6 ===
=======================================
ARV with 3 hidden neurons = 2.533819e-001
ARV with 6 hidden neurons = 2.290497e-001
ARV with 12 hidden neurons = 2.496079e-001
ARV with 24 hidden neurons = 2.388564e-001
ARV with 48 hidden neurons = 2.346452e-001
ARV with 96 hidden neurons = 2.316126e-001
ARV with 144 hidden neurons = 2.313823e-001
=======================================
 === ARV table with Predict_size = 12 ===
=======================================

ARV with 3 hidden neurons = 2.715612e-001
ARV with 6 hidden neurons = 2.983055e-001
ARV with 12 hidden neurons = 2.745780e-001
ARV with 24 hidden neurons = 2.583667e-001
ARV with 48 hidden neurons = 2.638117e-001
ARV with 96 hidden neurons = 2.862301e-001
ARV with 144 hidden neurons = 2.409670e-001
=======================================
 === ARV table with Predict_size = 48 ===
=======================================
ARV with 3 hidden neurons = 3.283175e-001
ARV with 6 hidden neurons = 3.000058e-001
ARV with 12 hidden neurons = 2.950053e-001
ARV with 24 hidden neurons = 2.731107e-001
ARV with 48 hidden neurons = 2.565473e-001
ARV with 96 hidden neurons = 2.444401e-001
ARV with 144 hidden neurons = 2.698794e-001
```

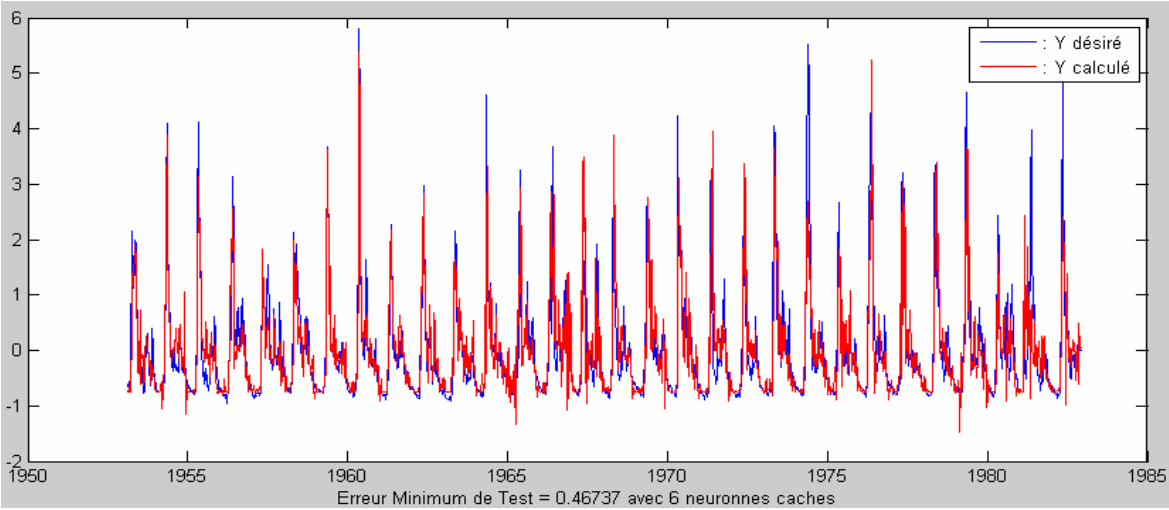


Figure 29 - Prédiction du MLP avec données centrées-réduites

En conclusion, il faut traiter les données pour obtenir des résultats utiles avec ce modèle ; la division du débit entrant par 1000 est une bonne approximation certes fortuite, cependant centrer et réduire toutes les données s'avère être le plus efficace.

4.2.3. Apprentissage

Afin d'éviter tout sur-apprentissage, nous utilisons la méthode d'*early stopping* pour décider de l'arrêt de l'apprentissage. Le principe est d'utiliser l'ensemble de validation pour le tester sur le réseau et voir ses résultats. Au début de l'apprentissage, le réseau performe de mieux en mieux sur l'ensemble de validation, puis passer un certains seuil les performances commencent à se dégrader : le réseau est alors en train de sur-apprendre, c'est-à-dire coller excessivement aux données d'apprentissage, il faut donc l'arrêter.

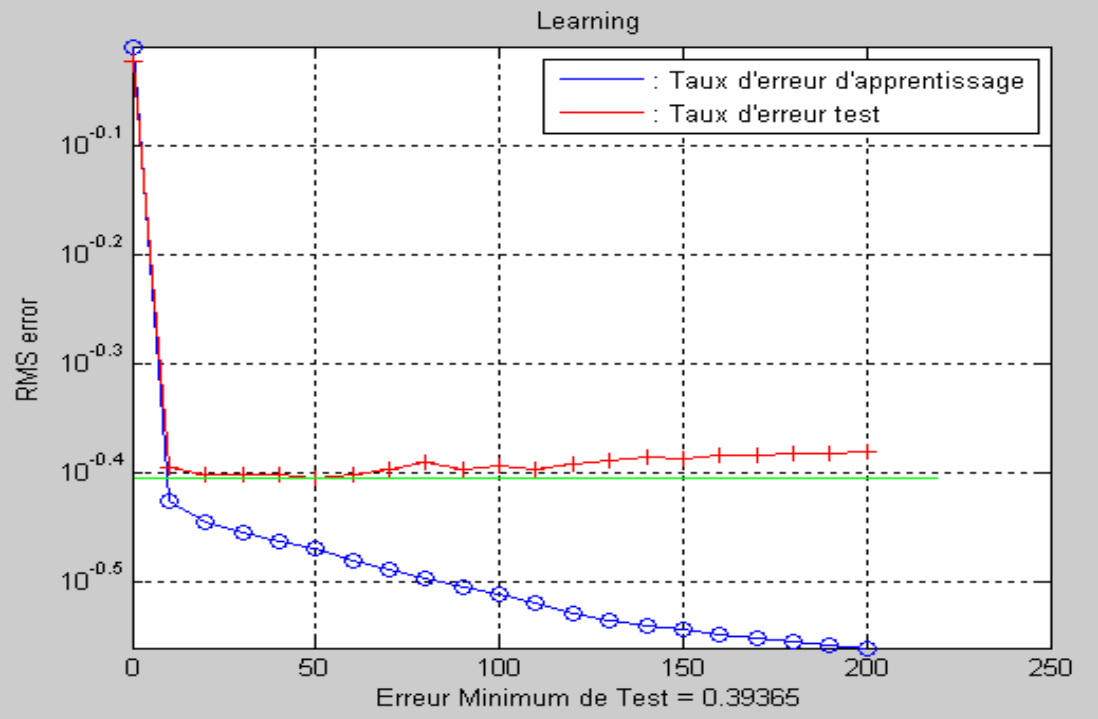


Figure 30 - Early stopping pour éviter le sur-apprentissage

4.2.4. Fonctions de sorties et fonctions d'activation

Nous avons choisi la fonction linéaire comme fonction de sortie, ainsi que l'algorithme SCG pour optimisation, car en testant sur les possibilités, les résultats étaient bien moins bons. Nous nous détaillerons pas ici les comparaisons des résultats par souci de concision.

4.2.5. Comparaison avec l'autocorrélation

Nous allons à présent comparer les résultats de ce modèle basé sur le perceptron multicouches par rapport aux résultats obtenus précédemment grâce à l'autocorrélation. Nous utiliserons pour cette comparaison la mesure de qualité ARV décrite précédemment. Afin que les résultats puissent être retrouvés aisément, nous allons seeder la fonction random servant à la construction initiale du réseau de neurones par 2 (code MATLAB : « `randn('state', 2);` » dans le script *mlpWrapper.m*)

Nous faisons varier le nombre de neurones d'entrée ainsi que le nombre de neurones cachés parmi ces deux ensembles :

```
nhiddenSet   = [1, 2, 3, 4, 5, 6, 8, 10, 12, 24, 48, 48*2, 48*3]
Predict_size = [1, 2, 3, 4, 5, 6, 8, 10, 12, 48, 2*48, 3*48, 4*48]
```

Le meilleur résultat trouvé est lorsque l'on prend `Predict_size = 1` et `nhidden = 48`. L'ARV obtenue est alors de **0.1819394** ce qui est bien meilleure que l'ARV trouvé précédemment avec l'autocorrélation (0.3551). L'ensemble des résultats est mis en annexe de ce document.

Nous constatons que :

ARV(MLP) / ARV(Autocorrélation) = 0.1819394 / 0.3551 ~= 0.512361,

ce que l'on pourrait rapprocher avec les résultats de l'Analyse en Composantes principales faite en première partie. En effet, le modèle bâti sur l'autocorrélation n'utilisait que la variable inflow, alors que le modèle construit sur le perceptron multicouches prend en compte toutes les trois variables. L'ACP avait mis en évidence que la première composante principale représentait 51.2742 % de la variance de notre série statistique. Or, nous avions également relevé que cette première composante principale correspondait à la variable débit entrant. Par conséquent, les mauvaises performances de l'autocorrélation par comparaison avec le perceptron multicouches s'expliqueraient essentiellement par le fait que l'autocorrélation ne prend pas en compte la fonte des neiges et la pluviométrie.

Cette explication est vérifiée lorsque nous utilisons le perceptron multicouches en prenant en compte seulement la variable débit entrant : le meilleur ARV obtenu dans ces conditions est de 0.2891588 lorsque l'on prend `Predict_size = 2` et `nhidden = 3`. Ce résultat permet d'affirmer en plus que nous ne disposions que de la variable débit entrant, le perceptron multicouches permettraient de faire de meilleure prédiction que le modèle issu de l'autocorrélation.

4.2.6. Etude des poids du réseau

Le perceptron multicouches optimal comportant 48 neurones dans sa couche cachée, il est difficile d'interpréter les poids un à un. Cependant, nous pouvons remarquer qu'il y a autant de neurones cachés que de mesures par année. Il serait toutefois difficile d'en conclure que chaque neurone représente une mesure en particulier, car comme nous pouvons le voir dans l'ensemble des résultats présents en annexe de ce document, nous nous croisons des perceptron multicouches présentant beaucoup moins de neurones mais dont la performance est pourtant proche du perceptron multicouches optimal trouvé.

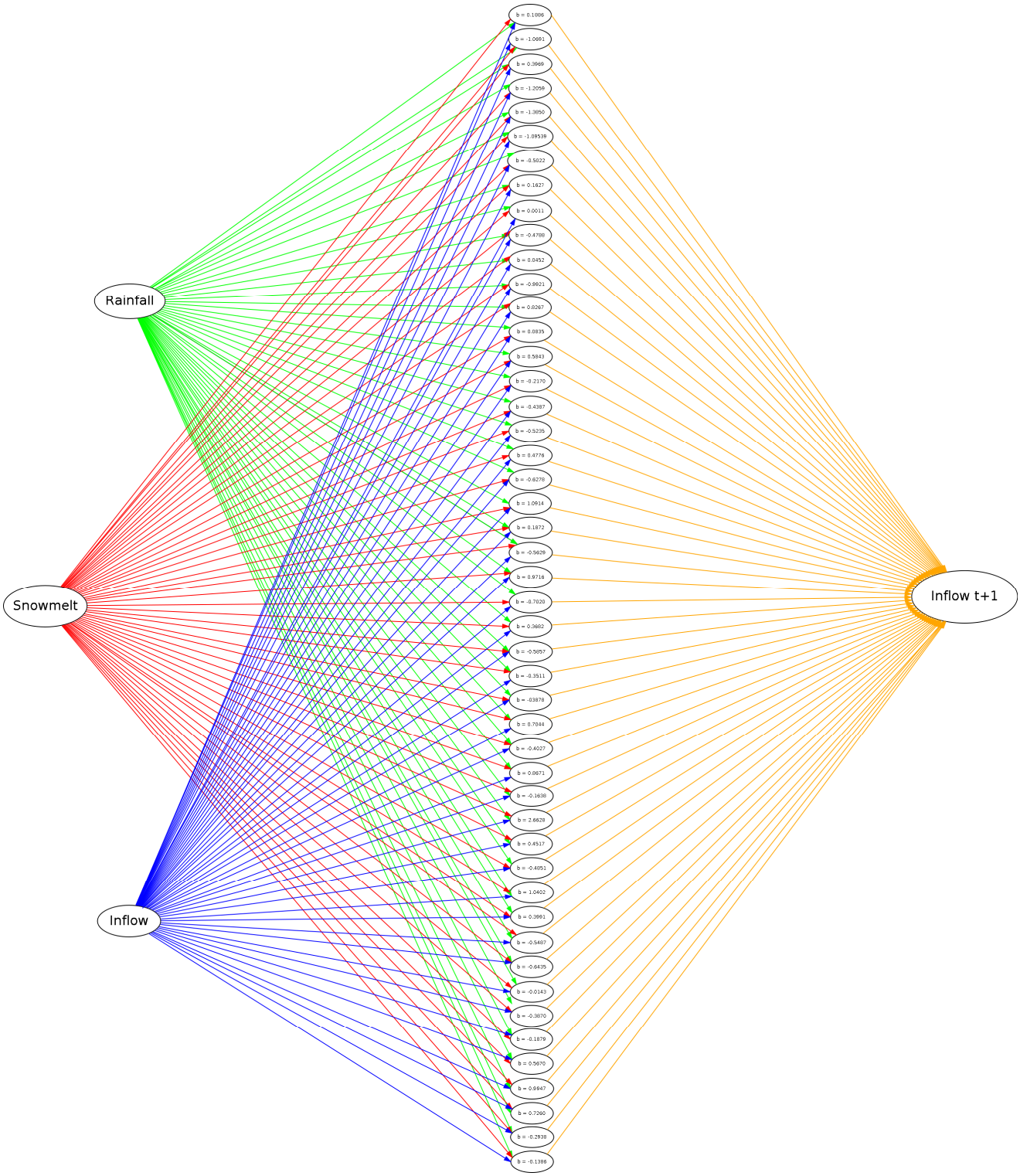


Figure 31 - MLP à 3 neurones en entrée, 48 cachés et 1 en sortie.

5. CONCLUSION

Après une étude sur la classification des données intéressantes basée sur plusieurs algorithmes connus, les modèles de prédiction se sont avérés efficaces, en particulier celui basé sur le perceptron multicouches. Nous avons obtenu avec ce derniers des résultats très intéressants que nous avons pu lier avec notre modèle issu de l'autocorrélation en utilisant l'analyse en composantes principales réalisées au chapitre 2 consacré à l'étude liminaire des données.

Il eût été intéressant d'introduire une seconde mesure de qualité autre que l'ARV, telle l'Erreur quadratique moyenne couramment utilisée pour évaluer des modèles de prédiction. Nous avions aussi voulu bâtir un modèle de prédiction basé sur la Support Vector Regression (SVR), une méthode apparue en 1996, ainsi qu'un modèle basé sur les chaînes de Markov cachée continues. Mais le temps a manqué et nous avons préféré prendre le temps d'étudier quelques modèles, plutôt que de risquer à les multiplier sans prendre le temps de les analyser correctement.

D'un point de vue personnel, ce projet fut extrêmement enrichissant tant pour la compréhension et l'approfondissement de diverses théories présentées en RCP209, que pour l'utilisation intensive de MATLAB, ce qui a permis des progrès notables dans son utilisation et dans sa connaissance de toolboxes variées.

Il ne fait aucun doute que les compétences acquises lors de sa réalisation seront des atouts de taille lors des prochains projets académiques ou professionnels dans ce domaine, et l'intérêt que je portais pour ce projet me conforte dans mon orientation vers la recherche en intelligence artificielle.

6. ANNEXE

```
                Predict_size = 1
        ==============================
         === ARV table with DTest ===
        ==============================
 ARV with 1 hidden neurons = 2.203380e-001
 ARV with 2 hidden neurons = 2.231542e-001
 ARV with 3 hidden neurons = 1.843551e-001
 ARV with 4 hidden neurons = 2.306667e-001
 ARV with 5 hidden neurons = 2.009054e-001
 ARV with 6 hidden neurons = 2.102271e-001
 ARV with 8 hidden neurons = 1.817335e-001
ARV with 10 hidden neurons = 1.833652e-001
ARV with 12 hidden neurons = 1.935792e-001
ARV with 24 hidden neurons = 2.045804e-001
ARV with 48 hidden neurons = 1.819394e-001
ARV with 96 hidden neurons = 2.283133e-001
ARV with 144 hidden neurons = 2.238045e-001

                Predict_size = 2
        ==============================
         === ARV table with DTest ===
        ==============================
 ARV with 1 hidden neurons = 2.005895e-001
 ARV with 2 hidden neurons = 1.924723e-001
 ARV with 3 hidden neurons = 1.954813e-001
 ARV with 4 hidden neurons = 2.240467e-001
 ARV with 5 hidden neurons = 2.238328e-001
 ARV with 6 hidden neurons = 2.340797e-001
 ARV with 8 hidden neurons = 2.255071e-001
ARV with 10 hidden neurons = 1.912066e-001
ARV with 12 hidden neurons = 1.886838e-001
ARV with 24 hidden neurons = 2.033288e-001
ARV with 48 hidden neurons = 1.908149e-001
ARV with 96 hidden neurons = 2.009704e-001
ARV with 144 hidden neurons = 2.155398e-001

                Predict_size = 3
        ==============================
         === ARV table with DTest ===
        ==============================
 ARV with 1 hidden neurons = 2.055504e-001
 ARV with 2 hidden neurons = 1.993399e-001
 ARV with 3 hidden neurons = 2.204853e-001
 ARV with 4 hidden neurons = 1.933933e-001
```

```
 ARV with 5 hidden neurons = 1.923204e-001
 ARV with 6 hidden neurons = 1.966903e-001
 ARV with 8 hidden neurons = 2.101883e-001
ARV with 10 hidden neurons = 1.875084e-001
ARV with 12 hidden neurons = 2.482768e-001
ARV with 24 hidden neurons = 2.464094e-001
ARV with 48 hidden neurons = 2.352848e-001
ARV with 96 hidden neurons = 2.183354e-001
ARV with 144 hidden neurons = 2.241612e-001

            Predict_size = 4
       ==============================
        === ARV table with DTest ===
       ==============================
 ARV with 1 hidden neurons = 2.128019e-001
 ARV with 2 hidden neurons = 2.056585e-001
 ARV with 3 hidden neurons = 1.917037e-001
 ARV with 4 hidden neurons = 2.341324e-001
 ARV with 5 hidden neurons = 2.312663e-001
 ARV with 6 hidden neurons = 2.155177e-001
 ARV with 8 hidden neurons = 2.318832e-001
ARV with 10 hidden neurons = 2.320869e-001
ARV with 12 hidden neurons = 1.985817e-001
ARV with 24 hidden neurons = 2.254444e-001
ARV with 48 hidden neurons = 2.514068e-001
ARV with 96 hidden neurons = 2.263992e-001
ARV with 144 hidden neurons = 2.129266e-001

            Predict_size = 5
       ==============================
        === ARV table with DTest ===
       ==============================
 ARV with 1 hidden neurons = 2.062703e-001
 ARV with 2 hidden neurons = 2.389428e-001
 ARV with 3 hidden neurons = 2.107614e-001
 ARV with 4 hidden neurons = 2.126020e-001
 ARV with 5 hidden neurons = 2.346372e-001
 ARV with 6 hidden neurons = 2.224268e-001
 ARV with 8 hidden neurons = 2.436321e-001
ARV with 10 hidden neurons = 2.525732e-001
ARV with 12 hidden neurons = 2.397564e-001
ARV with 24 hidden neurons = 2.154663e-001
ARV with 48 hidden neurons = 2.256831e-001
ARV with 96 hidden neurons = 2.129779e-001
ARV with 144 hidden neurons = 2.185727e-001

            Predict_size = 6
       ==============================
        === ARV table with DTest ===
       ==============================
 ARV with 1 hidden neurons = 2.082782e-001
 ARV with 2 hidden neurons = 1.994813e-001
 ARV with 3 hidden neurons = 2.533819e-001
 ARV with 4 hidden neurons = 2.266663e-001
 ARV with 5 hidden neurons = 2.315919e-001
```

```
 ARV with 6 hidden neurons = 2.290497e-001
 ARV with 8 hidden neurons = 2.343673e-001
ARV with 10 hidden neurons = 2.356483e-001
ARV with 12 hidden neurons = 2.496079e-001
ARV with 24 hidden neurons = 2.388564e-001
ARV with 48 hidden neurons = 2.346452e-001
ARV with 96 hidden neurons = 2.316126e-001
ARV with 144 hidden neurons = 2.313823e-001


            Predict_size = 8
       ==============================
        === ARV table with DTest ===
       ==============================
 ARV with 1 hidden neurons = 2.352311e-001
 ARV with 2 hidden neurons = 2.277938e-001
 ARV with 3 hidden neurons = 2.287229e-001
 ARV with 4 hidden neurons = 2.706733e-001
 ARV with 5 hidden neurons = 2.619420e-001
 ARV with 6 hidden neurons = 2.322185e-001
 ARV with 8 hidden neurons = 2.443067e-001
ARV with 10 hidden neurons = 2.249828e-001
ARV with 12 hidden neurons = 2.268183e-001
ARV with 24 hidden neurons = 2.129321e-001
ARV with 48 hidden neurons = 2.316703e-001
ARV with 96 hidden neurons = 2.300831e-001
ARV with 144 hidden neurons = 2.452892e-001

            Predict_size = 10
       ==============================
        === ARV table with DTest ===
       ==============================
 ARV with 1 hidden neurons = 2.243463e-001
 ARV with 2 hidden neurons = 2.375682e-001
 ARV with 3 hidden neurons = 2.874422e-001
 ARV with 4 hidden neurons = 2.575301e-001
 ARV with 5 hidden neurons = 2.599116e-001
 ARV with 6 hidden neurons = 2.688915e-001
 ARV with 8 hidden neurons = 2.673595e-001
ARV with 10 hidden neurons = 2.602546e-001
ARV with 12 hidden neurons = 2.689160e-001
ARV with 24 hidden neurons = 2.496505e-001
ARV with 48 hidden neurons = 2.359130e-001
ARV with 96 hidden neurons = 2.308238e-001
ARV with 144 hidden neurons = 2.236731e-001

            Predict_size = 12
       ==============================
        === ARV table with DTest ===
       ==============================
 ARV with 1 hidden neurons = 2.257942e-001
 ARV with 2 hidden neurons = 2.679510e-001
 ARV with 3 hidden neurons = 2.715612e-001
 ARV with 4 hidden neurons = 2.335533e-001
 ARV with 5 hidden neurons = 2.859287e-001
```

```
 ARV with 6 hidden neurons = 2.983055e-001
 ARV with 8 hidden neurons = 2.860448e-001
ARV with 10 hidden neurons = 2.637793e-001
ARV with 12 hidden neurons = 2.745780e-001
ARV with 24 hidden neurons = 2.583667e-001
ARV with 48 hidden neurons = 2.638117e-001
ARV with 96 hidden neurons = 2.862301e-001
ARV with 144 hidden neurons = 2.409670e-001

            Predict_size = 48
       ==============================
        === ARV table with DTest ===
       ==============================
 ARV with 1 hidden neurons = 2.903224e-001
 ARV with 2 hidden neurons = 2.893658e-001
 ARV with 3 hidden neurons = 3.283175e-001
 ARV with 4 hidden neurons = 2.914823e-001
 ARV with 5 hidden neurons = 3.497120e-001
 ARV with 6 hidden neurons = 3.000058e-001
 ARV with 8 hidden neurons = 2.907796e-001
ARV with 10 hidden neurons = 2.978004e-001
ARV with 12 hidden neurons = 2.950053e-001
ARV with 24 hidden neurons = 2.731107e-001
ARV with 48 hidden neurons = 2.565473e-001
ARV with 96 hidden neurons = 2.444401e-001
ARV with 144 hidden neurons = 2.698794e-001

            Predict_size = 96
       ==============================
        === ARV table with DTest ===
       ==============================
 ARV with 1 hidden neurons = 5.017224e-001
 ARV with 2 hidden neurons = 4.290527e-001
 ARV with 3 hidden neurons = 4.279755e-001
 ARV with 4 hidden neurons = 3.932672e-001
 ARV with 5 hidden neurons = 3.897071e-001
 ARV with 6 hidden neurons = 4.183855e-001
 ARV with 8 hidden neurons = 4.210123e-001
ARV with 10 hidden neurons = 3.718603e-001
ARV with 12 hidden neurons = 3.940910e-001
ARV with 24 hidden neurons = 3.497657e-001
ARV with 48 hidden neurons = 3.318670e-001
ARV with 96 hidden neurons = 4.005927e-001
ARV with 144 hidden neurons = 4.092691e-001

            Predict_size = 144
       ==============================
        === ARV table with DTest ===
       ==============================
 ARV with 1 hidden neurons = 4.406828e-001
 ARV with 2 hidden neurons = 3.924443e-001
 ARV with 3 hidden neurons = 3.723940e-001
 ARV with 4 hidden neurons = 3.659539e-001
 ARV with 5 hidden neurons = 4.453144e-001
 ARV with 6 hidden neurons = 3.910229e-001
```

```
 ARV with 8 hidden neurons = 3.684032e-001
ARV with 10 hidden neurons = 3.767156e-001
ARV with 12 hidden neurons = 4.248509e-001
ARV with 24 hidden neurons = 3.482406e-001
ARV with 48 hidden neurons = 3.451425e-001
ARV with 96 hidden neurons = 3.445793e-001
ARV with 144 hidden neurons = 3.845200e-001

            Predict_size = 192
       ============================
        === ARV table with DTest ===
       ============================
 ARV with 1 hidden neurons = 4.700531e-001
 ARV with 2 hidden neurons = 3.663336e-001
 ARV with 3 hidden neurons = 3.929684e-001
 ARV with 4 hidden neurons = 3.289303e-001
 ARV with 5 hidden neurons = 3.423164e-001
 ARV with 6 hidden neurons = 3.414302e-001
 ARV with 8 hidden neurons = 3.975448e-001
ARV with 10 hidden neurons = 4.417513e-001
ARV with 12 hidden neurons = 3.562159e-001
ARV with 24 hidden neurons = 3.049545e-001
ARV with 48 hidden neurons = 3.365084e-001
ARV with 96 hidden neurons = 3.282628e-001
ARV with 144 hidden neurons = 3.140193e-001
```

7. BIBLIOGRAPHIE ET URLOGRAPHIE

Voici la liste des principales références que j'ai consultées afin de réaliser ce projet :

1. Keith W. Hipel and A. Ian McLeod (1994), *Time Series Modelling of Water Resources and Environmental Systems*".
2. http://www.wikipedia.org/ : pour obtenir des informations générales sur différents sujets ;
3. http://home.online.no/~pjacklam/matlab/software/util/index.html : utils toolbox, utilisée dans ce projet pour étudier la matrice de corrélation.
4. http://www.stats.uwo.ca/faculty/aim/epubs/mhsets/thompsto/ : pour accéder aux données servant de base à cette étude.
5. http://www.eigenvector.com/MATLAB/corrmap.html : pour le script *corrmap.m* permettant de visualiser des matrices de corrélation.
6. http://www.mathworks.com/access/helpdesk/help/toolbox/stats/brkgqnt.html#f75476 : documentation officielle de MATLAB sur l'Analyse en Composantes Principales.
7. http://www1.aston.ac.uk/eas/research/groups/ncrg/resources/netlab/ : Netlab toolbox, utilisée dans ce projet pour la simulation de réseaux de neurones.
8. http://biosig-consulting.com/matlab/tsa/ : TSA toolbox, utilisée pour l'analyse de séries temporelles non linéaires.
9. http://www.cis.hut.fi/somtoolbox/ : SOM toolbox, utilisée pour établir des cartes auto organisatrices dites de Kohonen.