République Algérienne Démocratique et Populaire
Ministère de l'Enseignement Supérieur et de la Recherche Scientifique

UNIVERSITE ABDERAHMANE MIRA -BEJAIA
Faculté des Sciences Exactes
Département D'Informatique

Mémoire de fin de cycle

En vue de l'obtention du diplôme de Master Professionnel en Informatique
Option :
Administration et Sécurité des Réseaux

Thème :

Simulation statistique de l'écoulement fluvial en vue de dimensionnement d'un barrage

Présenté par :

*M^elle^* BOUFALA Samia
*M^elle^* DJENNADI Syria

Membres du jury :

Président : M^r^ AZNI Mohamed	Université de béjaia
Examinateur1 : M^me^ BELALTA Ramla	Université de béjaia
Examinateur2 : M^me^ Houha Amel	Université de béjaia
Promoteur1 : M^r^ LADJEL Mahmoud	Université de béjaia
Promoteur2 : M^r^ TOUAZI Djoudi	Université de béjaia

Année universitaire : 2016-2017

Remerciements

Nul doute que personne n'est né avec le savoir, il doit ainsi toujours quelque chose à quelqu'un, et c'est pour cette raison que nous tenons à remercier :

Notre dieu qui nous a donné la patience et le courage durant ces longues années d'études.

Nous souhaitons adresser nos remerciements à notre promoteur Mr LADJEL Mahmoud et Mr TOUAZI Djoudi pour leurs compréhension, leurs disponibilité, de savoir-faire, leurs conseils judicieux, et tout aide qu'ils nous ont rapportés.

Nous tenons à remercier Mr AZNI Mohamed qui nous a fait l'honneur de présider le jury de soutenance et Mme HOUHA Amel et BELALTA Ramla, d'avoir acceptées d'examiner le manuscrite. Qu'ils sachent que nous sommes très honorés par leurs présences au jury de soutenance.

Nos remerciements s'entendent également au corps professoral et administrative de la faculté de sciences exactes pour la qualité de leurs enseignements.

Dans l'impossibilité de citer tous les noms nos sincères remerciements vont à tous ceux et celles, qui de près ou de loin, ont permis par leurs conseils et leurs compétences la réalisation de ce mémoire.

Dédicaces

Je dédie ce modeste travail ;

À ma très chère mère, affable, honorable, aimable, tu représentes pour moi la source de la tendresse et l'exemple du dévouement qui n'a pas cessé de m'encourager et prier pour moi, ta prière et ta bénédiction m'ont été d'un grand secours pour mener à bien mes études ;

À mon très cher père, rien au monde ne vaut les efforts fourmis jour et nuit pour mon éducation et mon bien être, ce travail et le fruit de tes sacrifices que tu as consentis de mon éducation et ma formation ;

À mon chère grand-mère Wahshia, que dieu le porte dans ces paradis ;

À mes très chères sœurs, Aida, Nacéra, Djahida, Sonia et El djida et son épou et leurs fils, notre adorable Mayasse ;

À mes très chers frères, Ahsen, Lyazid et Samir ;

À mes chères cousines et mes chers cousins ;

Votre affectation de soutien m'ont été d'un grand secours au long de ma vie, les mots ne suffisent pas guère pour exprimer l'attachement, l'amour et l'affection que je porte pour vous ;

À mon cher et meilleur ami, Mohamed lamine qui est toujours à mes côtés, il a un grand soutien pour moi,

À mes chèr(e)s ami(e)s, vous êtes pour moi des frères et des sœurs sur qui je peux compter, je vous dédie avec tous mes vœux du bonheur, de santé et de réussite.

Son oublier ma binôme Syria et sa famille qui ma partager ce travail.

Samia

Dédicace :

Je dédie ce travail spécialement à :

À mes très chers parents, faible témoignage de ma reconnaissance pour leurs sacrifices et efforts tout au long de ma formation, que dieu vous garde pour moi *je vous a*ime *;*

À m*es chère frère Anis et Chakib ;*

À m*a petite sœur Elycia que j'aime ;*

À m*a grand-mère de côté paternel Aouicha ;*

À m*on grand-père de côté maternel Mohand el aid et ma grand-mère Yamina ;*

À m*es oncles et Mes tantes ;*

À mes cousins et mes cousines et à toute la famille ;

À ma très chère binôme et copine Samia et sa famille ;

À ma meilleure amie Kahina que j'aime, elle est pour moi une sœur;

À tous mes amis et tous ceux qui me sont chers ;

À tous ceux qui m'aimes et j'aime, merci pour tous ;

A tous mes enseignants, qui m'ont aidé pour avoir contribuée à ma formation.

Syria

Table des figures

Table des tableaux

La liste des abréviations

α : la restitution.

β utile : La valeur somme déficits par apport au fréquence P'%.

C_v : coefficient de variation.

C_s : coefficient d'asymétrie.

D_0 : La moyenne des déficits.

Di : les déficits.

F^*: la valeur Foster.

$F_{P'\%}$: La valeur Foster de pourcentage P'.

K_i **:** Coefficient modulaire d'écoulement.

$K_{P'\%}$: Le module de l'écoulement.

NA : Nombre aléatoire.

P : la fréquence à l'aide des nombres aléatoires.

P'% : La fréquence complémentaire de P.

V_0 : Le volume moyenne d'un barrage.

V_{utile} : Le volume utile d'un barrage.

X : la valeur de l'écoulement annuel.

X_0 : la moyenne de la série de l'écoulement.

X_i : la série de l'écoulement.

Table des matières

INTRODUCTION GENERALE

Le rôle essentiel de l'ordinateur est de traiter de l'information. D'où le terme informatique généralement utilisé pour désigner l'ensemble des techniques permettant d'automatiser un tel traitement. L'informatique, en tant que discipline scientifique, se révèle de nos jours un outil indispensable à la résolution de problèmes complexes dans le calcul ou le traitement et s'applique à des domaines aussi variés que la gestion, les calculs scientifiques, l'ingénierie, le diagnostic médical, etc.

Les ressources en eau en Algérie sont irrégulièrement réparties dans le temps et dans l'espace. Ces précipitations sont observées durant la saison pluvieuse, qui s'étale de septembre à avril et se caractérisent par une importante variabilité. Le plus souvent, le régime naturel de l'écoulement fluvial ne concorde pas avec le régime de son utilisation. Car les besoins en eau de différents secteurs économiques ne sont pas conformes avec le régime naturel de l'écoulement [1]. Ajouté à cela la diminution progressive, d'année en année, de cette ressource vitale. En effet l'eau devient de plus en plus la préoccupation majeure de certains pays, surtout d'Afrique. Le réchauffement de la planète terre a créé des conséquences très graves sur l'avenir de l'eau dans le monde. D'où le besoin inévitable de la gestion minutieuse et optimale de cette dernière est indispensable pour la bonne exploitation et usage de toutes les ressources hydriques dans le monde. L'outil informatique est le moyen primordial pour prendre en charge la gestion scientifique des données statistiques relevées et calculées périodiquement et de prendre les bonnes décisions se rapportant à cette ressource.

À ce stade, on pose le problème des calculs compliquées et énormes effectués par les hydrauliciens et surtout des données, à la base, aléatoires qui restent difficiles à générer par le cerveau humain. Ici, on voit bien l'intérêt d'informatiser tous ces calculs.

L'objectif visé par ce projet est la simulation statique de l'écoulement fluvial en vue de dimensionnement d'un barrage qui sera comme un calculateur plus exacte et assuré, qui permet à l'utilisateur de trouver des résultats, faire des comparaisons si nécessaire. Pour aboutir à cet objectif, notre travail est organisé en quatre chapitres :

Dans le premier chapitre titré '' **Méthodes de calcul hydraulique''** a pour but de définir les méthodes essentielles et les fonctions mathématiques sur lesquelles est fondé notre travail.

Le second chapitre intitulé « **Analyse et conception** » » a pour but d'atteindre les algorithmes dont l'objectif est de fournir un raisonnement informatique, afin de les présenter en organigrammes associés.

Le troisième chapitre est consacré à « l'**Implémentation dans un langage adéquat** ». À ce niveau, on va traduire les algorithmes en programme exécutable écrit en langage java.

Le dernier chapitre est consacré à « **La réalisation** ». Nous définirons les outils de développement que nous avons utilisés. Nous présenterons également l'interface et le fonctionnement du cette application.

Chapitre I : Méthodes de calculs hydrologique

Introduction :

Nous examinerons dans ce chapitre les méthodes générales de calcul de régularisation de l'écoulement fluvial, pour le dimensionnement hydrologique d'un barrage. De telles méthodes sont tributaires des données hydrométéorologiques et topographiques. Ces méthodes possèdent des caractéristiques générales comme le montre cette étude bibliographique.

On examine trois groupes de méthodes des calculs de gestion économique d'eau (régularisation), les méthodes chronologiques, les méthodes statistiques et la méthode de modélisation statistique.

I.1 Quelque notion de paramètres

Avant d'entamer ce chapitre on va donner quelques définitions pour les différentes paramètres utilisées à fin de facilités la compréhension de la démarche

L'écoulement fluvial : L'évaluation quantitative des variables hydrologiques constitue l'étape importante dans toute analyse statistique. Souvent en hydrologie, les calculs des écoulements probables s'appuient sur une distribution statistique donnée, qui nécessite au moins trois paramètres: la moyenne, le coefficient de variation et le coefficient d'asymétrie. [I.1]

Module de l'écoulement k: En hydrologie, le module correspond au débit moyen inter-annuel, c'est une synthèse des débits moyens annuels d'un cours d'eau sur une période de référence (années de mesures consécutives).

Coefficient de variation Cv : Le coefficient de variation est une mesure relative de la dispersion des données autour de la moyenne. Le coefficient de variation se calcule comme le ratio de l'écart-type rapporté à la moyenne, et s'exprime en pourcentage. Il permet de comparer le degré de variation d'un échantillon à un autre, même si les moyennes sont différentes.

Coefficient d'asymétrie Cs : Correspond à une mesure de l'asymétrie de la distribution d'une variable aléatoire réelle. C'est le premier des paramètres de forme, En termes généraux, l'asymétrie d'une distribution est positive

I.2 Les données hydrologiques et topographiques

I.2.1 Les données hydrologiques

Les paramètres du barrage et celles du régime de son exploitation sont déterminés sur la base des données de l'écoulement d'eau et celles du transport solide par le courant d'eau. Les informations de base sont les données des mesures hydrométriques à la section du projet de l'ouvrage hydraulique ou dans une autre section, où le régime hydrologique est étudié au cours d'une période suffisamment longue, sur une longue période de plusieurs années. L'élaboration de l'étude hydrologique, du projet des barrages, est soumise à des règles normalisées en vigueur, qui guident les démarches des calculs hydrologiques relatifs au dimensionnement des barrages. [I.2]

I.2.2 Les données topographiques

Les principales caractéristiques topographiques du barrage sont les courbes de dépendances des superficies du plan d'eau (S) et les volumes d'eau correspondant (V) en fonction du niveau d'eau. Pour le calcul de ces caractéristiques, il est nécessaire d'avoir, en général, les cartes topographiques à grande échelle.

I.2.3 La caractéristique générale des méthodes de régularisation

L'établissement des paramètres du barrage d'eau (les différents volumes, la restitution et le régime de son fonctionnement) est réalisé sur la base de l'analyse du régime de l'écoulement fluvial. Dans ce cas, il est entendu que les fluctuations des valeurs de l'écoulement sont de nature aléatoire, soumise à certaines lois seulement dans le sens des distributions probabilistes. Seule la variation saisonnière de l'écoulement a une régularité du caractère fonctionnel, qui se manifeste sous forme cyclique de changement des phases dans le temps durant l'année [I.3].

À partir des principes indiqués ci-dessus, dans la théorie existante de la régularisation de l'écoulement, on examine trois groupes des méthodes de calcul de gestion d'eau, concernant l'utilisation de la série d'observations hydrologiques de l'écoulement fluvial pour la période passée.

I.2.3.1 Les méthodes tabulaires (dites de chronologiques) :

C'est l'exécution directe des calculs chronologiques réels, des séries hydrologiques avec le traitement ultérieur statistique de leurs résultats, qui ne sont pas influencées par l'activité humaine. Cette méthode suppose que les données utilisées dans le calcul, des observations passées reflètent toutes les régularités complexes de l'écoulement dans le futur. Son avantage est sa clarté et sa polyvalence (plusieurs applications) dans le sens du domaine d'utilisation, qui est très importante lors de l'élaboration des plans d'exploitation des barrages et la régularisation compensatoire dans les systèmes complexes de gestion d'eau. Cependant une courte série d'observations de l'écoulement est entachée par des erreurs considérables pour la définition de n'importe quels paramètres de gestion d'eau.

I.2.3.2 Les méthodes probabilistes :

La réalisation des calculs de la régularisation de l'écoulement est fondée sur les paramètres statistiques ses séries de références d'écoulement. Les méthodes probabilistes permettent théoriquement l'estimation des probabilités pour diverses alternances d'hydraulicité des rivières. Elles permettent d'exclure finalement l'erreur possible dans l'estimation de la capacité régulatrice des barrages lors des calculs par des séries hydrologiques courtes.

I.2.3.3 La méthode de modélisation statistique :

Elle est appelée aussi la méthode des expériences statistiques de Monte-Carlo. L'idée principale de cette méthode consiste dans la création du modèle statistique de l'écoulement fluvial. La méthode a les caractères généreux des deux méthodes citées ci-dessus, concernant l'utilisation des données hydrologiques, dans l'accomplissement des calculs de gestion d'eau. Ce qui est commun avec la première méthode est qu'ici, les calculs de gestion économique de l'eau sont effectués directement par la série hydrologique, qui diffère de la série observée. Celle-ci est créée théoriquement par simulation du processus de l'écoulement fluvial. La réalisation d'une telle simulation, est obtenue connaissant les paramètres statistiques établis, c'est-à-dire par la fonction de la distribution des probabilités de l'écoulement, qui est considérée générale avec la deuxième méthode [I.3].

I.3 Description d'étapes à suivre :

I.3.1 Courte série :

Les données sont représentées par une série de variables (écoulements annuels) X_i.

I.3.1.1 Première étape :

Dans cette étape on va présenter les traitements à faire pour cette série X_i, est l'écoulement fluvial de chaque i année. Cette série de valeurs X est représenté comme un ensemble de valeurs: Xi = { X_1, X_2, …., X_n / i=1..n } ; avec n : nombre d'années et i qui fait référence à une année d'écoulement. Il est préférable de désigner l'écoulement sous forme de coefficient modulaire d'écoulement, c'est-à-dire ki=Xi/Xo, avec Xo comme moyenne arithmétique des valeurs de la série.

- **La moyenne X_0:**

On fait la somme de X_i diviser sur le nombre d'années (n), pour trouver la moyenne X_i .

$$X_0 = \frac{\sum_{i=1}^{n} X_i}{n} \qquad (1)$$

- **Coefficient modulaire d'écoulement k_i :**

$$k_i = \frac{X_i}{X_0} \qquad \{ k_1 = \frac{X_1}{X_0};\ k_2 = \frac{X_2}{X_0}; \cdots\cdots;\ k_i = \frac{X_i}{X_0} \} \qquad (2)$$

La caractérisation statistique de cet échantillon de n années doit être exprimée par un certain nombre de paramètres : moyenne, coefficient de variation et coefficient d'asymétrie. Ces paramètres sont déterminés par la méthode des moments.

- **Coefficient de variation :**

$$C_v = \sqrt{\frac{\sum_{i=1}^{n} (k_i - 1)^2}{n-1}} \qquad (3)$$

- **Coefficient d'asymétrie :**

$$C_s = \frac{n \sum_{i=1}^{n} (k_i - 1)^3}{(n-1)(n-2) C_V{}^3} \qquad (4)$$

D'autre cas, on utilise la fonction des nombres aléatoires NA, qui génère une série des nombres aléatoires comprit entre 0 et 1. Sur la bases de la série obtenue des nombres aléatoires, qu'on peut les transformer en fréquences P^* qui varient entre a et b. on exprime ces fréquences comme suit :

$$P^* = NA\,(b - a) + a \qquad (5)$$

Avec,

a = 0.01 borne inférieurs des fréquences.

b = 99.99 borne supérieurs des fréquences.

Le choix des valeurs de ces bornes permet l'utilisation du tableau Foster-Rebkine, exprimant les cordonnées de la loi binomiale à trois paramètres en annexe 1.

La détermination des valeurs des modules du l'écoulement correspondant à ces fréquences nécessite une opération d'interpolation pour déterminer les valeurs de nombre de Foster F*. Pour cela on doit recourir à la formule du Newton pour un pas irrégulier. Cette formule s'écrit :

$$F^* = F_{i,j} + \frac{(F_{i+1,j} - F_{i,j})}{(P_{i+1} - P_i)}(P^* - P_i) + \frac{(F_{i,j+1} - F_{i,j})}{(C_{sj+1} - C_{sj})}(C_s{}^* - C_{sj}) + \frac{(F_{i,j} + F_{i+1,j+1} - F_{i,j+1} - F_{i+1,j})}{(P_{i+1} - P_i)(C_{sj+1} - C_{sj})}(P^* - P_i)(C_s{}^* - C_{sj})$$

Avec :

i : l'ordres des colonnes et j : l'ordre des lignes.

$[P_i, P_{i+1}]$: intervalle contenant P*. $[C_{sj}, C_{sj+1}]$: intervalle contenant Cs*.

$[F_{i,j}, F_{i+1,j}, F_{i,j+1}, F_{i+1,j+1}]$: champs comprenant la valeur F* cherchée.

Ci-dessous, on donne un schéma explicatif de l'opération pour la détermination de la valeur du nombre de Foster F* correspondant aux valeurs de Cs* et P*.

Soit : P*=7%, dans l'intervalle : [P_i=5 et P_{i+1}=10]

et Cs*= 0.28. dans l'intervalle : [Cs_j=0.25 et Cs_{j+1}=0.30]

d'où le domaine de F* est $[F_{i,j}, F_{i+1,j}, F_{i,j+1}, F_{i+1,j+1}]$

La figure suivante détermine ces étapes et explique comment récupérer la valeur Foster d'une série.

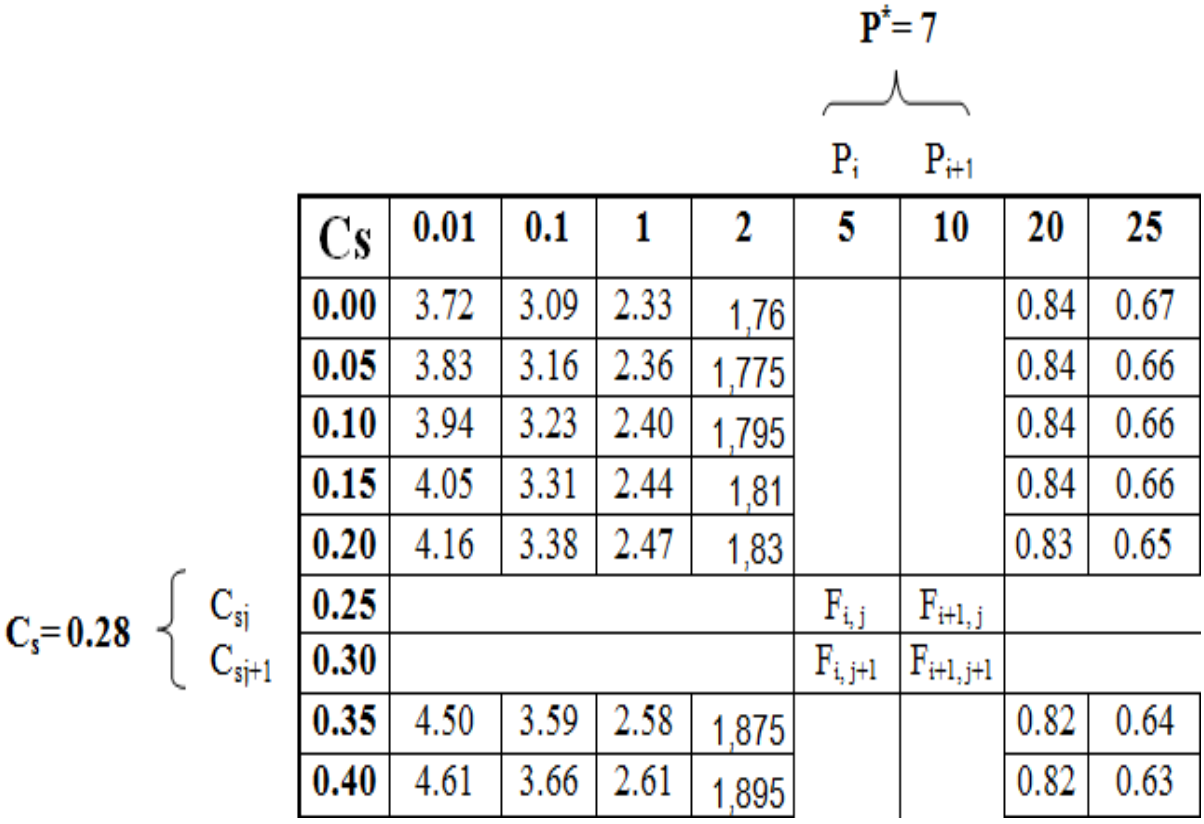


Figure I.1 : Schéma détermine comment récupérer la valeur Foster d'une série.

- **Module d'écoulement (nombre aléatoire) :**

Connaissant la valeur de F*, la valeur de module de l'écoulement ki est exprimée par la relation :

$$k^* = F^* C_v + 1$$

Ainsi pour le nombre des valeurs aléatoires modélisées, on obtient autant de valeurs des modules de l'écoulement :

$$k_i^* = F_i^* C_v + 1.$$

I.3.1.2 Deuxième étape :

Ayant fixé la valeur relative de la restitutionα, exprimant le volume d'eau des besoins par rapport au volume moyenne de l'écoulement. Sachant que le volume utile d'un barrage est égale au volume maximum des déficits sur une période de n années consécutives de fréquence P* = 100 – P où P exprime la garantie de la restitution.

Tout d'abord, on détermine les différences ki - α et on tient comptes uniquement des années déficitaires ki - α < 0. La sommation des déficits des années consécutives montre que le déficit peut s'étendre d'une année à plusieurs années, séparer par des périodes excédentaires quand ki - α > 0, ce que signifie que la restitution est garantie à 100%.

- **Les déficits D_i :**

La procédure du traitement des déficits consiste à calculer les différences ki – α pour toutes les années de la série modélisée. Seules les différences négatives ki – α < 0 font l'objet de ce traitement. Les déficits durant une année ou plusieurs années consécutives constituent la série des déficits, indépendamment de la taille des périodes déficitaires.

Le déficit d'une seule année est exprimé par :

$$D_i = k_i - \alpha$$

Le déficit sur une période de plusieurs années consécutives est exprimé par :

$$D_i = \Sigma(k_i - \alpha)$$

Ainsi, on obtient une série des déficits Di.

- **La valeur absolue de somme des déficits :**

Le traitement statistique de la série des valeurs des déficits SD_i , transformées en valeurs absolues , consiste à déterminer les paramètres statistique (valeur moyenne SDo, coefficient de variation Cv et coefficient d'asymétrie Cs). Le volume utile relatif du barrage est détermine, comme étant le déficit SD pour une fréquence égale a 100 – P%, soit :

$$\beta_{utile,\ P\%} = SD_{100 - P\%}$$

La valeur du déficit SD pour une fréquence égale a 100 – P% est exprimé comme suit :

$$SD_{100 - P\%} = (F_{100-P\%,\ Cs} * Cv + 1) * SDo$$

Où :

SDo: la valeur moyenne des déficits

Cv : coefficient de variation des déficits

Cs : coefficient d'asymétrie des déficits

P% : la garantie de la restitution

$SD_{100 - P\%}$: le déficit pour la fréquence complémentaire 100 – P%

$F_{100-P\%,\ Cs}$: nombre de Foster correspondant à la fréquence 100- P% et à la valeur de Cs.

- **Le volume utile V_{utile} :**

Le volume utile du barrage est exprimé par :

$$V_{utile,\ P\%} = \beta_{utile,\ P\%} * V_0$$

Où :

Vo : volume moyen de l'écoulement.
$\beta_{utile,\ P\%}$: Volume utile relative du barrage.

CHAPITRE II : Analyse et conception

Introduction :

Quand on utilise une carte à microprocesseur ou un microcontrôleur, il faut fournir au processeur, les instructions à exécuter pour réaliser la fonction désirée. Pour cela, on doit lui fournir un programme exécutable. Pour créer ce programme exécutable, il faut analyser de manière rigoureuse les tâches à faire exécuter au microprocesseur et en déduire les actions à effectuer, pour les traduire en programme.

II.1 Principaux composants physiques

Le mécanisme de traitement des informations peut être schématisé avec trois dispositifs d'entrée « E » pour l'entrée des données, de traitement « Traitement » sur ces données et de sortie « S » fournissant les résultats de ce traitement ;

- Pour pouvoir traiter les données rapidement et sans intervention en cours de calcul, il est nécessaire de pouvoir ranger et conserver les résultats intermédiaires, d'où la nécessité de disposer d'une mémoire
- La rapidité des échanges et du traitement implique l'exigence d'une unité qui fait le contrôle de ces différentes opérations.
- Certains problèmes exigent une grande place mémoire. Dans ce cas, une mémoire secondaire est à adjoindre au système. D'où le schéma général d'un ordinateur est :

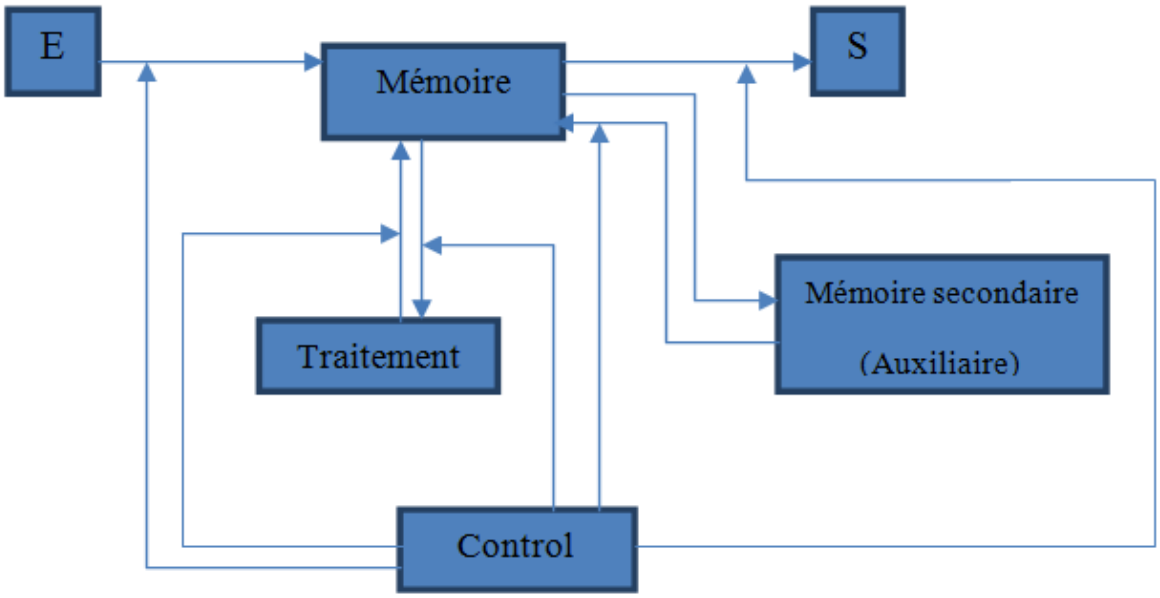


Figure 2.1 : Schéma résumant le mécanisme de traitement des informations.

Pour que l'ensemble fonctionne correctement, il est donc nécessaire de décrire à l'avance la suite complète des opérations à réaliser appelée « programme ». [II.1]

II.2 Étude et traduction

Après l'étude des tâches à réaliser, il faut traduire et décomposer le travail à effectuer en sous tâches à l'aide

II.2.1 Présentation

Il est impossible d'écrire un programme correct du premier coup, depuis la première ligne jusqu'à la dernière. Un programme est l'aboutissement d'un travail méthodique résumé par le schéma suivant : [R1]

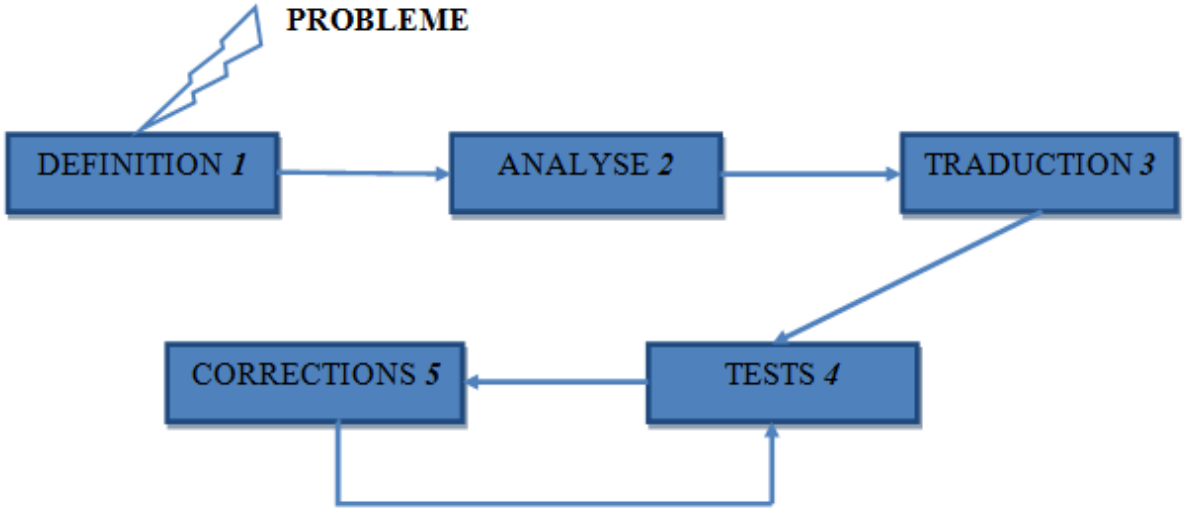


Figure II.2 : Schéma présentant les étapes à suivre pour créer un programme.

II.2.2 Définition du problème

Il faut être capable de définir très précisément tout ce que l'on attend du programme. Pour cela, il faut connaître quelles sont les informations nécessaires à la résolution du problème :

- Qui va les fournir et sous quelle forme ?
- Est-on sûr de la validité de ces données ou faut-il les contrôler pour s'assurer qu'elles sont conformes à ce que l'on avait prévu ?
- Il faut également réfléchir à la présentation des résultats.

Exemple: Écrire un programme qui permet d'entrer une série courte (n ; Xi) des nombres.

II.2.3 Analyse structurée et pseudo code ou Algorigramme

Une fois les données déterminées et les résultats explicités, il ne reste plus qu'à trouver la méthode qui permet d'obtenir les résultats à partir des données. Si la solution n'est pas évidente du premier coup d'œil, il est nécessaire de le découper en sous-problèmes, et de le structurer.

Lorsqu'un problème est trop complexe, il faut le découper en plusieurs tâches distinctes, auxquelles on attribue un nom significatif, de manière que le problème paraisse résolu avec la résolution des sous-problèmes. Puis on traite chaque tâche suivant le même processus, en la subdivisant en sous-tâches jusqu'à ce que celles-ci deviennent évidentes à résoudre.

La structuration consiste à organiser les sous-tâches obtenues par la méthode de découpage, de manière que leur agencement réponde effectivement au problème posé. Une fois cet agencement trouvé, il reste à le traduire en pseudo code.

Exemple : Trouver la moyenne X_0 à partir de la série courte saisis au clavier :

- ***En Pseudo Code :***

```
Algorithme : X0 ;
Variables :
      X : tableau [1...n] entier;
      i, n, Som, X0 : entier;
Début
   X[i];
   Som ← 0 ;
     Pour i allant de 1 à n faire
      Som ⟵ Som + X[i];
     Finpour;
     X0 ⟵ Som div n;
Fin ;
```

- ***En organigramme :***

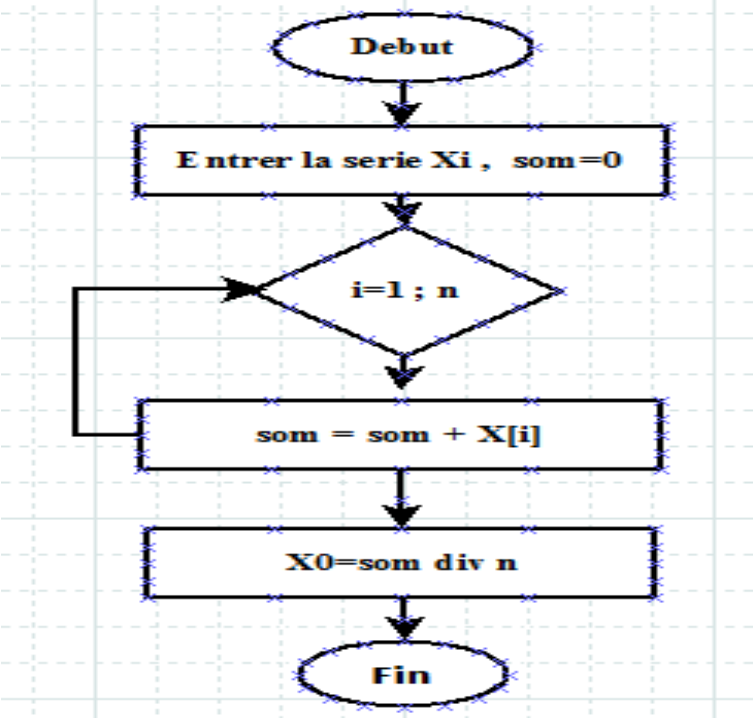


Figure II.3 : Algorigramme de la moyenne X0 de série Xi.

II.2.4 Traduction

Une fois la décision prise du langage utilisé, si l'analyse et la structuration et la traduction en pseudo code (ou en Algorigramme) ont été faite correctement, la traduction ou le codage au langage désiré est simple à réaliser.
On prend l'exemple de calcul de la moyenne en Java.

Exemple: Langage en java

```
public double X0(String chaine)
{int l=0;double som=0;
   double tableau[]=new double[chaine.length()];
final java.util.StringTokenizer tokenizer = new java.util.StringTokenizer(chaine,
"/");
while (tokenizer.hasMoreTokens()) {
   tableau[l]= Double.parseDouble(tokenizer.nextToken().toString());
   l++;
  }
int i=0;
while(i<l)
{
   som=som+tableau[i];
  // System.out.println(tableau[i]);
   i++;
}
jLabelX0.setText(" "+l);
return som/i;
}
```

Figure II.4: Un programme qui calcule la moyenne X_0 sous Java.

II.2.5 Tests et exécution du programme

Il reste maintenant à passer la dernière étape, qui est en fait l'aboutissement de tout notre travail, à savoir si notre programme remplit bien les tâches prévues. D'un autre terme, il est important de ***tester*** l'algorithme et de vérifier s'il ***réalise*** ce qu'on lui demande de faire.

- Exemple : On essaie de tester le programme, pour cela, on va calculer la moyenne de courte série Xi de (0.25, 0.5, 0.98, 1.5, 2.15, 1.96) :

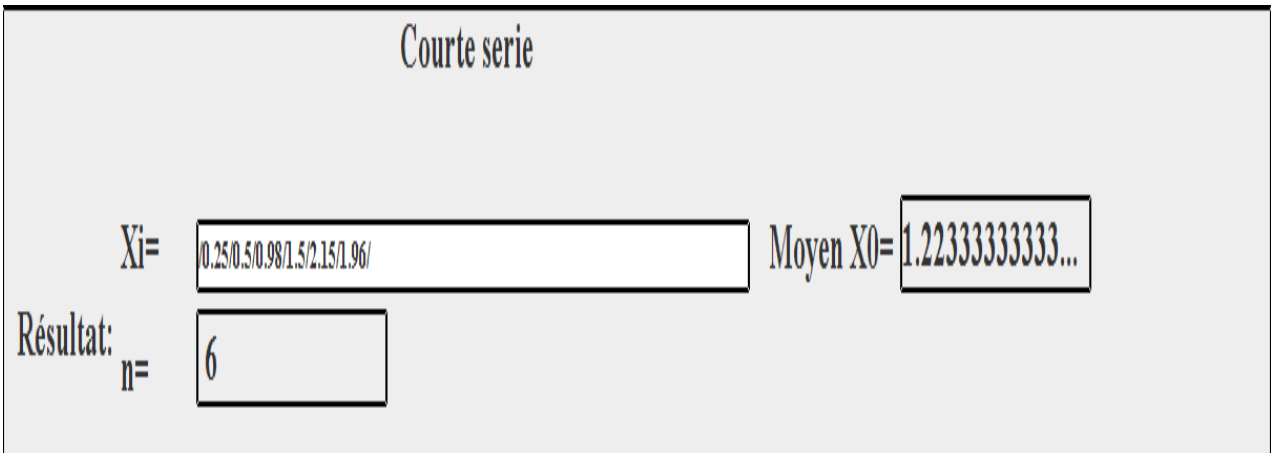


Figure II.5: Teste sur l'interface graphique.

II.2.6 Corrections

Cette étape est normalement inutile si l'analyse et la structuration ont été faites correctement.

II.2.7 Documentation

Cette étape est souvent négligée à tort car elle permet une meilleure utilisation du programme, et surtout une maintenance aisée. Il faut penser à l'utilisateur qui ne connaît pas forcément bien l'informatique. Vous vous êtes sûrement déjà retrouvé devant un programme avec sa documentation et ne pas savoir par où commencer.

II.3 Définition de problème

- Tout d'abord on va donner une définition précise du problème posé, d'après le premier chapitre là où on a précisé les formules qu'on doit suivre dans le but d'atteindre le résultat final qui est « trouvé le volume utile d'un barrage », sans oublier de signaler qu'il faut utiliser la méthode de 'Monter Carlo' sur la génération des nombres aléatoires :

Premièrement, on va présenter ces formules mathématiques nécessaires dans notre travail sous forme algorithmique.
Deuxièmement, il faut à chaque fois tester et contrôler les résultats retournés par un exemple déjà vérifier, pour assurer la fiabilité des résultats.
Finalement, on pense à la présentation finale ; utiliser des champs de saisi et d'affichage.

- Avant d'entamer la deuxième étape il faut d'abord passer par une présentation générale sur les algorithmes et les algorigrammes, pour comprendre la signification des syntaxes.

II.4 Généralités sur les algorithmes et les algorigrammes

II.4.1 Définitions

- **Algorithme :** procédure de calcul bien définie qui prend en entrée une valeur, ou un ensemble de valeurs, et qui donne en sortie une valeur, ou un ensemble de valeurs. Un algorithme est donc une séquence d'étapes de calcul qui transforment l'entrée en sortie [II.2]. Autrement dit, un algorithme est une spécification d'un schéma de calcul sous forme d'une suite finie d'opérations élémentaires reflétant les instructions d'un programme. Ainsi, cela représente la description des étapes à suivre pour réaliser un travail. [II.3]
- **Algorigramme :** c'est une représentation graphique de l'algorithme utilisant des symboles normalisés. ils sont relativement simples et peuvent être compris par tout le monde.
- **Diagramme :** est une suite de directives composées d'actions et de décisions qui doivent être exécutés selon un enchaînement strict pour réaliser une tâche (ou séquence).

II.4.2 Présentation syntaxique d'un algorithme

```
Algorithme   nom de l'algorithme          // entête
Var          constant, réel, entier       // déclaration des variables
Proc/Fonc    procedure ou fonction ;      // déclaration des fonctions et procédures.
Debut
        Action1                           // corps principale de l'algorithme.
        Action2
```

Fin

II.4.3 Caractéristiques d'un algorithme et d'algorigramme

Un algorithme doit avoir les caractéristiques suivantes :

- La netteté : c'est-à-dire, la précision qui ne laisse aucune place à l'arbitraire. Grâce à cette propriété, la réalisation d'un algorithme est procédé mécanique.
- L'efficacité : c'est-à-dire la propriété de conduire au résultat recherché après un nombre fini d'opérations.

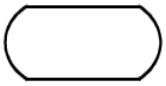
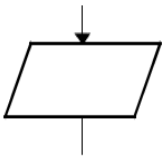
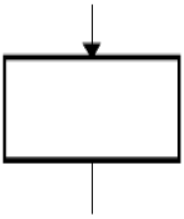
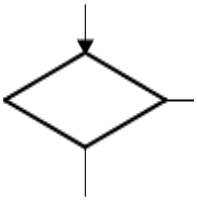
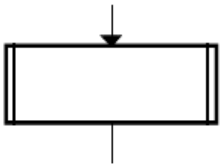
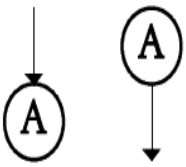
Un algorigramme doit avoir les caractéristiques suivantes :

- Un algorigramme montre bien la succession des instructions et d'opération guidés par les liaisons (flèches), un algorigramme se construit de haut en bas.
- Les symboles standards sont normalisés
- Il peut être compris par des personnes n'ayant aucune connaissance de la programmation.
- Il peut servir à diviser le projet entier en sous-tâches. Il sert alors à évaluer les progrès accomplis.
- Il décompose les séquences d'opérations, ainsi trouver les erreurs. [II.4]

II.4.4 Symboles d'un algorigramme et significations algorithmiques

Afin de présenter un algorigramme, il existe un ensemble de symboles normalisés. Chaque symbole a une signification d'une instruction en algorithme, le tableau suivant résume les plus fréquents :

Tableau II.1: Présentation des différents symboles d'un algorigramme et leurs significations.

Symbole	Désignation	Symbole	Désignation
	**Début et fin** Utiliser au début et fin d'un programme.		**Entrée / Sortie** Fait pour la lecture et l'écriture.
	**Traitement :** opération sur des données, instructions, ou opération pour laquelle il n'existe aucun symbole normalise.		**Test ou branchement conditionnel :** Décision d'un choix parmi d'autre en fonction des conditions.
	**Sous-programme :** appel d'un sous-programme procédure ou fonction.		**Commentaire** Dire quelque chose dans un commentaire.
	**Liaison :** Les différents symboles sont reliés entre eux pas des lignes de liaison. Le cheminement va de haut en bas et de gauche à droite. Un cheminement différent est indiqué à l'aide d'une flèche.		**Renvoi vers un autre point du programme :** Quand le programme est assez complexe on peut remplacer une longue ligne de connexion par des repères numérotés pour clarifier la représentation.

II.4.5 Les instructions de base d'un algorithme

On note que les types de données de base « variables » peuvent êtres : entiers (12, 732), caractères ('a', 'g', '3'), réels (-12.3 0,5 _ p3), chaines (de caractères), booléens (vrai ou faux), tableaux ([1.2.6.5.4]) [II.5].

- **Les Entée/Sortie**

Lecture d'une donnée : Lire (une variable)

Affichage d'une expression : Écrire (liste d'expressions, de variables ou de constantes)

- **Affectation**

L'affectation : Variable ← expression

- **Les conditions**

Sélection si-sinon : **Si** expression-logique **alors**
instruction(s) à effectuer si l'expression est vraie
sinon
instruction(s) à effectuer si l'expression est fausse
finsi ;

- **Les boucles**

Itération répéter : **repeter**
instruction(s) à répéter
jusqu'à expression-logique

Itération tant-que : **Tantque** expression-logique
instruction(s) à répéter
fintantque ;

Itération pour :

Pour variable <- val-initial a val-fin> faire
instruction(s) à répéter
finpour ;

II.4.6 Déclaration des fonctions, procédures

Les fonctions et les procédures sont deux structures fondamentaux dans l'algorithmique, vue leurs intérêt, de simplifier les algorithmes en fusant appelle au sous-programme (fonction ou procédure) et de minimiser le taux d'erreurs.

II.4.6.1 Déclaration de la fonction

Une fonction est un bloc d'instructions nommée et paramétrée, réalisant une certaine tâche. Elle admet zéro, un ou plusieurs paramètres et renvoie toujours un résultat. La syntaxe de déclaration d'une fonction est la suivante :
Fonction nom fonction (paramètre1 : type,…, paramètre n : type): type valeur ;

II.4.6.2 Déclaration de la procédure

Tout comme une fonction, une procédure est un ensemble d'ordre accomplissant une tâche particulière, mais ne renvoie pas des résultats. La syntaxe de déclaration d'une procédure est la suivante :

Procédure nom procédure (paramètre1 : type,…, paramètre n : type) ;

II.4.6.3 Appel de sous-programme

Un appel de fonction est une expression du type de retour de la fonction.

- **Que se passe-t-il lors de l'appel ?**

X ← Moyenne (a,b,c,d) ;

Les données sont remplacées par des valeurs (ou des expressions)
Le code de la fonction est exécuté jusqu'au premier return.

Le résultat retourné par la fonction est la valeur de l'expression du return.
Ce résultat (valeur) est récupéré dans la variable X [II.6].

II.5 Algorithme et algorigramme des fonctions

Dans ce que suit, on décrit les algorithmes (fonction et procédure) qui réalise une analyse des fonctions mathématiques présenté dans le premier et les algorigramme associent.

- **Algorithme Calculs**

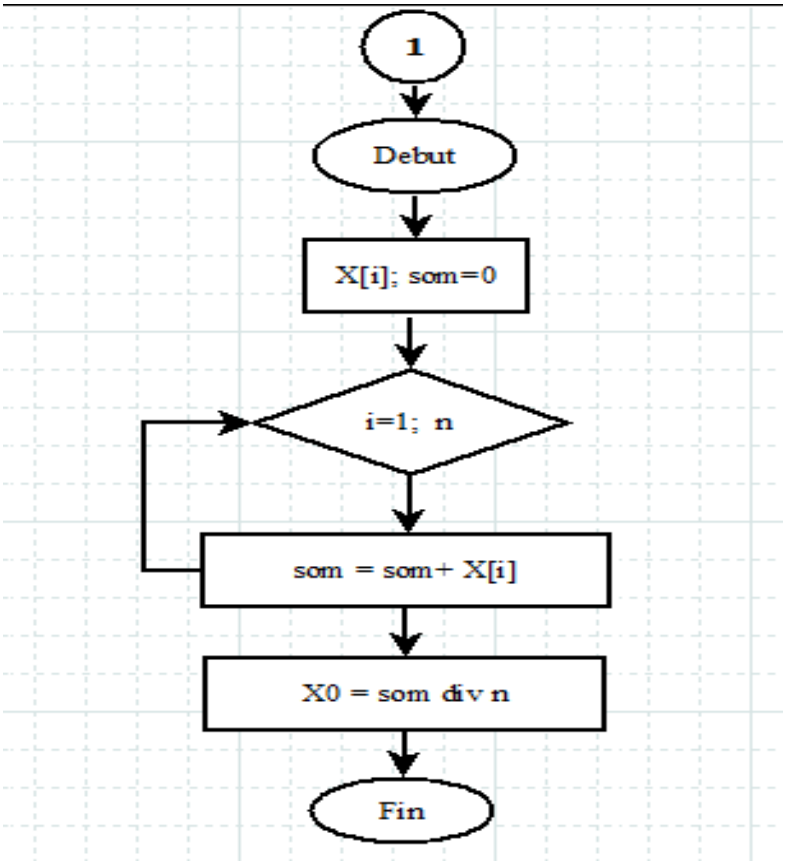
Var βutil , Vutil ,V0, Kprim ,Fprim,

dCv ,dCs :entier ;

- **Algorithme qui calcul la moyenne X0**

Fonction moyenne (X:tableau[1..n]entier) :entier
Var i, n, Som : entier;
Début
Lire(X[i]) ;
Som ⟵ 0 ;
Pour i allant de 1 à n faire
Som ⟵ Som + X[i];
Finpour;
Moyenne ⟵ Som div n;
Fin ;

- **Algorigramme de la moyenne X0**

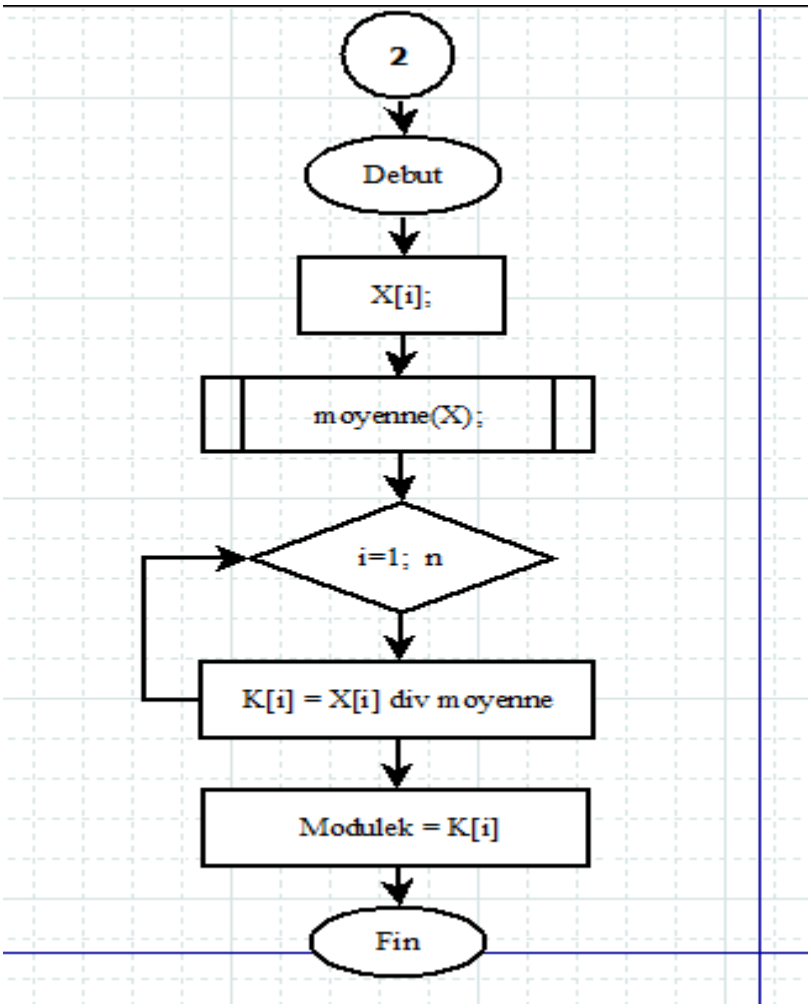


FigureII.6 : Algorigramme de la fonction moyenne.

- **Algorithme qui calcul le Coefficient de l'écoulement ki**

```
Fonction  ModuleK(X : tableau[1…n]entier) :entier
Var   i, n, k : entier ;
Debut
    Lire(X[i]) ;
    Lire(moyenne) ;
  Pour i allant de i à n faire
     K[i]   ⟵   X[i] div moyenne ;
   Finpour ;
ModuleK   ⟵   K[i] ;
Fin ;
```

- **Algorigramme de coefficient de l'écoulement ki**



FigureII.7 : Algorigramme de module de l'écoulement.

- **Algorithme coefficient de variation :**

Fonction CoefV(K :tableau[1…n]entier) : entier

Var i, n, k, S : enteir ;

Debut

S ⟵ 0;

Lire(ModuleK ( ));

Pour i allant de 1 à n faire

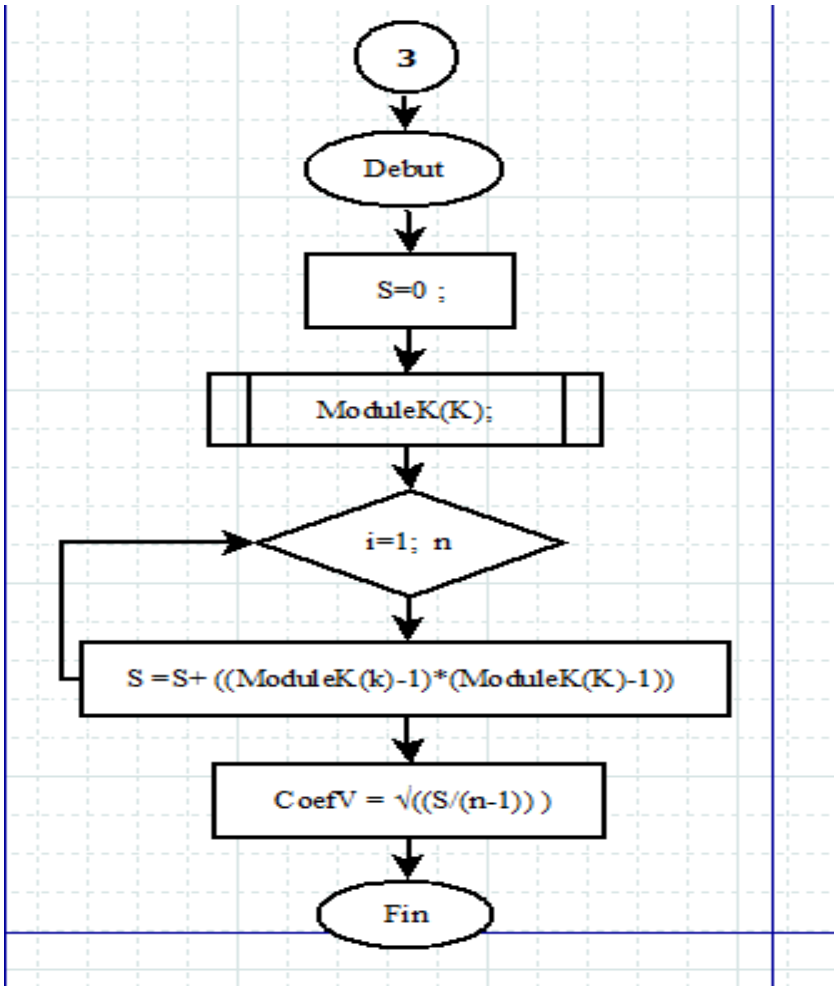
S ⟵ S + (ModuleK(k) -1)*(ModuleK(k) -1);

Finpour;

CoefV ⟵ $\sqrt{(S/(n-1))}$;

Fin ;

- **Algorigramme de coefficient de variation**

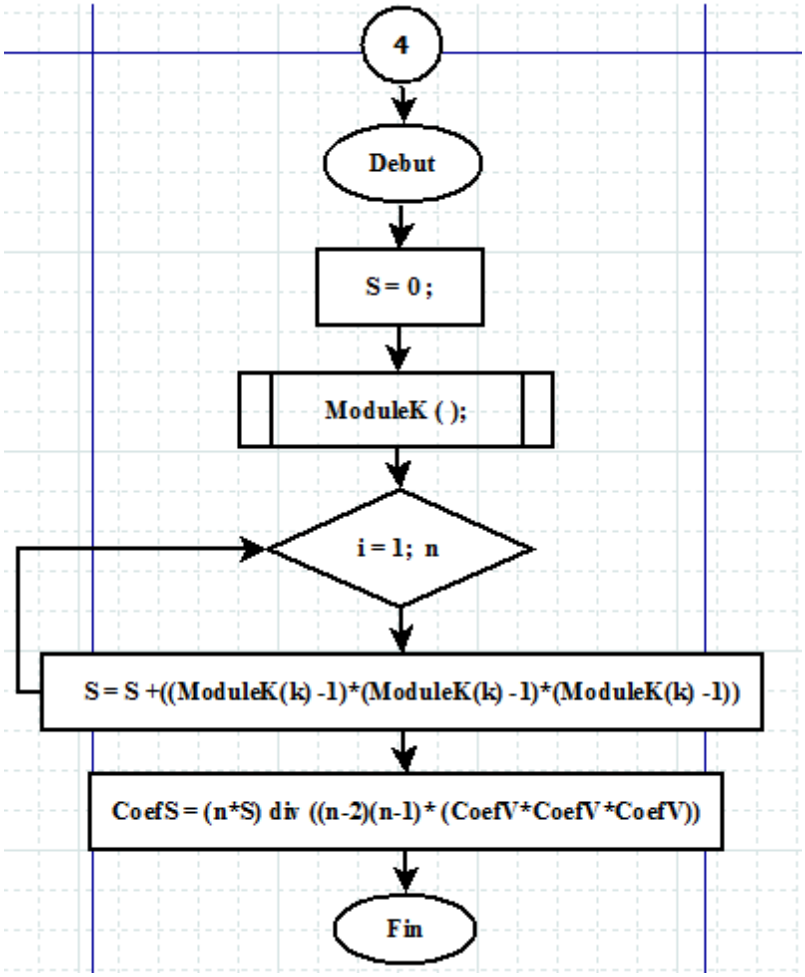


FigureII.8 : Algorigramme de coefficient de variation.

- **Algorithme coefficient d'asymétrie :**

```
Fonction  CoefS(tabk :tableau[1…n]entier) : entier
Var          i, n, k, S : enteir ;
Debut
   S ⟵  0;
   Lire( ModuleK () );
   Pour i allant de 1 à n faire
      S ⟵  S + (ModuleK (k) -1)*(ModuleK(k) -1)* (ModuleK(k) -1);
   Finpour;
  CoefS  ⟵  (n*S) div ((n-2)(n-1)* (CoefV*CoefV*CoefV)) ;
Fin;
```

- **Algorigramme de coefficient d'asymétrie**



FigureII.9 : Algorigramme de coefficient d'asymétrie.

- **Algorithme qui génère le nombre aléatoire :**

Fonction GenerNa(Na : entier) : entier

Var i, n, alea : entier ;

Debut

Pour i allant de 1 à Na faire

Générer (alea) u ⟶ u [0,1]

alea = F^{-1} (U) ;

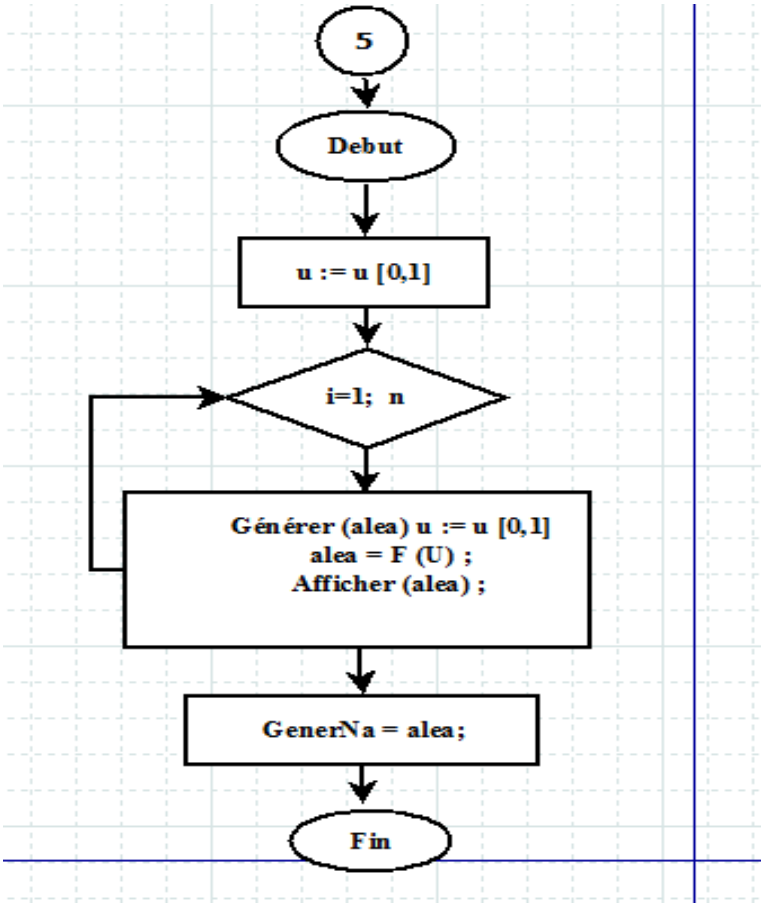
Afficher (alea) ;

Finpour ;

GenerNa ⟵ alea ;

Fin ;

- **Algorigramme génère le nombre aléatoire**

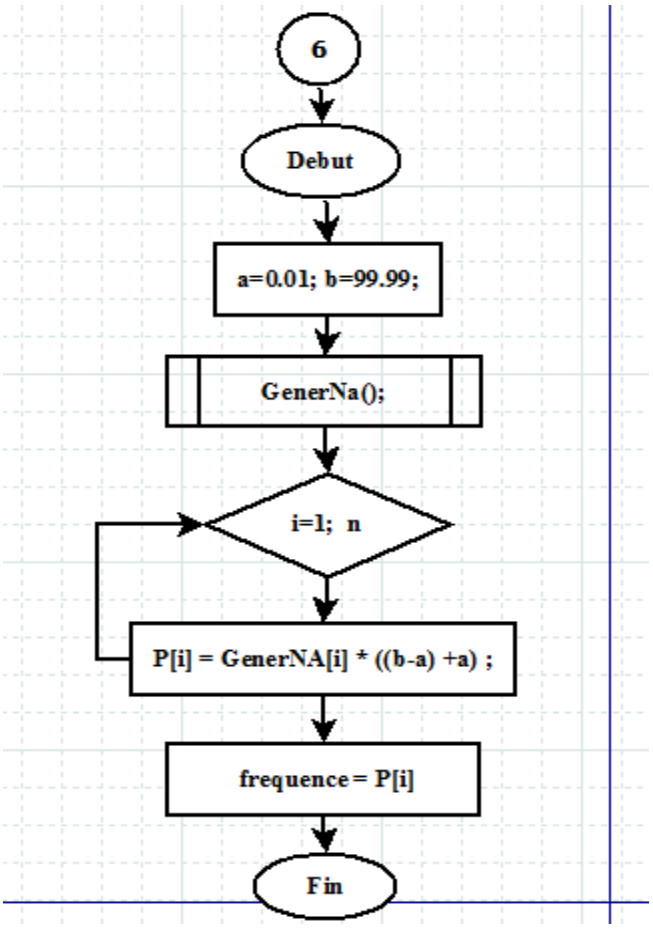


FigureII.10 : Algorigramme génère le nombre aléatoire.

- **Algorithme qui calcul la fréquence :**

```
Fonction  freqence(Na :entier ;)entier
Var       P : tableau[0…Na]entier;
          P, i, a, b, n :entier ;
Debut
   Lire (GenerNa(Na));
   a=0.01; b=99.99;
      Pour i allant de 1 à Na faire
        P[i]  ⟵  GenerNa[i] * ((b-a) +a);
     Finpour;
   Freqence  ⟵  P[i] ;
Fin;
```

- **Algorigramme la de fréquence P***

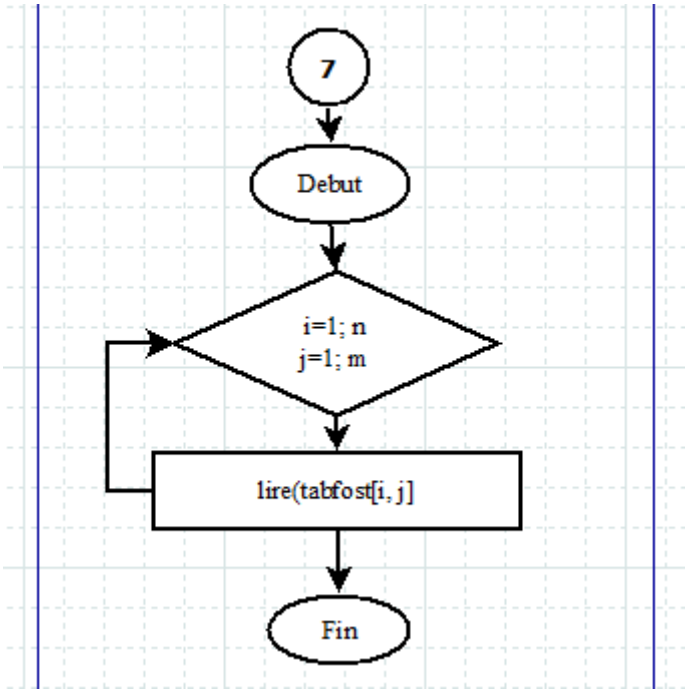


FigureII.11 : Algorigramme de fréquence P*.

- **Algorithme qui remplit le tableau foster :**

```
Procedure  tabfoster(tabfost: tableau[0..n][0..m] entier);
Var            i, j, n, m: entier;
Debut
  Pour i allant de 1 à n faire
     Pour i allant de 1 à m faire
       Lire(tabfost(i,j)) ;
     Finpour ;
  Finpour ;
Fin ;
```

- **Algorigramme de tableau foster**



FigureII.12 : Algorigramme de tableau Foster.

- **Algorithme qui calcul la fréquence complémentaire**

Fonction Prim (P :entier) :entier ;

Var P' : eniter ;

Debut

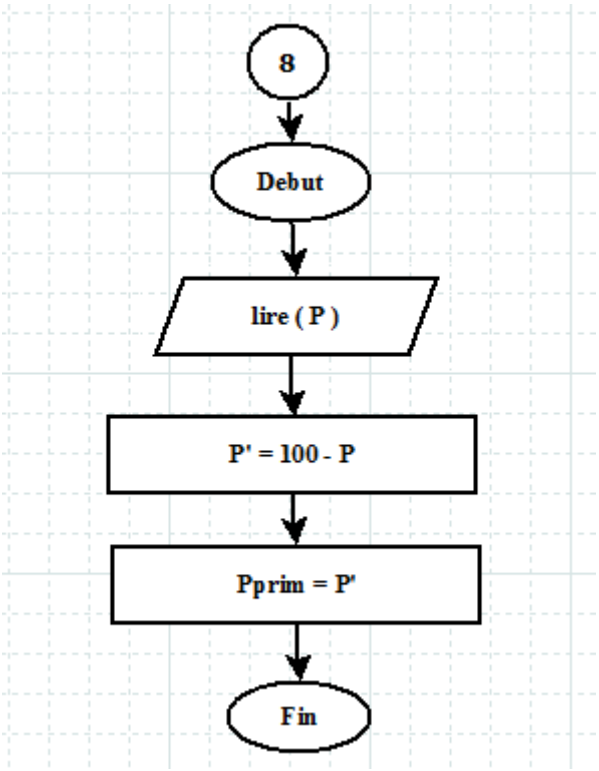
Lire(P) ;

P' ⟵ 100 - P ;

Prim ⟵ P' ;

Fin ;

- **Algorigramme de la fréquence complémentaire P'%**



FigureII.13 : Algorigramme de la fréquence complémentaire P'%.

- **Algorithme qui calcul la valeur Foster** :

```
Fonction   valfoster(P :entier ; Cs :entier ;)entier ;
Var          i, j, n, m, matfost: entier ;
Debut
   tabfoster(n, m);
    répéter
       i=i+1;
    jusqu'à ( P  >  tabfoster [i,0]  &&  i < =  n)
    répéter
         j=i+1;
    jusqu'à ( Cs  >  tabfos t[0,j]  &&  j < =  m)
 pour  i allant de 1 à n faire
  pour j allant de 1 à m faire
         matfost[0,0]  ⟵  tabfost[i-1,j-1];
         matfost[0,0]  ⟵  tabfost[i-1,j];
         matfost[0,0]  ⟵  tabfost[i,j];
         matfost[0,0]  ⟵  tabfost[i,j-1];
  F  ⟵  matfost[0][0]+(( matfost [1][0]- matfost[0][0])/(tabfost [i+1][0]-t tabfost [i][0]))*(P-
    tabfost [i][0])+(( matfost[0][1]-matfost[0][0]/ tabfost [0][j+1]-tableaufoster [0][j]))*(Cs- tabfost
    [0][j]) +(( matfost [0][0]+ matfost [1][1]- matfost [0][1]- matfost[1][0]) /(tabfost [i+1][0]- tabfost
    [i][0])*( tabfost [0][j+1]- tabfost [0][j]))*   (P- tabfost [i][0])* (Cs- tabfost [0][j]);
Finpour;
Finpour;
Fin;
```

- **Algorigramme de la valeur Foster**

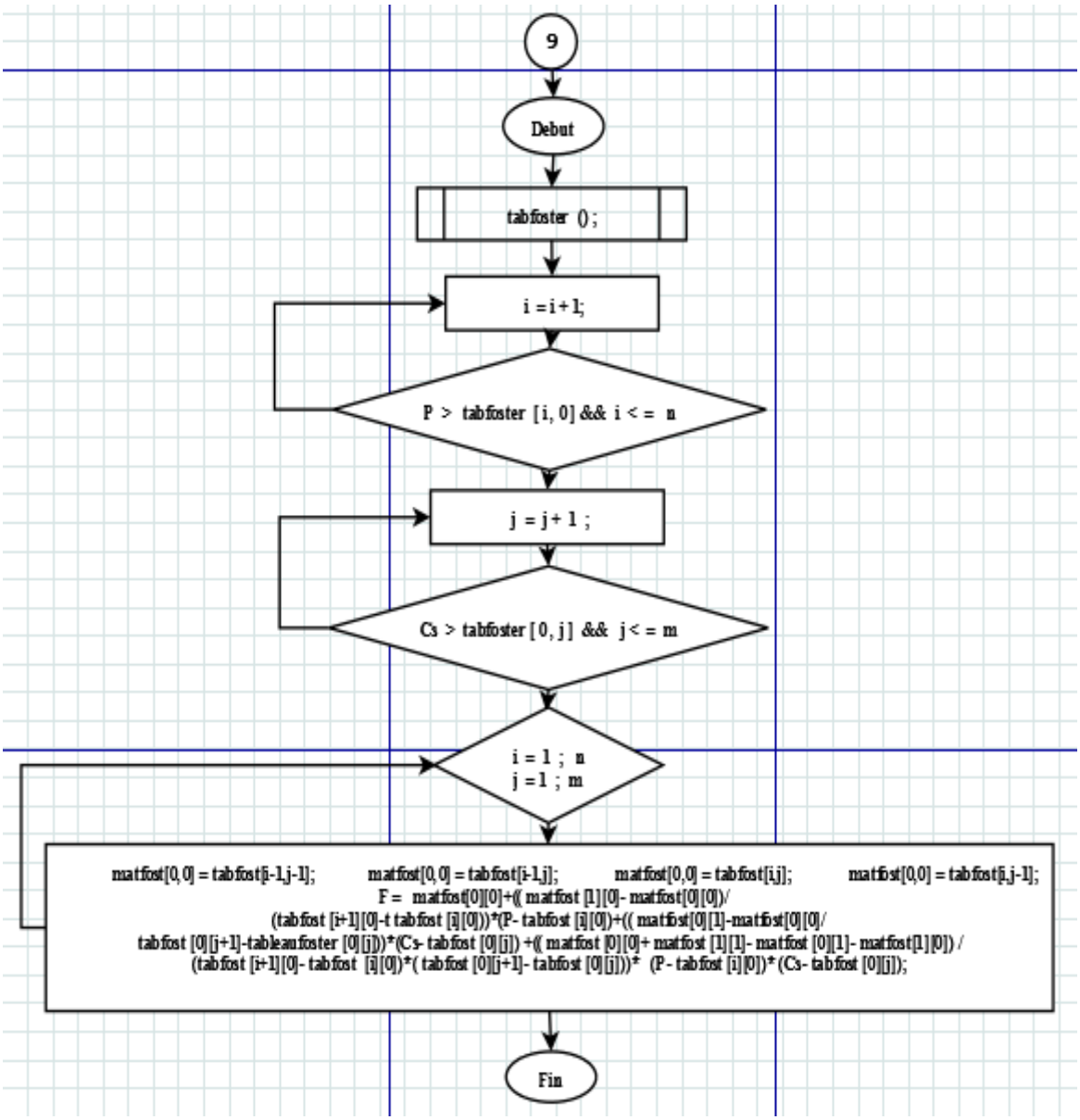


Figure II.14 : Algorigramme de la valeur Foster.

- **Algorithme qui récupère le module de fréquence**

```
Fonction        ModuleF ( tabF: tableau[0...n]entier;   Cv :entier) entier ;
Var             K=tableau[0..n]entier;i,  n : entier ;
Debut
Lire (valfoster [i]) ;
 Pour i allant de 1 à n faire
    tabF [i]  ⟵  valfoster [i];
    K[i]  ⟵  (valfoster[i] * Cv ) +1 ;
 Fin pour;
  ModuleF ⟵  k[ i] ;
Fin ;
```

- **Algorigramme de module d'écoulement de fréquence P'%**

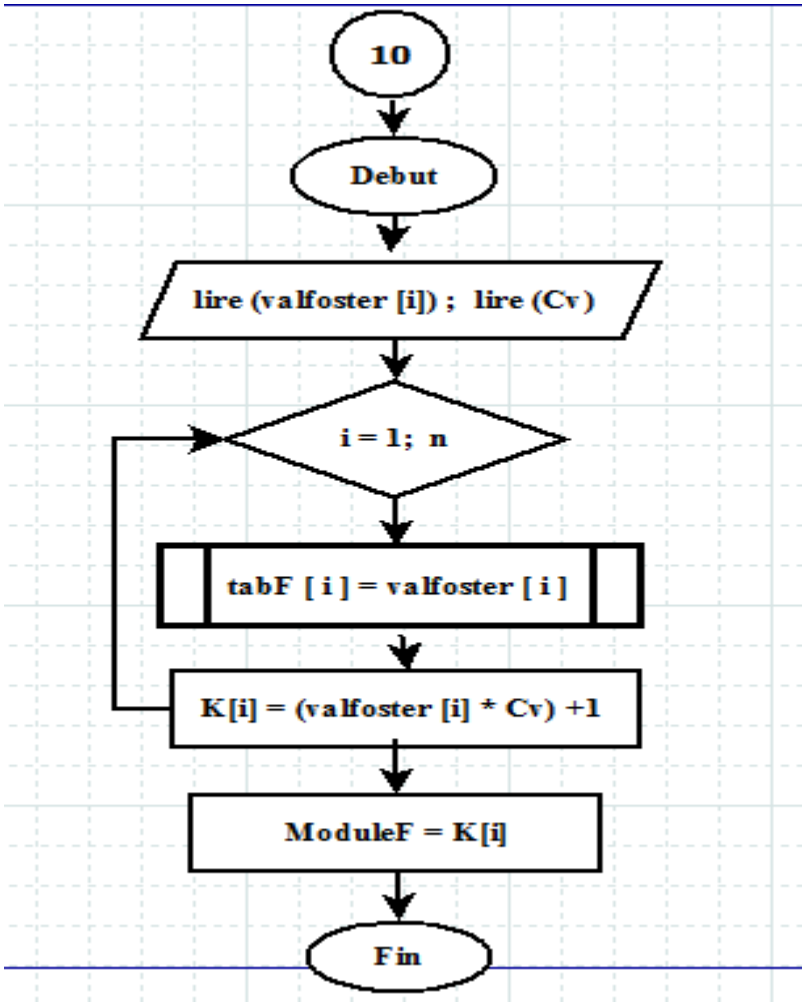


Figure II. 15: Algorigramme de module d'écoulement de fréquence P'%.

- **Algorithme qui calcul les déficits**

Fonction Deficits(tab k : tableau([0..n] :entier; α :entier) :entier ;

var i,n,a,val, D : entier ;

debut

lire(val);

α=val;

pour i allant de 1 à n faire

D[i] ⟵ ModuleF[i]- α ;

Finpour ;

Deficits ⟵ D[i];

Fin;

- **Algorigramme de la série des déficits**

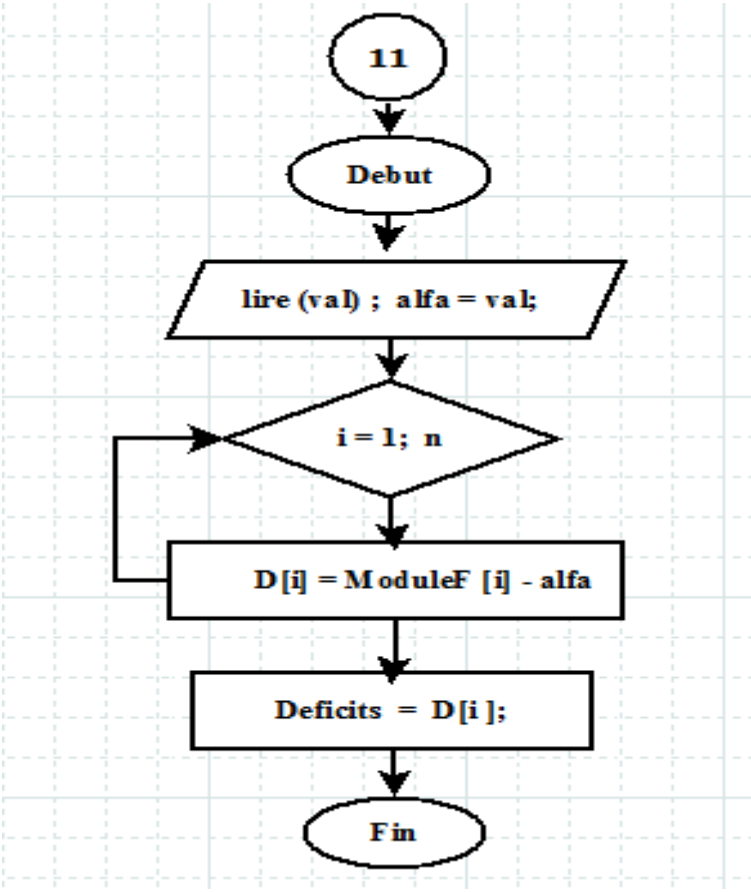


Figure II.16 : Algorigramme des déficits.

- **Algorithme qui calcul la somme des déficits**

```
Fonction        Somdeficits (D : tableau[0..n] :entier ) :entier ;
var             S=  tableau[0..n] entier ; S,i,n :entier ;
debut
   lire(D[i]);
  pour i allant de 1 à n faire
    Si D[i]>0 alors
       i  ⟵  i+1 ;
    Sinon
      Si (D[i] < 0 et D[i+1] < 0) alors
     S[i]  ⟵  abs ( D[i] + D[i+1] ) ;
        Sinon
          Si (D[i] < 0 et D [i+1]> 0 ) alors
            S[i]  ⟵  abs ( D[i] ) ;
   Finsi;
Finsi;
Finsi;
  Ecrire( S[i] );
Finpour;
Somdeficits  ⟵  S[i];
Fin;
```

- **Algorigramme de la série des déficits**

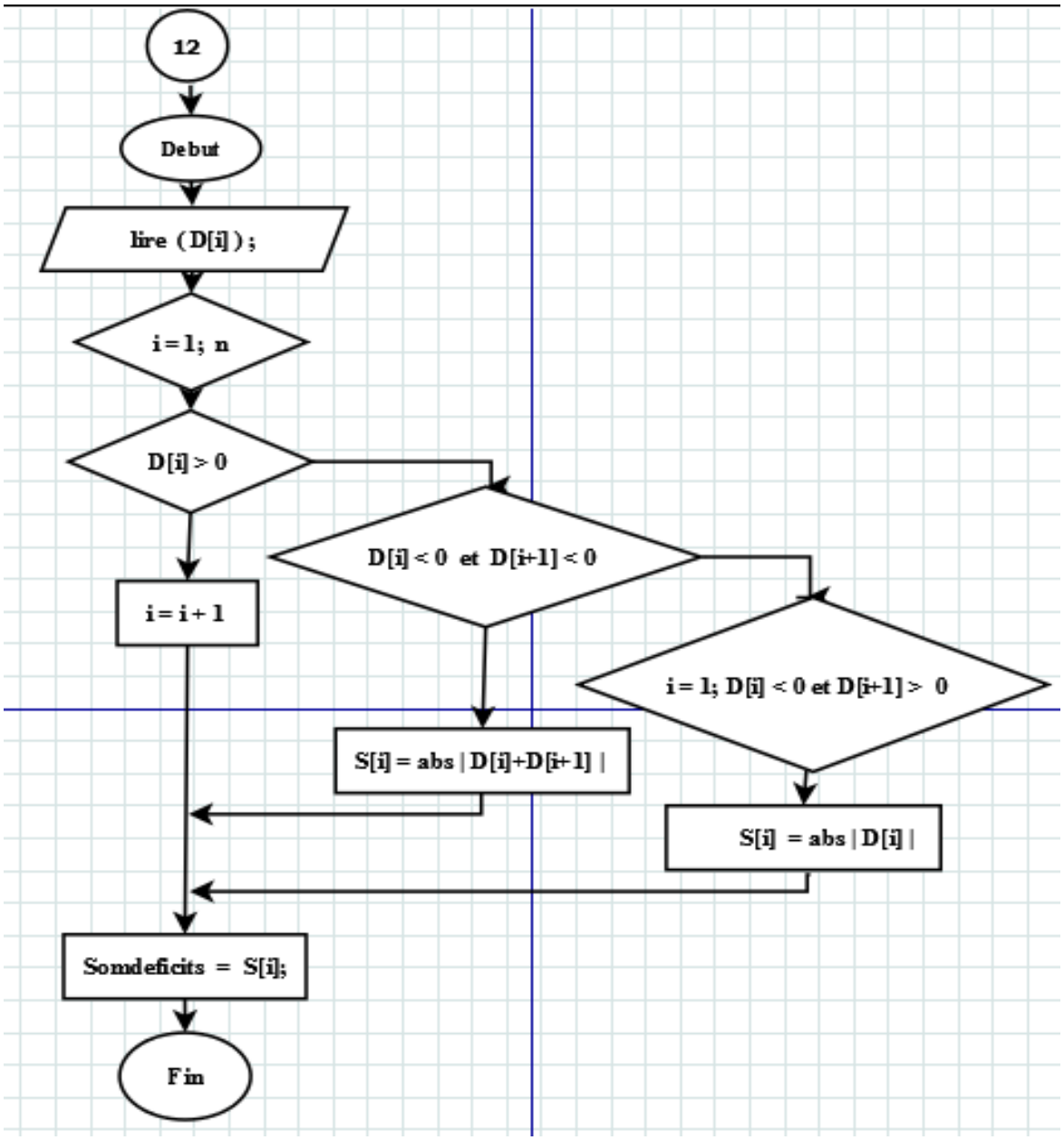


Figure II.17 : Algorigramme de somme des déficits.

- **Programme principal**

À ce niveau-là il reste qu'à faire des appels aux sous-programmes pour assurer la cohérence des calculs.

Debut

Lire(V0) ;

Lire(X [i]) ;

Moyenne (X [i]) ;

Modulek (X [i]) ;

Cv ⟵ CoefV (X [i]) ;

Cs ⟵ CoefS (X [i]) ;

GenerNa (Na) ;

Freqence (Na);

Tabfost (22,79);

Valfoster(Frequence (Na), Cs);

ModuleF(valfoster ( ),Cv);

Deficits(Modulek ( ) , α);

Somdeficits(D ( ));

Lire(SD ( ));

Moyenne(SD ( ));

dCv ⟵ coefv(SD [i]) ;

dCs ⟵ coefs(SD[i]);

Fprim ⟵ valfoster(P'prim, dCs);

Kprim ⟵ (Fprim*dCv) + 1;

βutil ⟵ kprim*moyenne(Somdeficits [i]);

Vutil ⟵ βutil *V0;

Ecrire (Vutile) ;

Fin ;

- **Algorigramme principal**

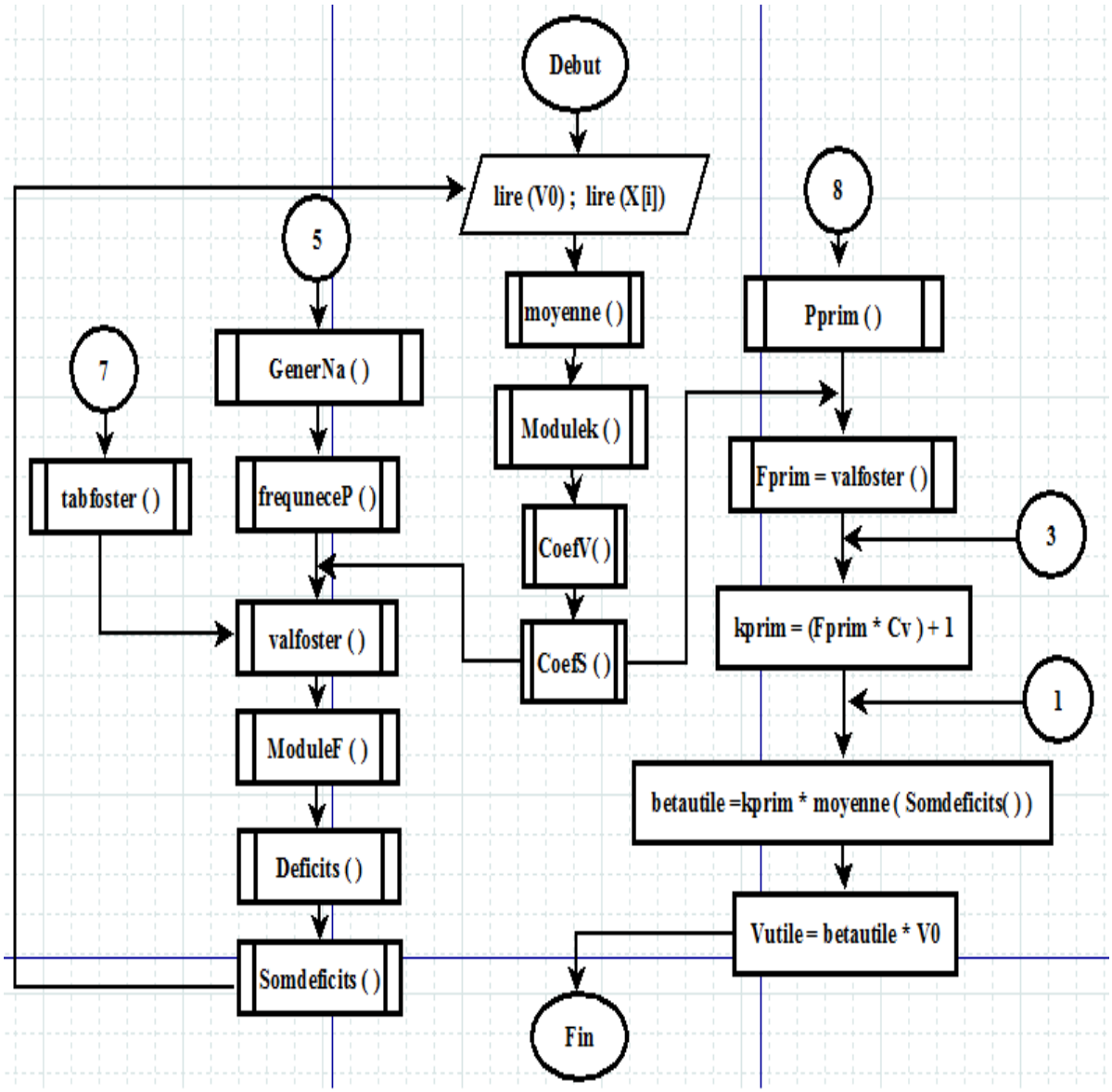


Figure II.18 : Algorigramme du programme principal.

Chapitre III : Implémentation en langage adéquat

Ce chapitre présente la syntaxe des instructions de base java permettant de définir des variables de différents types dits "primitifs" et de contrôler le déroulement du programme, ces syntaxes qui vont nous assurer de toucher à la troisième étape (traduction).

III.1 Terminologie de base

Adresse : Indique la localisation d'une information dans la mémoire.

Affectation : L'affectation est l'opération qui attribue une valeur à une variable. En java cet opérateur est '='. Il se lit : "prend la valeur de ".

Appel de méthodes : L'appel d'une méthode exécute le bloc d'instruction de la méthode. L'appel se fait en écrivant le nom de la méthode suivie d'une paire de parenthèses avec éventuellement une liste de paramètres effectifs séparés par une virgule. Au moment de l'exécution de l'appel, les valeurs de paramètres effectifs sont effectuées aux paramètres formels selon l'ordre dans lequel ils apparaissent. Si le type renvoyé par la méthode est différent de *void*, l'appel de la méthode doit être affecté à une variable de même type.

Attribut : Les attributs sont des variables associées à un objet.

Console : La console est une interface textuelle qui permet de demander à l'ordinateur d'exécuter des programmes. Elle est souvent considérée à tort comme étant obsolète. Pour autant il s'agit d'une des interfaces les plus puissantes à utiliser puisque l'on peut directement programmer de la console. De plus, il s'agit bien souvent de l'unique façon d'accéder à des machines à distance.

Compilation : La compilation permet de transformer une source écrit dans un langage de programmation en langage machine (ou langage binaire) exécutable par l'ordinateur.

Déclaration : Avant de pouvoir utiliser un variable ou une méthode, il faut la déclarer. La déclaration d'une variable associe un type à un nom réserve un emplacement mémoire dans lequel est stocké la valeur de la variable, si son type est primitif ou l'adresse du début de la plage mémoire ou les stocké la variable si son type est primitif la définition d'une méthode s'appelle également déclaration. Elle consiste le nom de la méthode, le type de la valeur, retournée la liste de ses paramètres formel et son bloc d'instruction.

Décrémenter : L'opération de décrémentation s'applique à une variable de type entier. Elle consiste à retirer une valeur entière à la variable, classiquement la valeur 1 (x = x - 1).

Exécution : L'exécution est le processus par lequel une machine met œuvre les instructions d'un programme informatique.

Incrémenter : L'incrémentation est l'opération qui consiste à ajouter une valeur entière fixée à une variable de type entier. La valeur ajoutée est le plus souvent la valeur 1 (x = x + 1).

Initialisation : Lorsque une déclare variable, un emplacement mémoire est associé à la variable. Or, celui-ci contient une valeur quelconque. Il est nécessaire d'initialisé la valeur de la variable c'est-à-dire de réaliser une première affectation d'une valeur à la variable, afin que cette dernier contienne une valeur appropriée.

Instance de class : On appelle instance d'une classe, un objet avec un comportement (méthodes) et un état (attributs), tous deux définis par la classe. Il s'agit donc d'un objet constituant un exemplaire de la classe.

Instancier : Réserver l'espace mémoire nécessaire pour stocker toutes les valeurs de l'objet. De manière plus générale, fabrique un exemplaire d'un élément à partir d'un modèle qui lui sert quelque sorte de moule.

Opérateur : Un opérateur est une fonction spéciale dont l'identificateur s'écrit avec des caractères non autorisés pour les identificateurs ordinaires (les variables) il s'agit souvent des équivalents aux opérateurs mathématiques pour la programmation informatique.

Paramètres effectifs : Il s'agit de valeur fournie à une méthode lors de son appel. Ces valeurs peuvent être constantes ou des variables.

Paramètres formels : On parle aussi de paramètre muet. Il s'agit variables utilisées dans le bloc d'instructions d'une méthode. Ces paramètres permettent de décrire comment les données en entrée de la méthode sont transformées par celui- ci.

Référence : Une référence est une valeur qui permet l'accès en lecteur et/ou écriture à une donnée en mémoire. Une référence n'est pas la donnée elle-même mais seulement une information de localisation, l'adresse de la valeur dans la mémoire.

Types primitifs : les types primitifs (entier, flottant, booléen et caractère) permettent de manipuler directement les données les plus courantes et ont la particularité d'être accessibles directement en mémoire.

Types non primitifs : Les types non primitifs permettent d'associer plusieurs valeurs à une variable. L'emplacement mémoire associé à la variable permet de stocker l'adresse l'emplacement mémoire où se trouvent ses valeurs. La variable et ainsi liée à ses valeurs de manière indirecte. [III.1].

III.2 Présentation du langage java

Java est un langage objet permettent le développement d'applications complètes s'appuyant sur les structures de données classiques (tableaux. Fichiers) et utilisant abondamment l'allocation dynamique de mémoire pour créer des objets en mémoire. La nation de structure, ensemble de données décrivant une entité (un objet en java) est remplacée par la nation de classe un sens de la programmation objet.

Le langage java permet également la définition d'interfaces graphiques facilitant le développement d'applications interactives et permettent à l'utilisateur de " piloter " son programme dans un ordre non imposé par le logiciel. [III.2]

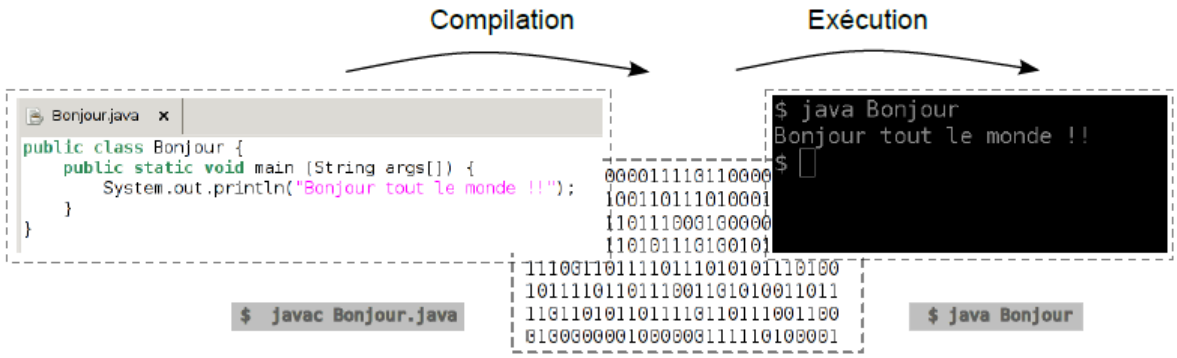
III.2.1 Qu'est-ce qu'un programme ?

L'objectif de la programmation est de créer des logiciels ou des programmes ceux-ci sont constitués d'un ensemble de traitements qui permettent de transformer des données numérique (les entrées) en d'autres données numériques (les sorties). Les données sorties prouvent être affichées sous une forme graphique (avec des fenêtres comme le fond les programmes) ou plus simplement affichées dans une console sous forme de texte.

Que se passe-t-il pour l'ordinateur lorsqu'on exécute un programme ?

Il va lire le fichier exécutable du programme comme une suit de 0 et de 1 (codage binaire) exécuter l'une après l'autre les instructions ainsi codées. Cette suite de 0 et de 1 et appelée langage machine et est directement exécutable par le microprocesseur de l'ordinateur. Or il est très difficile pour un humain de programmer directement en binaire c'est pourquoi on utilise un langage de programmation écrit en langage textuel dans un fichier source (fichier.java). Le fichier source est ensuite compilé en langage binaire

(fichier.class) puis exécuté afin de réaliser les traitements comme montrer dans la figure suivante : [III.2]



Frigure III.1 : démarche d'exécution d'un programme.

III.2.2 Déclaration des données

III.2.2.1 Rôle des déclarations

Un programme manipule des données caractérissées par un nom et un type. Ces données sont stockées en mémoire. Au moment de la traduction du programme, le compilateur affecte à chaque donnée un emplacement en mémoire caractérisé par une adresse et une taille. Il le fait en s'appuyant sur les déclarations figurant au début du programme. Par ailleurs, celles-ci permettent au compilateur de détecter des erreurs de programmation. [III.3]

III.2.2.2 Exécution d'un programme java

Pour mettre en œuvre notre programme, il faut bien sûr le saisir et le sauvegarder dans un fichier suscité. Ce dernier devra impérativement se nommer programme.java. En effet, nous verrons que quel que soit l'environnement concerné, le Code source d'une classe publique doit toujours se trouver dans un fichier portant le même nom et possédant l'extension java. Ensuite, on procède à la compilation de ce fichier source. Rappelons que celle-ci produit non pas du code machine mais un code intermédiaire formé de « bytecodes ». Si la compilation s'est bien déroulée, on obtiendra un fichier portant le même de nom que le fichier source d'extension class, notre cas sera programme. Class. On pourra lancer l'exécution des bytecodes ainsi obtenus par l'intermédiaire de la machine virtuelle java; bien entendu, on pourra exécuter autant de fois qu'on voudra un même programme sans avoir besoin de le recompiler. [III.4]

III.3 Généralité sur langage java :

III.3.1 Les variables

III.3.1.1 Les types primitifs

On distingue 4 catégories de types primitifs (entier, réel, booléen, caractère). L'intervalle de valeurs représentables pour chacun des types peut varier en fonction de l'espace mémoire qu'il occupe.

- **Les entiers :** En java, tous les types permettant de représenter des entiers sont signés. Ainsi sur n bits, on peut coder les entiers de $-(2^{n-1})$ à $(2^{n-1}-1)$. Les valeurs négatives sont encodées en complément à 2. Les différents types d'entier sont les suivants :

Tableau III.1 : types d'entiers et taille.

Nom	Taille	Intervalle représentable
byte	1 octet	$[-128 \dots 127]$ ou $[-2^7 \dots 2^7 - 1]$
short	2 octets	$[-32768 \dots 32767]$ $[-2^{15} \dots 2^{15} - 1]$
int	4 octets	$[-2^{31} \dots 2^{31} - 1]$
long	8 octets	$[-2^{63} \dots 2^{63} - 1]$

Déclaration :

```
int valA =7 ;
int valB=2 ;
 int valC=valA/valB ; //valC contient la valeur 3
 int valD=valA%valB ;// valD contient la valeur 1
```

- **Les boolèens :** un type variable trés utile en informatique et le type booléen, qui prend deux valeurs VRAIS ou FAUX (sur un octet). On peut appliquer des opérateurs logiques (et, ou, non) (&&**,** | **|,** !) à des variables où des expressions booléennes. Le résultat de cette opération également du type booléen.

Tableau III.2 : calcul logique.

a	b	a ou b
Faux	Faux	Faux
Faux	Vrai	Vrai
Vrai	Faux	Vrai
Vrai	Vrai	Vrai

a	b	a et b
Faux	Faux	Faux
Faux	Vrai	Faux
Vrai	Faux	Faux
Vrai	Vrai	Vrai

Déclaration

```
Boolean boolA=TRUE;
Boolean boolB=FALSE ;
Boolean nonA=!boolA//nonA vaut FLASE
Boolean aetB=boolA&& boolB//A ET B vaut FALSE
```

- **Les réels**: la description de codage réeel et décrite dans le cours de numération sur la page du CIPC. En java, il existe deux types de représentation pour les nombres réels: simple et double précision (respectivement les types float (4octets) et double (8octets)).

Déclaration :

```
double réelA=7 ;
double réelB=2 ;
double division =réelA /réelB ;//la variable division contient la valeur
```

3.5

- **Les caractères :** Le type caractère peut correspondre à n'importe quel symbole du clavier (lettre en majuscule ou minuscule. Chiffre. Ponctuation et symboles)

Déclaration :

char caratere=’à ;// la variable caractere contient la valeur a
int a=3 ;// la variable a contient la valeur 3. [III.2]

III.3.2 Les structures de contrôle

Dans un programme, les instructions sont exécutées séquentiellement. Or la puissance et le comportement intelligent d’un programme proviennent essentiellement de la possibilité de s’affranchir de cet ordre pour effecteur des *choix* et des *boucles* (répétitions). Tous les langages disposent d’instructions de contrôle, permettant de vérifier les conditions d’entrées. Elles peuvent être :

- fondées essentiellement sur la notion de branchement (conditionnel ou inconditionnel)
- Ou, au contraire, traduire fidèlement les structures fondamentales de la programmation structurées.

Java dispose d’instructions structurées permettant de réaliser :

- Deux choix : instructions *if...else* et *switch*
- Des boucles (répétitions) d’instructions *do... while*, *while* et *for*

Toutefois, la notion de branchement n’est pas totalement absente de java, comme nous le constatons :

- Il dispose d’instructions de branchement inconditionnel : *break et continue.* [III.5]

III.3.2.1 Tableau contenant les structures de contrôles

On présente sur le tableau suivant les structures de données boucles et les branchements conditionnels : [III.6]

Tableau III.6 : Tableau résumant la syntaxe des structures de contrôles.

Type de structure	**Instruction**	**Syntaxe**	**Description**

Boucles	**For**	For (initialisation;
		condition; modification)
		{ instructions
		}
			Si la condition est vérifiée, on fait le traitement demandé (exécuter les instructions)

	While	while (boolean)
		{
		... // code à exécuter
		dans la boucle
		}
		do {
		...
		} while
		(boolean);
			Tant que la condition est vraie, on exécute le code, ou bien on répète l'opération (instruction), jusqu'à atteindre la condition donnée (condition d'arrêt)

Branchements conditionnes	**If**	if (condition) {
		...
		} else {
		... }
			L'instruction if teste si la condition est vérifiée, on exécute le cas 1 sinon on exécute l'autre cas 2.

	Switch	Switch (variable){ Case 'valeur1': action1; break; Case 'valeur2': action2; break; Case 'valeur3':'valeur4': action3; break; Default: action4; break ;}	L'instruction Switch est utile quand vous devez gérer beaucoup de if / else if / else. Elle a une syntaxe plus courte et est plus appropriée pour ce type.

Chapitre IV : La réalisation

Après avoir terminé la partie analyse, conception et implémentation en langage adéquat, nous allons passer à la partie réalisation (dernière étape du schéma de développement teste) dans laquelle nous allons commencer par les outils utilisés de développement de notre application, ainsi la présentation de l'interface de notre application.

Comme on a utilisé le logiciel « Dia » qui nous a aidé de dessiner les algorigrammes associés aux algorithmes trouvés.

III.1 Logiciel « Dia »

Dia est un logiciel libre de création de diagrammes développé en tant que partie du projet. Dia charge et sauve les diagrammes dans son propre format XML. On a travaillé avec la version (diaw.exe 0.97.2), il est téléchargeable sur le site : http://live.gnome.org/Dia/ puis cliquez sur 'Download'. Téléchargez la dernière version stable en cliquant sur le lien http://dia-installer.de/ .

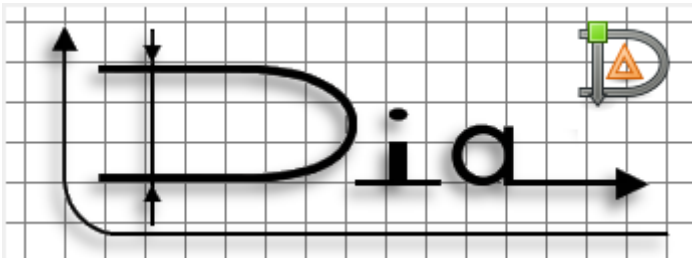


Figure IV.1 : logo du logiciel Dia.

III.2 Les outils de développement de l'application

Afin de réaliser notre simulation, nous avons utilisé un langage de programmation adéquat d'évaluation Java programmation et un environnement de développement eclipse (eclipse Version: 3.4.0) qui nous a assuré un éditeur de code polyvalent pour écrire le corps du programme à l'aide des méthodes et fonctions prédéfinies. Ces fonctionnalités, ainsi qu'un compilateur seront utilisés pour exécuter les codes du programme et détecter les erreurs.

III.2.1 L'environnement de développement éclipse

Éclipse est l'environnement de développement utilisé pour réaliser notre travail. Il est fourni de fonctionnalités qui nous ont simplifiés la tâche de saisir les codes. Comme il contient une bibliothèque de classes et de fonctions prédéfinies rapide à insérer. Éclipse dispose d'un détecteur d'erreur de déclarations (syntaxes) instantanées et des erreurs détectées après l'exécution. Il dispose aussi d'un localisateur d'erreur (lien ou bien un point rouge) qui nous guide directement à la ligne d'erreur.

III.2.2 Choix du langage Java

Le choix de Java revient aux avantages que porte ce langage de programmation. L'un des avantages réside dans le fait que la syntaxe de Java est analogue à celle de C++. Ce qui le rend économique et professionnel. Comme Java possède une importante bibliothèque, Java a été conçu pour que les programmes qui l'utilisent soient fiables sous différents aspects. Sa conception encourage le programmeur à traquer préventivement les éventuels problèmes, à lancer des vérifications dynamiques en cours d'exécution et à éliminer les situations génératrices d'erreurs **[IV.1].**

III.3 Description de l'interface

Nous allons donner un aperçu général de l'interface finale et ses différentes fonctionnalités, et comment allons-nous entrer les données, récupérer le résultat et éviter quelques erreurs ?

III.3.1 L'aperçu de l'interface

Une fois qu'on clique sur l'icône du bureau  pour ouvrir l'exécutable, une fenêtre contenant des champs de calculs va apparaître comme c'est montré dans la figure suivante :

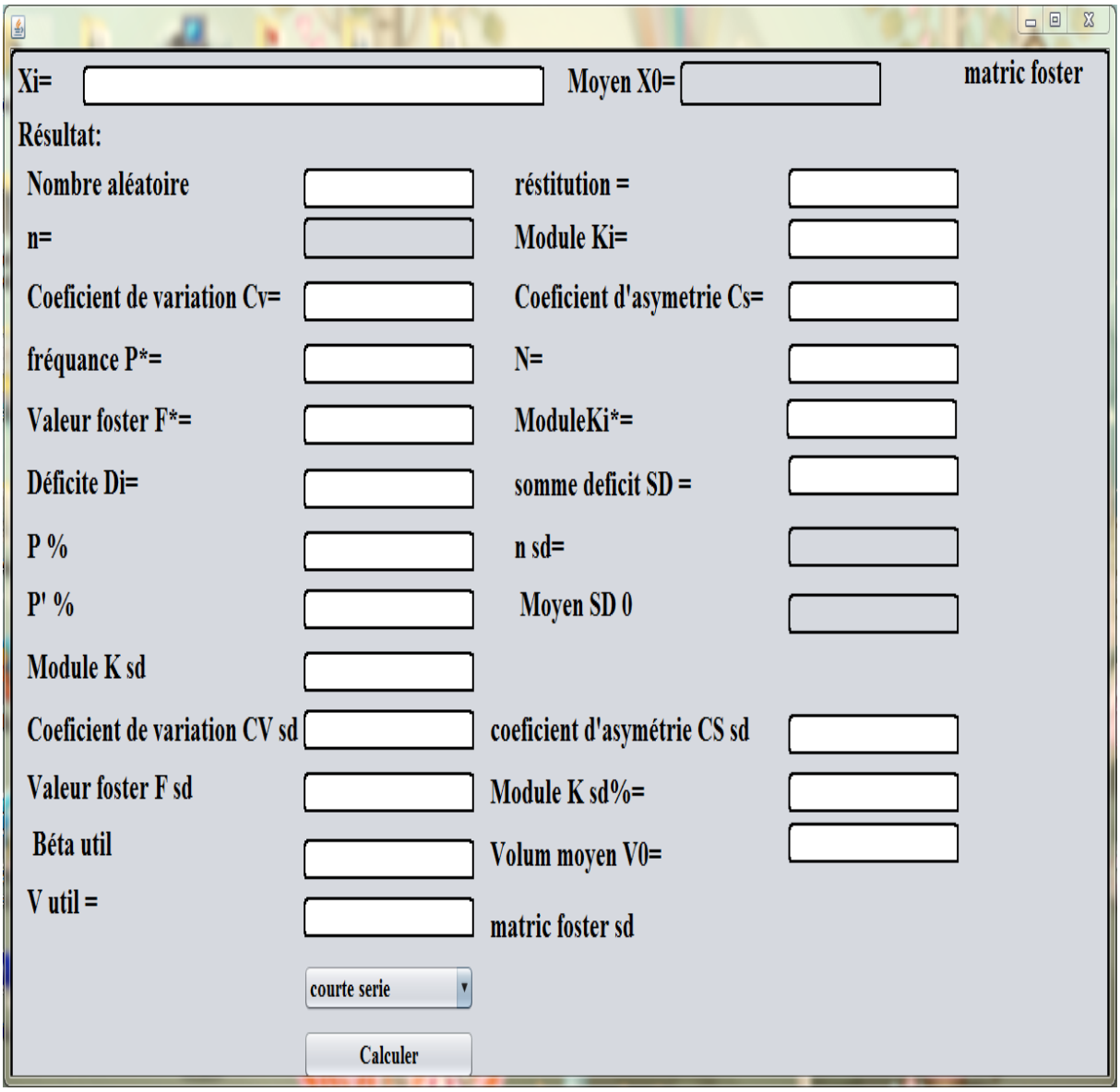


Figure IV.2 : aperçu générale de l'interface.

Avant de commencer la saisie des valeurs dans leurs champs, on doit tout d'abord choisir le mécanisme à suivre (absence des données ou bien courte série) dans la liste de choix comme si montre ci-dessus :

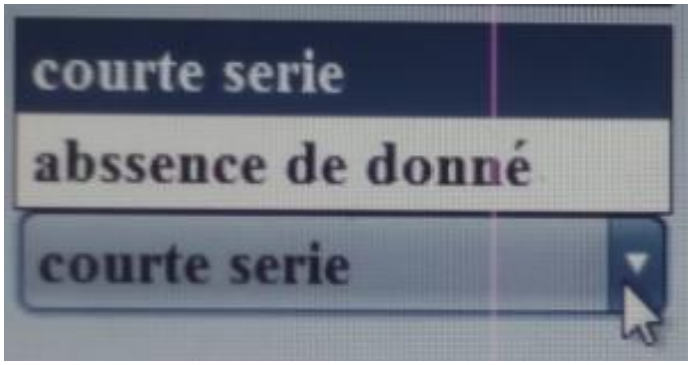


Figure IV.3 : choix de mécanisme de calcul.

III.3.2 Les Entrées / sorties

Les champs de cette application subdivisent en deux catégories : les champs d'entrées des données et les champs de sorties des résultats.

III.3.2.1 Les champs d'entrées

Sont ceux qu'on doit obligatoirement remplir pour assurer la cohérence des calculs et la finesse du résultat. Pour cela, on a imposé la saisie des valeurs d'entrées par renvoi d'erreur, c'est-à-dire une fois un champ d'entrée n'est pas saisi, un message d'erreur sera envoyé en indiquant l'erreur.

III.3.2.1.1 Les erreurs renvoyées

On donne un aperçu de quelques erreurs dans les exemples suivants :

- Le cas où on a oublié d'entrer la série des écoulements annuels, un message d'erreurs sera envoyé, indiquant qu'on devra entrer cette série ou bien on change le choix d'item en l'absence de données. Le message s'affiche comme suit :

Figure IV.4: message d'erreur sur la série Xi

- Une fois le champ de saisie du choix du nombre aléatoire à générer n'est pas donné, une erreur s'affiche comme le montre la figure suivante :

Figure IV.5 : Message d'erreur du nombre de valeur à générer aléatoirement.

- De même si le champ de saisie du choix de fréquence P% n'est pas rempli ou bien le type n'est pas un réel, une erreur s'affiche comme le montre la figure suivante :

Figure IV.6 : Message d'erreur sur la valeur de la fréquence P*.

III.3.2.2 Les champs de sortie

Ce sont des champs réservés pour afficher le résultat du calcul pour chaque valeur. Une fois les champs de saisie (les entrées) sont remplis de valeurs (la série Xi, la valeur de nombre aléatoire à générer, la restitution α, la fréquence complémentaire P% et le volume moyen du barrage V0), on clique sur le bouton « calcul ». Des résultats seront affichés sur les champs de sortie.

III.4 Exemple de pratique

- **Les Entrées :**

Série :(/19.5/15.9/19.5/30.1/21.3/14.2/15.9/21.3/40.7/23/14.2/17.7/54.9/44.3/37.2/65.6/23/23 /14.2/5.3/5.3/15.9/35.4/10.6/23/12.4/17.7/5.3/17.7/10.6/35.4/10.6/)

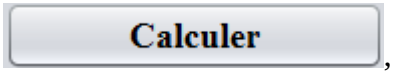
Nombre aléatoire à générer : 300.

La restitution : $\alpha = 0.98$.

La fréquence P% : 80 %

Le volume moyen d'un barrage V0 : 10000000 (m^3).

- **Les sorties :**

Après avoir cliqué sur le bouton calculer 

, le traitement des calculs s'exécute et renvoie les résultats finaux comme c'est montré dans la figure suivante :

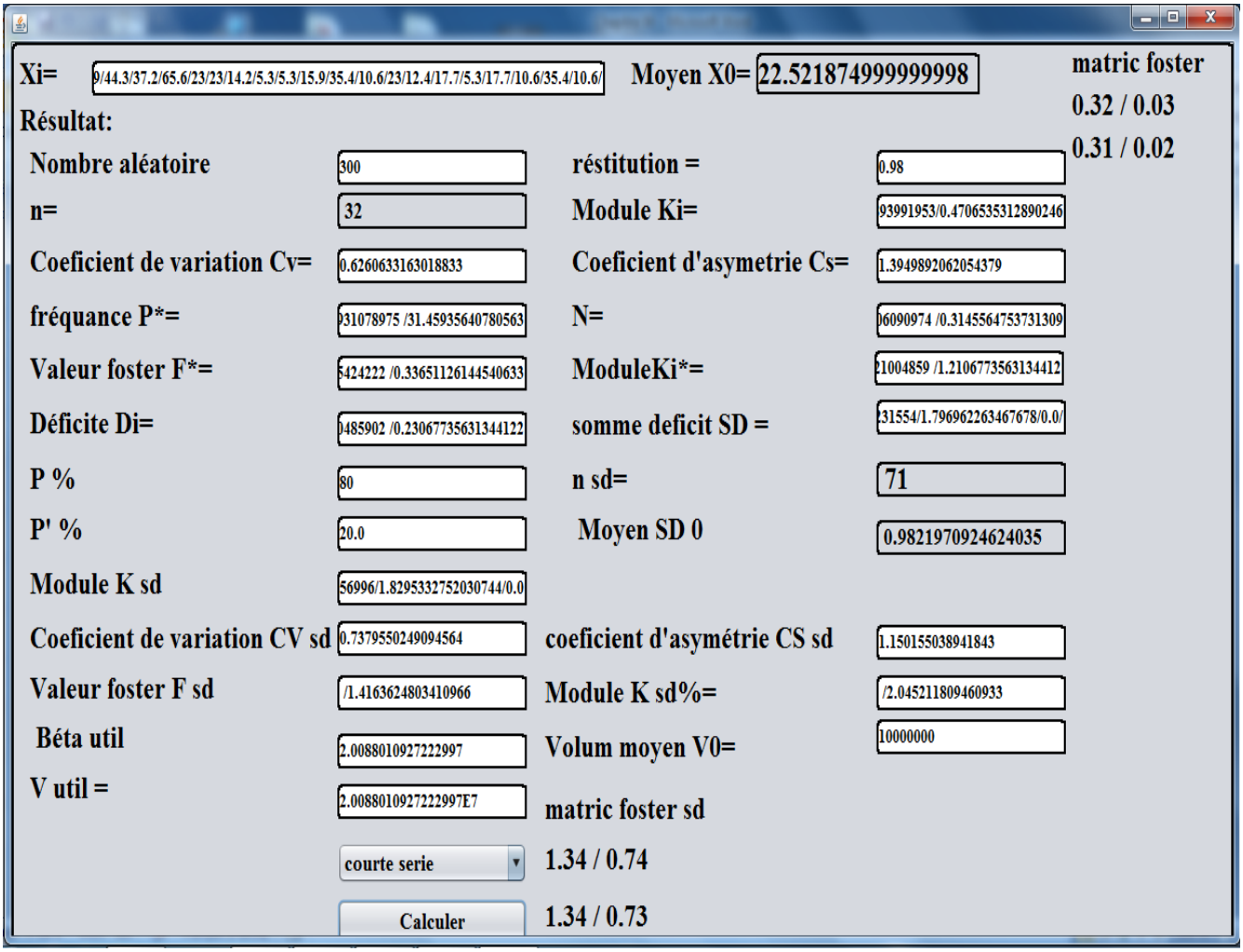


Figure IV.7 : Réaliser un calcul sur l'interface.

Pour les données inscrites (entrées) au-dessus, on trouve le volume utile du barrage égale à :

V utile = 2.0088010927222997E7 = 20088010.927222997 m^3.

III.5 Sauvegarder les résultats

La plupart des résultats sont affichés sous forme d'une série, surtout pour des nombres aléatoires plus importants, le résultat de fréquence P*, Ki, Di,.. etc.., est une série de valeurs.

Comme ce n'est pas pratique de les afficher sur l'interface, alors on a pensé de les enregistrer sous format document 'texte' qu'on trouve dans le répertoire (même dossier contenant l'exécutable). On donne l'aperçu dans la figure suivante :

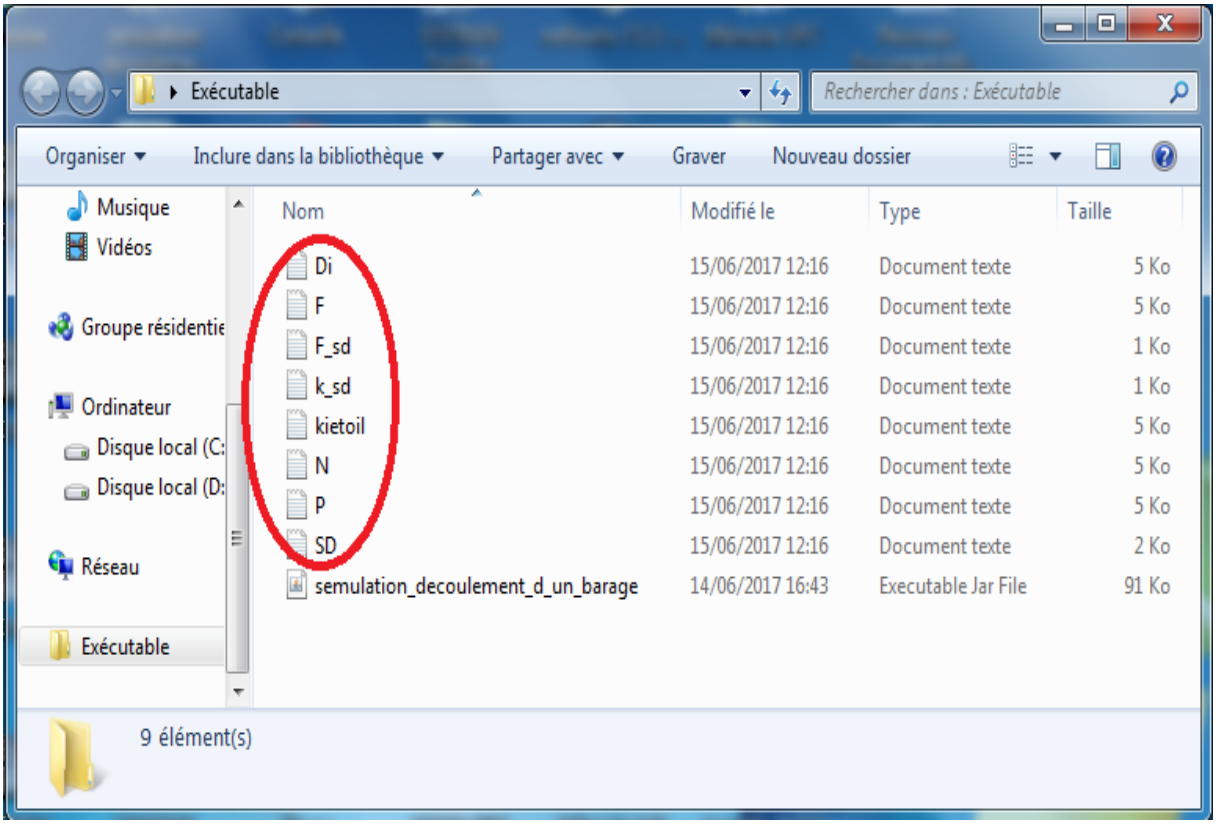


Figure IV.8 : Le répertoire de fichier 'texte' enregistrer.

Les fichiers :

N : reçoit les nombres aléatoires générés.

P : reçoit les fréquences calculées.

Kietoil : reçoit les valeurs du module k.

F : reçoit les valeurs de nombre Foster.

Di : reçoit la valeur de déficit, ki - α.

SD : reçoit les valeurs de somme des déficits.

F_sd : reçoit les valeurs de nombre Foster de la série déficit.

K_sd : reçoit les valeurs du module k de la série déficit.

Par exemple, pour voir les résultats trouvés pour la valeur de la fréquence P*, on ouvre le fichier associe ‘P. texte’. On trouve les résultats affichés et les valeurs sont séparées par un slash ‘/’, comme c’est montré dans la figure suivante :

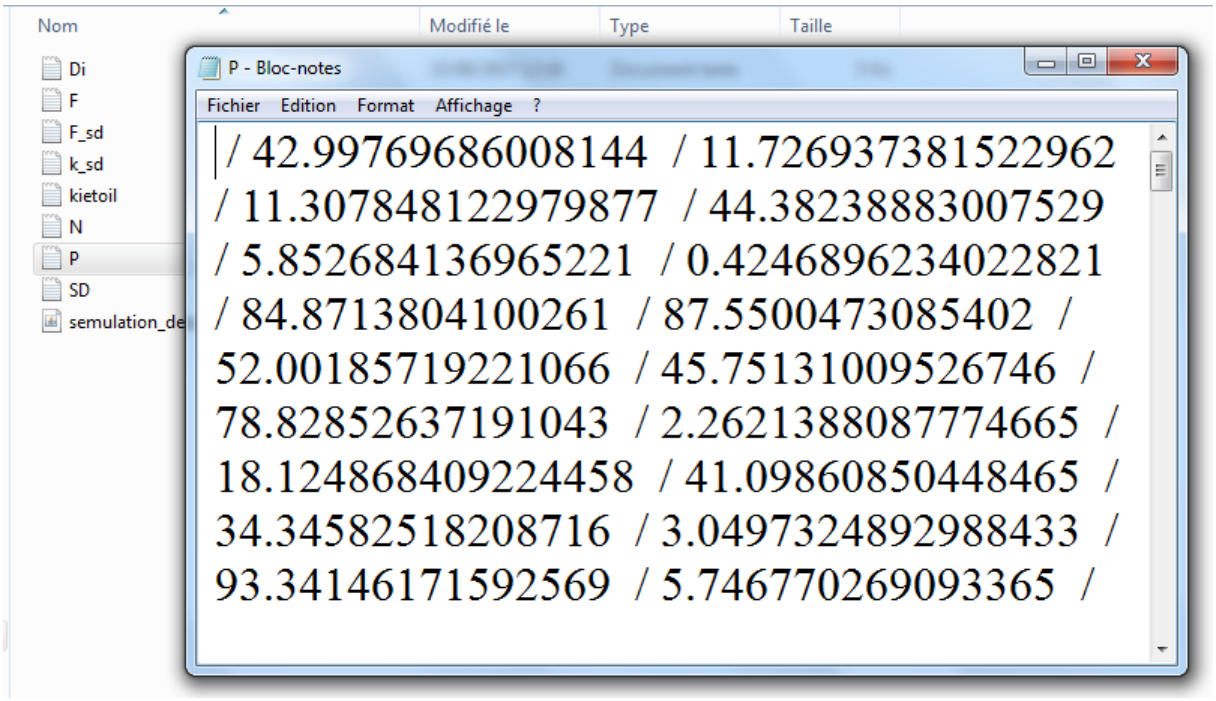


Figure IV.9: L’aperçu de résultats des valeurs de P* trouvées et sauvegardées.

Conclusion :

Ce chapitre a décrit la phase de réalisation où nous avons défini les outils de développement utilisés, ensuite la simulation des données et à la fin une présentation de l’interface pour bien illustrer le projet.

CONCLUSION GENERALE

Au terme de notre travail qui consiste à concevoir et réaliser une application pour les calculs de dimensionnement d'un barrage, donnant ainsi l'opportunité pour les hydrologues d'optimiser et de faciliter leurs calculs.

Pour la réalisation de notre application nous avons utilisé l'outil de développement dédié à la programmation qui est le langage Java, avec l'environnement de travail l'éclipse qui a l'avantage d'être régulièrement mis à jour. Et le logiciel Dia qui nous a permis de dessiner les algorigrammes.

Ce projet nous a permis d'enrichir nos connaissances théoriques et nos compétences dans le domaine de la conception et la programmation et de mettre en application nos connaissances acquises tout au long du cycle d'études en nous mettant dans un environnement pratique.

Estimons que notre travail sera bénéfique pour les étudiants qui feront référence à ce mémoire, puisse répondre aux besoins fixés et satisfaire toutes les personnes qui ont contribué à sa réalisation.

En perspective, il serait souhaitable de projeter ce travail à une échelle plus large, en réseau, en centralisant les données d'une façon cohérente et en accédant d'une façon distribuée, à distance avec des droits d'accès attribués aux spécialistes et aux experts.

Liste Bibliographique

[1] : Ladjel Mahmoud, Contribution à la méthode de régularisation de l'écoulement en des oueds. 2ème Séminaire National sur l'Eau et l'Environnement '2SN2E', 12 et 13 Novembre 2005, Béchar.

[I.1] : Ladjel Mohmoud, 1er Séminaire International sur la Ressource en eau au sahara : Evaluation, Economie et Protection, le 19 et 20 janvier 2011(ouargla)

[I.2] : Jelezniakov Gu. V. et al Hydrologie, hydromètrie et régularisation de l'écoulement.kolos, Moscou, 1984.

[I.3] : Harizi, extrait d'une thèse de magister,16 mai 2017.

[II.1]: BAGHDALI-OURBIH Latifa, livre : Algorithmique « cours et exercices corrigés » ; Fortran77, Méthodes d'analyse numérique, rappel de cours algorithme des méthodes. Septembre 2010.

[II.2] : Thomas Cormen, Charles Leiserson, Ronald Rivest et Clifford Stein, INTRODUCTION À L'ALGORITHMIQUE, Cours et exercices Dunod, Paris, 2004, pour la présente édition ISBN 2 10 003922 9

[II.3]: Anonyme, Support de cours : Université de Bretagne Occidentale IUP Ingénierie Informatique 1ère année.

[II.4]: Anonyme (NFA07 © CNAM 2012), support du cours sur les algorithmes, et Algorithmique et Programmation, 2012.Consultable sur le site électronique suivant: http://deptinfo.cnam.fr/Enseignement/CycleA/APA/nfa07/docs/ Algorithmique et Programmation.pdf

[II.5] : Anonyme, support de cours de l'Université Bretagne Occidentale, IUP ingénierie informatique 1ère année : Algorithme et structure de données, dernière révision le 06 mai 2002.

[II.6] : Anonyme, cours de l'université de Lille 1 Polytech Lille: Algorithmique et programmation, IMA (cours 3c -Récursivité),

[III.1] : Carole Frindel et Céline Robardet, Résumé du cours de Programmation Java : consultable sure le site : http://liris.cnrs.fr/celine.robardet/doc/poly2013.pdf

[III.2] : Michel Divay, LA PROGRAMMATION OBJET EN JAVA, « Dunod, Paris, 2006 ISBN 2 10 049697 2 »

[III.3] : Serge Tahé, - ISTIA - Université d'Angers, APPRENTISSAGE DU LANGAGE JAVA , Septembre 98 - Révision juin 2002

[III.4] : Claude delannoy, PROGRAMMER EN JAVA, Dépôt légal : juin 2006 N° d'éditeur : 7474 Imprimé en France © Groupe Eyrolles, 2006, pour la nouvelle édition, ISBN : 2-212-11988-7

[III.5] : Claude delannoy, Programmer en Java : La 5e édition de cet ouvrage.

[III.6] : Jeans-Michel DOUDOUX, Développons en java

[III.10]: Anonyme article intitulé, Les conditions if / else / else if / switch case en JAVA ; Consultable sous forme électronique sur: http://www.vulgarisation-informatique.com/java-conditions.php

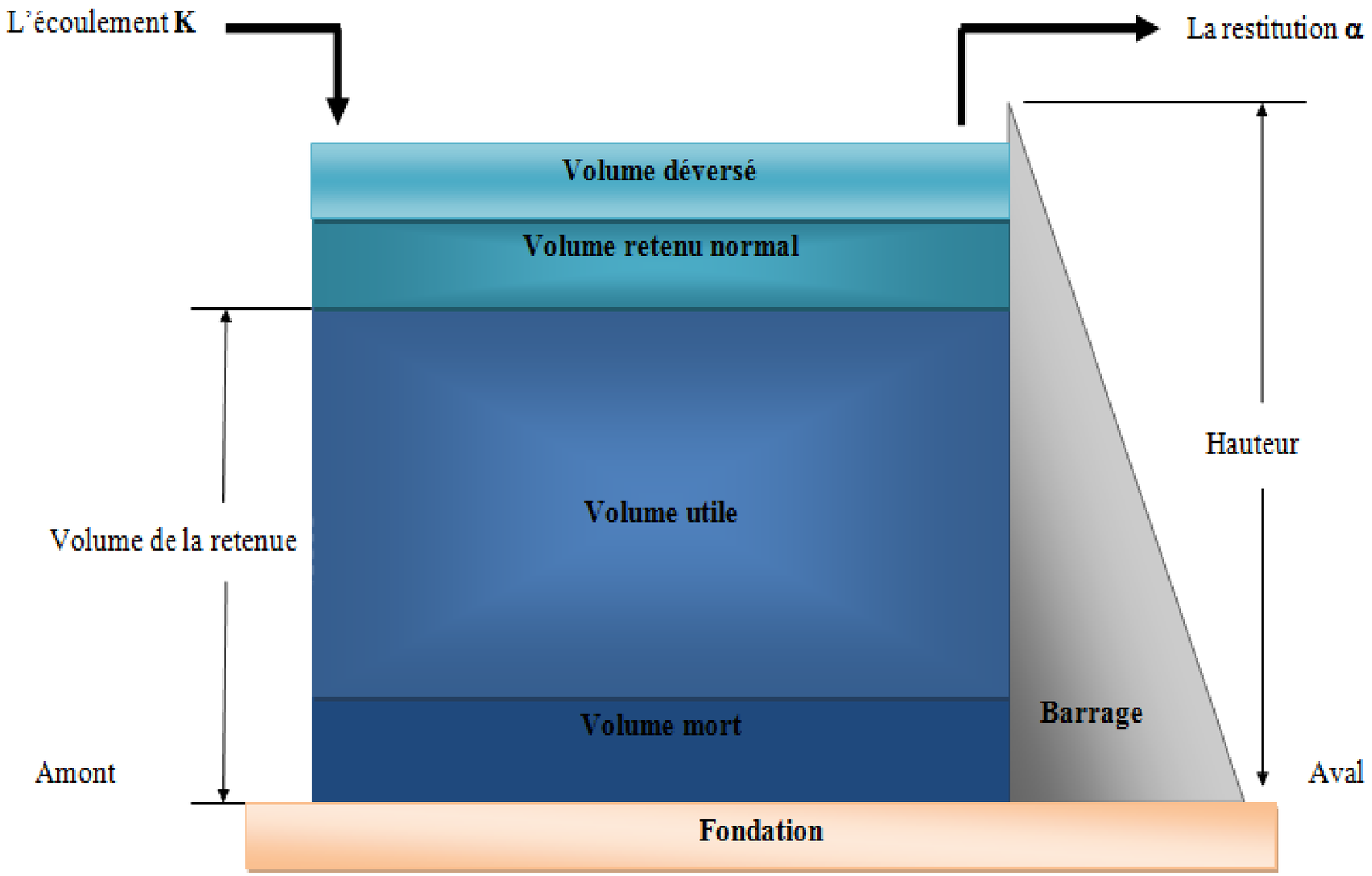
[IV.1]: Anonyme, site N^0 1 des cours et exercices :' JAVA', consultable sur le site électronique: www.Mcours.com

Tableau Foster-Rebkine- LOI BINOMIALE- Pearson III

Cs	0.01	0.1	1	2	5	10	20	25	30	40	50	60	70	75	80	90	95	97	99	99.9	99.99	Cs
0.00	3.72	3.09	2.33	1,76	1.64	1.28	0.84	0.67	0.52	0.25	0.00	-0.25	-0.52	-0.67	-0.84	-1.28	-1.64	-1.88	-2.33	-3.09	-3.67	**0.00**
0.05	3.83	3.16	2.36	1,775	1.65	1.28	0.84	0.66	0.52	0.24	-0.01	-0.26	-0.52	-0.68	-0.84	-1.28	-1.62	-1.86	-2.29	-3.02	-3.57	**0.05**
0.10	3.94	3.23	2.40	1,795	1.67	1.29	0.84	0.66	0.51	0.24	-0.02	-0.27	-0.53	-0.68	-0.85	-1.27	-1.61	-1.84	-2.25	-2.95	-3.48	**0.10**
0.15	4.05	3.31	2.44	1,81	1.68	1.30	0.84	0.66	0.50	0.23	-0.02	-0.28	-0.54	-0.68	-0.85	-1.26	-1.60	-1.82	-2.22	-2.88	-3.39	**0.15**
0.20	4.16	3.38	2.47	1,83	1.70	1.30	0.83	0.65	0.50	0.22	-0.03	-0.28	-0.55	-0.69	-0.85	-1.26	-1.58	-1.79	-2.18	-2.81	-3.30	**0.20**
0.25	4.27	3.45	2.50	1,845	1.71	1.30	0.82	0.64	0.49	0.21	-0.04	-0.29	-0.56	-0.70	-0.85	-1.25	-1.55	-1.77	-2.14	-2.74	-3.20	**0.25**
0.30	4.38	3.52	2.54	1,86	1.72	1.31	0.82	0.64	0.48	0.20	-0.05	-0.30	-0.56	-0.70	-0.85	-1.24	-1.55	-1.75	-2.10	-2.67	-3.11	**0.30**
0.35	4.50	3.59	2.58	1,875	1.73	1.32	0.82	0.64	0.48	0.20	-0.06	-0.30	-0.56	-0.70	-0.85	-1.24	-1.53	-1.72	-2.06	-2.60	-3.02	**0.35**
0.40	4.61	3.66	2.61	1,895	1.75	1.32	0.82	0.63	0.47	0.19	-0.07	-0.31	-0.57	-0.71	-0.85	-1.23	-1.52	-1.70	-2.03	-2.54	-2.94	**0.40**
0.45	4.72	3.74	2.64	1,91	1.76	1.32	0.82	0.62	0.46	0.18	-0.08	-0.32	-0.58	-0.71	-0.85	-1.22	-1.51	-1.68	-2.00	-2.47	-2.84	**0.45**
0.50	4.83	3.81	2.68	1,9	1.77	1.32	0.81	0.62	0.46	0.17	-0.08	-0.33	-0.58	-0.71	-0.85	-1.22	-1.49	-1.66	-1.96	-2.40	-2.75	**0.50**
0.55	4.94	3.88	2.72	1,94	1.78	1.32	0.80	0.62	0.45	0.16	-0.09	-0.34	-0.58	-0.72	-0.85	-1.21	-1.47	-1.64	-1.92	-2.32	-2.65	**0.55**
0.60	5.05	3.96	2.75	1,96	1.80	1.33	0.80	0.61	0.44	0.16	-0.10	-0.34	-0.59	-0.72	-0.85	-1.20	-1.45	-1.61	-1.88	-2.27	-2.58	**0.60**
0.65	5.16	4.03	2.78	1,975	1.81	1.33	0.80	0.60	0.44	0.15	-0.11	-0.35	-0.60	-0.72	-0.85	-1.19	-1.44	-1.59	-1.84	-2.20	-2.49	**0.65**
0.70	5.28	4.10	2.82	1,985	1.82	1.33	0.79	0.59	0.43	0.14	-0.12	-0.36	-0.60	-0.72	-0.85	-1.18	-1.42	-1.57	-1.81	-2.14	-2.41	**0.70**
0.75	5.39	4.17	2.86	1,995	1.83	1.34	0.78	0.58	0.42	0.13	-0.12	-0.36	-0.60	-0.72	-0.86	-1.18	-1.40	-1.54	-1.78	-2.08	-2.33	**0.75**
0.80	5.50	4.24	2.89	2,01	1.84	1.34	0.78	0.58	0.41	0.12	-0.13	-0.37	-0.60	-0.73	-0.86	-1.17	-1.38	-1.52	-1.74	-2.02	-2.25	**0.80**
0.85	5.62	4.31	2.92	2,025	1.85	1.34	0.78	0.58	0.40	0.12	-0.14	-0.38	-0.60	-0.73	-0.86	-1.16	-1.35	-1.49	-1.70	-1.96	-2.17	**0.85**
0.90	5.73	4.38	2.96	2,04	1.86	1.34	0.77	0.57	0.40	0.11	-0.15	-0.38	-0.61	-0.73	-0.85	-1.15	-1.35	-1.47	-1.66	-1.90	-2.09	**0.90**
0.95	5.84	4.46	2.99	2,055	1.87	1.34	0.76	0.56	0.39	0.10	-0.16	-0.38	-0.62	-0.73	-0.85	-1.14	-1.34	-1.44	-1.62	-1.84	-2.01	**0.95**
1.00	5.96	4.53	3.02	2,065	1.88	1.34	0.76	0.55	0.38	0.09	-0.16	-0.39	-0.62	-0.73	-0.85	-1.13	-1.32	-1.42	-1.59	-1.79	-1.94	**1.00**
1.05	6.07	4.60	3.06	2,07	1.88	1.34	0.75	0.54	0.37	0.08	-0.17	-0.40	-0.62	-0.74	-0.85	-1.12	-1.30	-1.40	-1.56	-1.74	-1.88	**1.05**
1.10	6.18	4.67	3.09	2,085	1.89	1.34	0.74	0.54	0.36	0.07	-0.18	-0.41	-0.62	-0.74	-0.85	-1.10	-1.28	-1.31	-1.52	-1.68	-1.80	**1.10**
1.15	6.30	4.74	3.12	2,1	1.90	1.34	0.74	0.53	0.36	0.06	-0.18	-0.42	-0.62	-0.74	-0.84	-1.09	-1.26	-1.36	-1.48	-1.63	-1.73	**1.15**
1.20	6.41	4.81	3.15	2,11	1.91	1.34	0.73	0.52	0.35	0.05	-0.19	-0.42	-0.63	-0.74	-0.84	-1.08	-1.24	-1.33	-1.45	-1.58	-1.66	**1.20**
1.25	6.52	4.88	3.18	2,12	1.92	1.34	0.72	0.52	0.34	0.04	-0.20	-0.42	-0.63	-0.74	-0.84	-1.07	-1.22	-1.30	-1.42	-1.53	-1.60	**1.25**
1.30	6.64	4.95	3.21	2,13	1.92	1.34	0.72	0.51	0.33	0.04	-0.21	-0.43	-0.63	-0.74	-0.74	-1.06	-1.20	-1.28	-1.38	-1.48	-1.55	**1.30**
1.35	6.76	5.02	3.24	2,145	1.93	1.34	0.72	0.50	0.32	0.03	-0.22	-0.44	-0.64	-0.74	-0.84	-1.05	-1.18	-1.26	-1.35	-1.44	-1.50	**1.35**
1.40	6.87	5.09	3.27	2,155	1.94	1.34	0.71	0.49	0.31	0.02	-0.22	-0.44	-0.64	-0.73	-0.83	-1.04	-1.17	-1.23	-1.32	-1.39	-1.44	**1.40**
1.45	6.98	5.16	3.30	2,16	1.94	1.34	0.70	0.48	0.30	0.01	-0.23	-0.44	-0.64	-0.73	-0.82	-1.03	-1.15	-1.21	-1.29	-1.35	-1.40	**1.45**
1.50	7.09	5.23	3.33	2,17	1.95	1.33	0.69	0.47	0.30	0.00	-0.24	-0.45	-0.64	-0.73	-0.82	-1.02	-1.13	-1.19	-1.26	-1.31	-1.35	**1.50**
1.55	7.20	5.30	3.36	2,18	1.96	1.33	0.69	0.46	0.29	-0.01	-0.24	-0.46	-0.64	-0.73	-0.82	-1.00	-1.12	-1.16	-1.23	-1.28	-1.32	**1.55**
1.60	7.31	5.37	3.39	2,19	1.96	1.33	0.68	0.46	0.28	-0.02	-0.25	-0.46	-0.64	-0.73	-0.81	-0.99	-1.10	-1.14	-1.20	-1.24	-1.27	**1.60**
1.65	7.42	5.44	3.42	2,195	1.96	1.32	0.67	0.45	0.27	-0.02	-0.26	-0.46	-0.64	-0.72	-0.81	-0.98	-1.08	-1.12	-1.17	-1.20	-1.22	**1.65**
1.70	7.54	5.50	3.44	2,205	1.97	1.32	0.66	0.44	0.26	-0.03	-0.27	-0.47	-0.64	-0.72	-0.81	-0.97	-1.06	-1.10	-1.14	-1.17	-1.196	**1.70**
1.75	7.65	5.57	3.47	2,215	1.98	1.32	0.65	0.43	0.25	-0.04	-0.28	-0.48	-0.64	-0.72	-0.80	-0.96	-1.04	-1.08	-1.12	-1.14	-1.162	**1.75**
1.80	7.76	5.64	3.50	2,22	1.98	1.32	0.64	0.42	0.24	-0.05	-0.28	-0.48	-0.64	-0.72	-0.80	-0.94	-1.02	-1.06	-1.09	-1.11	-1.129	**1.80**
1.85	7.78	5.70	3.52	2,23	1.98	1.32	0.64	0.41	0.23	-0.06	-0.28	-0.48	-0.64	-0.72	-0.80	-0.93	-1.00	-1.04	-1.06	-1.08	-1.096	**1.85**
1.90	7.98	5.77	3.55	2,24	1.99	1.31	0.63	0.40	0.22	-0.07	-0.29	-0.48	-0.64	-0.72	-0.79	-0.92	-0.98	-1.01	-1.04	-1.05	-1.063	**1.90**
1.95	8.10	5.84	3.58	2,25	2.00	1.30	0.62	0.40	0.21	-0.08	-0.30	-0.48	-0.64	-0.72	-0.78	-0.91	-0.96	-0.99	-1.02	-1.02	-1.041	**1.95**

2.00	8.21	5.91	3.60	2,255	2.00	1.30	0.61	0.39	0.20	-0.08	-0.31	-0.49	-0.64	-0.71	-0.78	-0.90	-0.95	-0.97	-0.99	-1.00	-1.008	**2.00**
2.10	8.33	6.04	3.65	2,27	2.01	1.29	0.59	0.37	0.18	-0.10	-0.32	-0.50	-0.64	-0.70	-0.76	-0.87	-0.914	-0.93	-0.945	-0.953	-0.957	**2.10**
2.20	8.63	6.14	3.68	2,28	2.02	1.27	0.57	0.35	0.16	-0.12	-0.33	-0.50	-0.64	-0.69	-0.75	-0.842	-0.882	-0.895	-0.905	-0.91	-0.906	**2.20**
2.30	8.75	6.26	3.73	2,29	2.01	1.26	0.55	0.32	0.14	-0.13	-0.34	-0.50	-0.63	-0.68	-0.74	-0.815	-0.85	-0.86	-0.867	-0.87	-0.868	**2.30**
2.40	9.00	6.37	3.78	2,3	2.00	1.25	0.52	0.29	0.12	-0.14	-0.35	-0.51	-0.62	-0.67	-0.72	-0.792	-0.82	-0.826	-0.83	-0.834	-0.831	**2.40**
2.50	9.16	6.5	3.82	2,31	2.00	1.23	0.50	0.27	0.10	-0.16	-0.36	-0.51	-0.62	-0.66	-0.71	-0.768	-0.79	-0.795	-0.80	-0.8	-0.798	**2.50**
2.60	9.39	6.54	3.86	2,315	2.00	1.21	0.48	0.25	0.085	-0.17	-0.37	-0.51	-0.61	-0.66	-0.70	-0.746	-0.764	-0.766	-0.77	-0.77	-0.765	**2.60**
2.70	9.57	6.75	3.92	2,32	2.00	1.19	0.46	0.24	0.07	-0.18	-0.38	-0.51	-0.61	-0.65	-0.68	-0.724	-0.736	-0.739	-0.74	-0.74	-0.737	**2.70**
2.80	9.77	6.86	3.96	2,325	2.00	1.18	0.44	0.22	0.057	-0.20	-0.39	-0.51	-0.60	-0.64	-0.67	-0.703	-0.711	-0.714	-0.715	-0.715	-0.709	**2.80**
2.90	9.96	7.00	4.01	2,325	1.99	1.15	0.41	0.20	0.041	-0.21	-0.39	-0.51	-0.60	-0.63	-0.65	-0.681	-0.689	-0.69	-0.69	-0.69	-0.685	**2.90**
3.00	10.16	7.10	4.05	2,315	1.97	1.13	0.39	0.19	0.027	-0.22	-0.40	-0.51	-0.59	-0.62	-0.64	-0.661	-0.665	-0.666	-0.666	-0.666	-0.661	**3.00**
3.10	10.36	7.23	4.09	2,315	1.97	1.11	0.37	0.17	0.01	-0.23	-0.40	-0.51	-0.58	-0.60	-0.62	-0.641	-0.645	-0.646	-0.646	-0.646	-0.643	**3.10**
3.20	10.55	7.35	4.11	2,31	1.96	1.09	0.35	0.15	-0.006	-0.25	-0.41	-0.51	-0.57	-0.59	-0.61	-0.621	-0.625	-0.625	-0.625	-0.625	-0.625	**3.20**
3.30	10.75	7.44	4.15	2,305	1.95	1.08	0.33	0.13	-0.022	-0.26	-0.41	-0.50	-0.56	-0.58	-0.59	-0.605	-0.606	-0.606	-0.606	-0.606	-0.606	**3.30**
3.40	10.9	7.54	4.18	2,3	1.94	1.06	0.31	0.11	-0.036	-0.27	-0.41	-0.50	-0.55	-0.57	-0.58	-0.586	-0.587	-0.588	-0.588	-0.588	-0.588	**3.40**
3.50	11.14	7.64	4.21	2,295	1.93	1.04	0.29	0.085	-0.049	-0.28	-0.41	-0.50	-0.54	-0.55	-0.56	-0.57	-0.751	-0.571	-0.571	-0.571	-0.570	**3.50**
3.60	11.3	7.72	4.24	2,295	1.93	1.03	0.28	0.064	-0.072	-0.28	-0.42	-0.49	-0.54	-0.54	-0.55	-0.555	-0.556	-0.556	-0.556	-0.552	-0.552	**3.60**
3.70	11.52	7.86	4.26	2,285	1.91	1.01	0.26	0.048	-0.084	-0.29	-0.42	-0.48	-0.52	-0.53	-0.54	-0.541	-0.541	-0.541	-0.541	-0.541	-0.541	**3.70**
3.80	11.67	7.97	4.29	2,275	1.90	1.00	0.24	0.032	-0.095	-0.30	-0.42	-0.48	-0.51	-0.52	-0.52	-0.526	-0.526	-0.526	-0.526	-0.527	-0.527	**3.80**
3.90	11.91	8.08	4.32	2,275	1.90	0.98	0.23	0.02	-0.11	-0.30	-0.41	-0.47	-0.50	-0.51	-0.51	-0.513	-0.513	-0.513	-0.513	-0.513	-0.513	**3.90**
4.00	12.02	8.17	4.34	2,275	1.90	0.96	0.21	0.01	-0.12	-0.31	-0.41	-0.46	-0.49	-0.49	-0.50	-0.500	-0.500	-0.500	-0.500	-0.500	-0.500	**4.00**
4.10	12.28	8.29	4.36	2,27	1.89	0.95	0.20	0.00	-0.13	-0.41	-0.46	-0.48	-0.484	-0.486	-0.487	-0.487	-0.487	-0.487	-0.487	-0.487	-0.487	**4.10**
4.20	12.40	8.38	4.39	2,26	1.88	0.93	0.19	0.01	-0.13	-0.31	-0.41	-0.45	-0.47	-0.473	-0.475	-0.476	-0.476	-0.476	-0.477	-0.477	-0.477	**4.20**
4.30	12.65	8.49	4.40	2,255	1.87	0.92	0.17	0.021	-0.14	-0.32	-0.40	-0.44	-0.46	-0.462	-0.462	-0.465	-0.465	-0.465	-0.465	-0.465	-0.465	**4.30**
4.40	12.76	8.60	4.42	2,245	1.86	0.91	0.15	0.032	-0.15	-0.32	-0.40	-0.44	-0.451	-0.454	-0.455	-0.455	-0.455	-0.455	-0.455	-0.455	-0.455	**4.40**
4.50	13.02	8.69	4.44	2,235	1.85	0.89	0.14	0.042	-0.16	-0.32	-0.40	-0.43	-0.441	-0.444	-0.445	-0.445	-0.445	-0.445	-0.445	-0.445	-0.445	**4.50**
4.60	13.12	8.79	4.46	2,23	1.84	0.87	0.13	0.052	-0.17	-0.32	-0.40	-0.42	-0.432	-0.434	-0.435	-0.435	-0.435	-0.435	-0.435	-0.435	-0.435	**4.60**
4.70	13.37	8.89	4.49	2,22	1.83	0.85	0.11	0.064	-0.18	-0.32	-0.40	-0.42	-0.424	-0.425	-0.425	-0.426	-0.426	-0.426	-0.426	-0.426	-0.426	**4.70**
4.80	13.51	8.96	4.50	2,205	1.81	0.82	0.10	0.075	-0.19	-0.32	-0.41	-0.416	-0.416	-0.416	-0.416	-0.416	-0.416	-0.416	-0.417	-0.417	-0.417	**4.80**
4.90	13.73	9.04	4.51	2,2	1.80	0.80	0.084	0.087	-0.19	-0.33	-0.386	-0.401	-0.407	-0.408	-0.409	-0.409	-0.409	-0.409	-0.409	-0.409	-0.409	**4.90**
5.00	13.87	9.12	4.54	2,19	1.78	0.78	0.068	0.099	-0.20	-0.33	-0.38	-0.395	-0.399	-0.400	-0.400	-0.400	-0.400	-0.400	-0.400	-0.400	-0.400	**5.00**
5.10	14.08	9.20	4.57	2,18	1.76	0.76	0.051	0.11	-0.21	-0.33	-0.376	-0.388	-0.391	-0.392	-0.392	-0.392	-0.392	-0.392	-0.392	-0.392	-0.392	**5.10**
5.20	14.25	9.27	4.59	2,17	1.74	0.73	0.035	-0.12	-0.21	-0.33	-0.37	-0.38	-0.384	-0.385	-0.385	-0.385	-0.385	-0.385	-0.385	-0.385	-0.385	**5.20**
5.40	14.60	9.42	4.62	2,15	1.70	0.67	0.02	-0.10	-0.21	-0.33	-0.37	-0.37	-0.37	-0.37	-0.37	-0.370	-0.370	-0.370	-0.370	-0.370	-0.370	**5.40**
5.60	14.95	9.59	4.65	2,135	1.67	0.62	0.00	-0.12	-0.21	-0.30	-0.36	-0.36	-0.36	-0.36	-0.36	-0.360	-0.360	-0.360	-0.360	-0.360	-0.360	**5.60**
5.80	15.32	9.70	4.70	2,12	1.64	0.57	-0.02	-0.14	-0.21	-0.30	-0.35	-0.35	-0.35	-0.35	-0.35	-0.350	-0.350	-0.350	-0.350	-0.350	-0.350	**5.80**
6.00	15.67	9.84	4.70	2,1	1.60	0.51	-0.05	-0.15	-0.21	-0.30	-0.34	-0.34	-0.34	-0.34	-0.34	-0.340	-0.340	-0.340	-0.340	-0.340	-0.340	**6.00**
6.20	16.04	9.95	4.71	2,08	1.56	0.47	-0.05	-0.15	-0.21	-0.30	-0.34	-0.34	-0.34	-0.34	-0.34	-0.340	-0.340	-0.340	-0.340	-0.340	-0.340	**6.20**

Schéma explicatif de différents niveaux d'un barrage



Résumé :

L'algorithmique représente, pour un informaticien, le premier pas à faire pour entrer dans le monde de l'automatisme des traitements. Elle est la base de la méthode utilisée dans le cycle de développement des programmes informatiques. Proposer une solution informatique à un problème donné, grâce à un programme informatique, est le but final de la programmation.

La méthode de modélisation statistique appelée « La méthode de Monte-Carlo», désigne une famille de méthodes algorithmiques visant à calculer une valeur numérique approchée en utilisant des procédés aléatoires c'est-à-dire des techniques probabilistes. Elle est utilisée dans la création du modèle statistique de l'écoulement fluvial d'un barrage. Cette méthode est présentée sous forme de fonctions mathématiques d'une structure logique, simple à informatiser et analyser en utilisant les bases de l'algorithme et d'algorigramme que nous avons schématisé à l'aide d'un logiciel Dia. Et pour la programmation, nous avons utilisé le langage Java programmation sur l'environnement eclipse.

L'objet majeur de ce projet est de réaliser une application sous forme d'un calculateur qui réalise des calculs longs et complexes, lesquels, auparavant, sont faits manuellement par les hydrologues dans leurs activités basées sur des séries statistiques annuelles de l'écoulement fluvial.

Mot clé : Algorithme, Monte-Carlo, Dia, Java, Eclipse.

Abstract:

The algorithmic one represents, for a data processing specialist, the first step to insert in the world of the automatism of the treatments. It is the base of the method used in the development cycle of the computer programs. To propose an IT solution with a given problem, thanks to a computer program, is the final goal of the programming.

The method of statistical modeling called "the method of Monte Carlo", designates a family of algorithmic methods aiming at calculating an approximate numerical value by using random processes i.e. probabilistic techniques, consists in the creation of the statistical model of the river flow of a stopping, these methods are presented like mathematical functions of a logical structure, simple to computerize and analyze it by using the bases of the algorithm and algorigramme which one schematized using a Dia software. And for the programming one programmed with Java programming on the environment eclipses.

The major object of this project and to carry out an application in the form of a calculator which makes long and complex calculations, made by the hydrologists in their statistical experiments.

Keyword: Algorithm, Monte Carlo, Dia, Java, Eclipse.