Institutional Repository - Research Portal
Dépôt Institutionnel - Portail de la Recherche
researchportal.unamur.be
University of Namur

THESIS / THÈSE

MASTER EN SCIENCES INFORMATIQUES

Chaos, fractales et spectre de Fourier
Les nouveaux outils d'analyse du rythme cardiaque

LEGRAND, Denis

Award date:
1993

Awarding institution:
Universite de Namur

Link to publication

General rights
Copyright and moral rights for the publications made accessible in the public portal are retained by the authors and/or other copyright owners and it is a condition of accessing publications that users recognise and abide by the legal requirements associated with these rights.

- Users may download and print one copy of any publication from the public portal for the purpose of private study or research.
- You may not further distribute the material or use it for any profit-making activity or commercial gain
- You may freely distribute the URL identifying the publication in the public portal ?

Take down policy
If you believe that this document breaches copyright please contact us providing details, and we will remove access to the work immediately and investigate your claim.

Download date: 02. May. 2026

Chaos, fractales et spectre de Fourier : Les nouveaux outils d'analyse du rythme cardiaque.

Legrand Denis Septembre 1993

Mémoire de Licence et Maîtrise en Informatique présenté par Legrand Denis en septembre 1993

Promoteur: Professeur Jean Fichefet.

Remerciements

Je remercie Monsieur Fichefet qui a accepté d'être mon promoteur et qui m'a proposé ce sujet intéressant.

Je remercie le docteur Fauche ainsi que les infirmières du service de cardiologie de la clinique Sainte-Camille pour leur dévouement et leur gentillesse.

Je remercie Jean-Benoît Pirot et Frédéric Laurent pour leurs renseignements concernant les logiciels utilisés pour la réalisation de ce projet.

En résumé, je tiens à remercier toutes les personnes qui se sont dévouées pour m'aider à réaliser ce projet.

Table des matières :

Chaos, fractales et spectre de Fourier : Les nouveaux outils d'analyse du rythme cardiaque

Résumé : A l'heure actuelle, certains physiologistes remettent en question un vieux paradigme de la médecine traditionnelle : "*une personne saine possède un coeur qui bat régulièrement*". En effet, d'après diverseses recherches, il semblerait que le rythme cardiaque d'un individu en bonne santé serait plutôt chaotique afin de s'adapter aux variations de l'environnement. Ces conclusions sont tirées d'une analyse du rythme cardiaque à l'aide de la théorie des systèmes dynamiques non linéaires et de la géométrie fractale. Elles sont confirmées par une analyse spectrale du signal émis par les battements du coeur.

L'optique de ce travail est de fournir un logiciel d'analyse de rythme cardiaque d'une personne au repos au moyen de la dynamique chaotique, de la géométrie fractale et du spectre de Fourier d'un signal discrétisé. Un développement théorique de ces divers outils est donné dans les deux premières sections, tandis que la troisième section décrit les étapes successives qui ont conduit au logiciel final.

Abstract : By now, physiologists question an old paradigm of the traditionnal medicine : " *a healthy person has a heart that beats periodically*". Indeed, several researchs seem to show that the cardiac rhythm of a healthy person would be rather chaotic in order to adapt itself to the variations of the environment. This conclusions are the results of a cardiac rhythm analysis using non-linear dynamical system theory and fractal geometry. The spectral analysis of the signal given by heart beatings leads to the same results.

The purpose of this work is to provide a software that gives cardiac rhythm analysis of a person at rest. The software uses chaotic dynamic, fractal geometry and Fourier spectrum of a discretely sampled signal. A theoretical development about these tools is given in the first two sections while a third section gives the steps that lead to the final software.

Chapitre 0 : Introduction

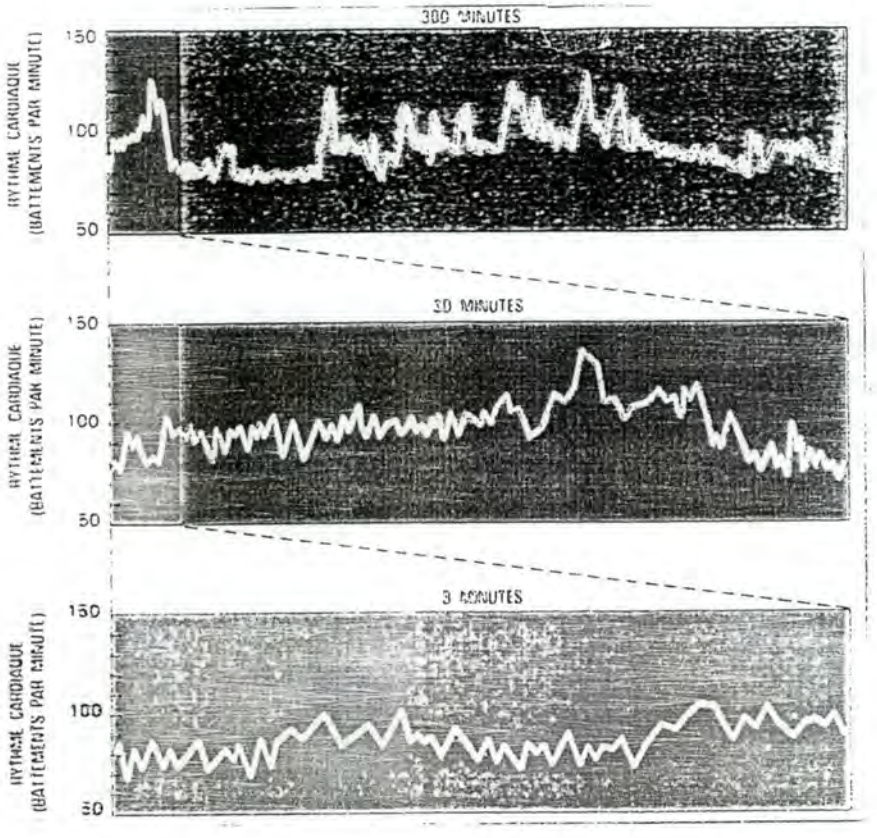
Un médecin contrôlant le rythme cardiaque observe que celui-ci change parfois considérablement de minute en minute ou d'heure en heure. Il pressent que l'intervalle entre deux battements varie de façon aléatoire. De ces observations, physiologistes et médecins prennent conscience des possibilités ouvertes par la dynamique chaotique et les architectures fractales pour leurs recherches. Les résultats de leurs recherches remettent en question de vieux dogmes de la médecine. En effet, la sagesse médicale classique attribuait la maladie et le vieillissement à des forces qui déréglaient un système ordonné et automatique: elles perturbaient le mécanisme en introduisant des effets aléatoires qui modifient les rythmes périodiques normaux du corps. Or, des études ont permis de découvrir que le coeur jeune et sain peut avoir un comportement plus fortement chaotique qu'un coeur vieux et malade. Paradoxalement, un comportement plus régulier accompagne parfois le vieillissement et la maladie!! De tels résultats ont amené les chercheurs à analyser la flexibilité et la résistance de structures fractales irrégulières et l'adaptabilité et la robustesse de systèmes manifestant un comportement apparemment chaotique.

Chaos et fractales sont associés à la dynamique non linéaire des systèmes dont la réponse n'est pas proportionnelle aux stimuli. Les systèmes non linéaires déterministes peuvent se comporter de façon instable et chaotique. Toutefois signalons que le chaos déterministe résultant de tels systèmes n'est pas le chaos au sens ordinaire, c'est-à-dire la désorganisation complète, l'aléatoire. Il traite d'une forme spécifique de hasard, associée à la géométrie fractale.

Certains organes de l'être humain présentent des structures rappelant les fractales, c'est-à-dire des figures de tailles et d'orientations diverses, mais qui sont toutes de formes similaires. Parmi ces organes, nous pouvons citer le système de neurones, les réseaux de vaisseaux sanguins . La fractale la plus étudiée de l'organisme est le système de tubules véhiculant les gaz qui entrent et sortent des poumons.

Les systèmes de type fractal jouent un rôle vital dans la dynamique mécanique et électrique du coeur, notamment le réseau des veines et artères coronaires qui conduisent le sang entrant et sortant du muscle cardiaque. La géométrie fractale permet d'expliquer les anomalies de la circulation sanguine dans le coeur sain qui peuvent entraîner un infarctus du myocarde (mieux connu sous le terme *crise cardiaque*). L'architecture fractale se manifeste aussi dans la structure arborescente de certains organes du coeur, notamment dans le faisceau de Hiss qui conduit les impulsions électriques des oreillettes aux muscles cardiaques des ventricules. La structure fractale d'arborescence ou de plis accroît considérablement la surface utilisable pour l'absorption (comme dans l'intestin), la distribution ou la collecte (par les vaisseaux sanguins, les canaux biliaires ou les bronches), ou le transport d'information (par les nerfs). En partie grâce à sa redondance et ses irrégularités, elle est robuste et résistante aux lésions.

L'application de la théorie du chaos aux systèmes physiologiques commença au début des années 80, les chercheurs désirant associer le chaos au vieillissement ou à la maladie. Cette tendance a été incitée par la tradition médicale. La représentation du rythme cardiaque normal comme une pseudo-sinusoïde régulière a été motivée par l'utilisation du stéthoscope. En effet, pour un individu au repos, l'intensité du pouls et l'intervalle entre deux battements semblent approximativement constants. Une analyse plus fine révèle que les rythmes cardiaques des individus sains varient fortement, même lorsque ceux-ci sont au repos. Au cours d'une journée, le rythme cardiaque de jeunes adultes en bonne santé peut osciller entre 40 et 180 pulsations par minute. Ces fluctuations du rythme cardiaque étaient interprétées en termes d'homéostasie: les systèmes physiologiques visent à réduire la variabilité et à maintenir la constance des diverses fonctions internes. Le principe d'homéostasie suggérait que les variations du rythme cardiaque étaient de simples réactions transitoires à un environnement fluctuant. On en déduisait que, sous l'influence de l'âge ou de la maladie, le corps était moins apte à maintenir un rythme cardiaque constant au repos. Les chercheurs ont remarqué qu'il n'en est rien; les mesures des variations de l'intervalle entre les battements cardiaques portés sur un graphique journalier font apparaître une suite de points déchiquetée, irrégulière et, à première vue, complètement aléatoire. Toutefois, les fluctuations du rythme des battements de coeur à différentes échelles de temps ont une similitude interne, à l'image de la similitude d'une fractale géométrique.



Grâce à cette découverte, les physiologistes suggèrent que le mécanisme qui contrôle le rythme cardiaque est intrinsèquement chaotique. Le coeur peut varier considérablement, même en l'absence de toute fluctuation des stimuli externes, au lieu de revenir à un état stationnaire homéostatique.

Pour déterminer les propriétés chaotiques ou périodiques des variations de l'intervalle entre les battements du coeur, on examine le ***spectre de Fourier*** du rythme cardiaque. Celui-ci révèle les composantes périodiques: Le spectre d'un rythme constant, d'une pulsation par seconde, présente un pic aigu à la fréquence de un battement par seconde, alors que le spectre d'un rythme cardiaque chaotique aura, soit de larges pics, soit aucun pic nettement défini.

Un autre outil d'analyse de la dynamique d'un système non linéaire complexe est sa représentation dans l'***espace des phases***. Cette technique fait ressortir les valeurs des variables indépendantes qui changent avec le temps. Cependant, vu la complexité des systèmes, on se limite à représenter les valeurs d'une variable en fonction du temps, avec un pas temporel fixe. Une telle représentation fournit une suite de points à des instants successifs esquissant une courbe qui décrit l'évolution du système. Pour identifier la nature de la dynamique du système (chaotique ou périodique), nous déterminons les trajectoires pour de nombreuses conditions initiales et nous recherchons un attracteur, une région de l'espace des phases qui "attire" toutes les trajectoires. La complexité géométrique de cet attracteur permet de définir le type du système dynamique étudié.

Ces deux outils d'analyse des sytèmes dynamiques non linéaires existent indépendamment, dans des théories mathématiques différentes. L'objectif de ce mémoire est de fournir un logiciel qui réunit le spectre de Fourier et la représentation dans l'espace des phases afin d'étudier le rythme cardiaque d'un individu au repos. Dans le but de rendre possible la réalisation d'un tel logiciel, il est nécessaire de maîtriser les théories de ces deux outils. Suivant cet objectif, le premier chapitre est consacré au développement théorique relatif à la transformée de Fourier conduisant au spectre de Fourier. Le premier chapitre décrit aussi un algorithme de calcul de la transformée de Fourier d'une fonction discrétisée. Le second chapitre, quant à lui, est consacré au développement théorique relatif aux systèmes dynamiques non linéaires. Il reprend l'étude dynamique d'un système et la relation existante avec les fractales. Le troisième chapitre fournit la démarche suivie pour la réalisation du logiciel. Pour finir, la conclusion fournit des résultats théoriques sur l'analyse de rythmes cardiaques effectuée par ***Ary Golberger*** et son équipe. Elle énumère les difficultées sous-jacentes à la réalisation du logiciel ainsi que les améliorations envisageables.

Chapitre 1 . Le spectre de Fourier , un outil pratique pour l'étude du rythme cardiaque :

§1 Introduction :

L'étude de la dynamique du rythme cardiaque ne peut être réalisée de manière aisée à partir de sa représentation h(t) dans le domaine temporel. En effet, les propriétés de la dynamique d'un processus physique sont plus facilement mises en évidence lorsqu'on utilise la représentation H(f) dans le domaine des fréquences. H(f) décrit l'amplitude et la phase du processus en fonction de la fréquence f. Ces deux représentations différentes décrivent le même processus et peuvent se déduire l'une de l'autre à l'aide des équations de la transformée de Fourier :

$$H(f) = \int_{-\infty}^{+\infty} h(t) e^{2\Pi i f t} dt$$

$$h(t) = \int_{-\infty}^{+\infty} H(f) e^{-2\Pi i f t} df \qquad (1)$$

Ce chapitre a pour but de préciser les bases théoriques nécessaires pour cette transformation et d'expliciter un algorithme numérique qui permet d'effectuer la transformation. Il est composé de cinq parties :

- La première partie fournit la théorie de base des transformées de Fourier appliquées à des fonctions continues.

- La seconde partie reprend les diverses modifications appliquées à la théorie de base pour permettre le calcul des transformées de Fourier de fonctions discrétisées.

- La troisième partie explique les divers mécanismes utilisés par un algorithme pour obtenir la transformée de Fourier d'un ensemble de valeurs numériques complexes.

• La quatrième partie révèle toutes les astuces utilisées par l'algorithme pour le calcul de la transformée de Fourier d'un ensemble de valeurs numériques réelles.

• La cinquième partie explique les dernières modifications à apporter aux résultats obtenus dans la partie précédente afin de calculer le spectre de Fourier des données initiales. Elle explique également l'utilité de ce spectre dans l'étude de la dynamique d'un processus cardiaque.

§2 Théorie de base des transformées de Fourier :

Dans le contexte de ce mémoire, je n'envisage que les transformées qui s'appliquent à des fonctions temporelles. Celles-ci permettent de faire passer une fonction **h(t)**, où t est exprimé en seconde, à une fonction **H(f)** où f est exprimée en cycles par seconde ou hertz. Le passage d'une fonction à l'autre est rendu possible grâce aux équations (1) décrites dans l'introduction de ce premier chapitre.

Une analyse rapide des équations (1) permet de prouver que la transformation de Fourier est une transformation linéaire :

$$TF\,(h(t)+g(t)) = \int_{-\infty}^{+\infty}(h(t)+g(t))e^{2\Pi ift}\,dt$$

$$= \int_{-\infty}^{+\infty}h(t)e^{2\Pi ift}\,dt + \int_{-\infty}^{+\infty}g(t)e^{2\Pi ift}\,dt$$

$$= TF(h(t)) + TF(g(t))$$

$$TF(\alpha^*\ h(t)) = \int_{-\infty}^{+\infty}(\alpha^*\ h(t))e^{2\Pi ift}\,dt$$

$$= \alpha^* \int_{-\infty}^{+\infty}h(t)e^{2\Pi ift}\,dt$$

$$= \alpha^*\ TF(h(t))$$

Le tableau ci-dessous reprend les propriétés des transformées de Fourier pour des fonctions possédant certaines symétries parmi les suivantes :

- paire h(t) = h(-t)
- impaire h(t) = -h(-t)
- purement réelles
- purement imaginaires.

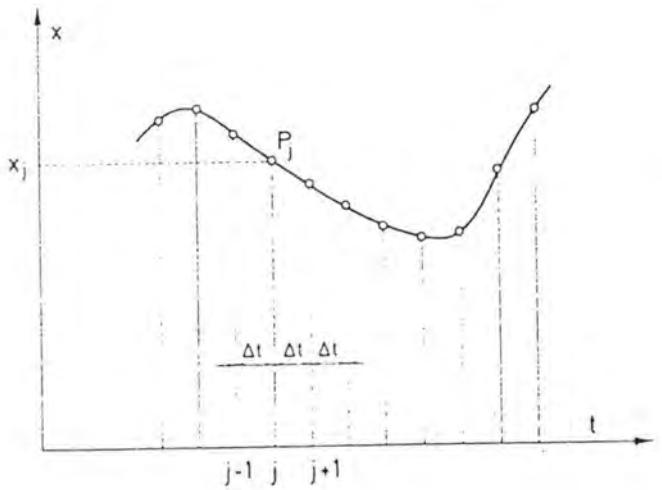
Dans ce tableau, $[H(f)]^*$ signifie le complexe conjugué de H(f). Pour rappel, le complexe conjugué de $Ae^{i\omega t}$ n'est rien d'autre que $Ae^{-i\omega t}$.

h(t) réelle	$H(-f) = [H(f)]^*$
h(t) imaginaire	$H(-f) = - [H(f)]^*$
h(t) paire	H(f) paire
h(t) impaire	H(f) impaire
h(t) réelle et paire	H(f) réelle et paire
h(t) réelle et impaire	H(f) imaginaire et impaire
h(t) imaginaire et paire	H(f) imaginaire et paire
h(t) imaginaire et impaire	H(f) réelle et impaire

Les symétries décrites ci-dessus seront d'une grande utilité dans les troisième et quatrième parties de ce chapitre. Elles permettront d'améliorer l'efficacité de l'algorithme de calcul des transformées de Fourier.

§3 Théorie des transformées de Fourier adaptée à la discrétisation :

Dans la pratique, l'étude d'un processus ne s'effectue pas à partir de la fonction h(t) le décrivant, mais à partir d'un ensemble de valeurs numériques, résultat de la discrétisation de la fonction h(t).



Notons Δ l'intervalle séparant deux données consécutives. Nous obtenons pour l'ensemble des valeurs numériques :

$h_n = h(n\Delta)$ où n = ..., -3, -2, -1, 0, 1, 2, 3,....

3.1. Fréquence critique de Nyquist :

A chaque intervalle Δ correspond une fréquence spéciale $\mathbf{f_c}$ appelée **fréquence critique de Nyquist** et équivalente à :

$$f_c = \frac{1}{2\Delta} \qquad (2)$$

Cette fréquence est intéressante et ce pour deux raisons :

A) Théorème de discrétisation :

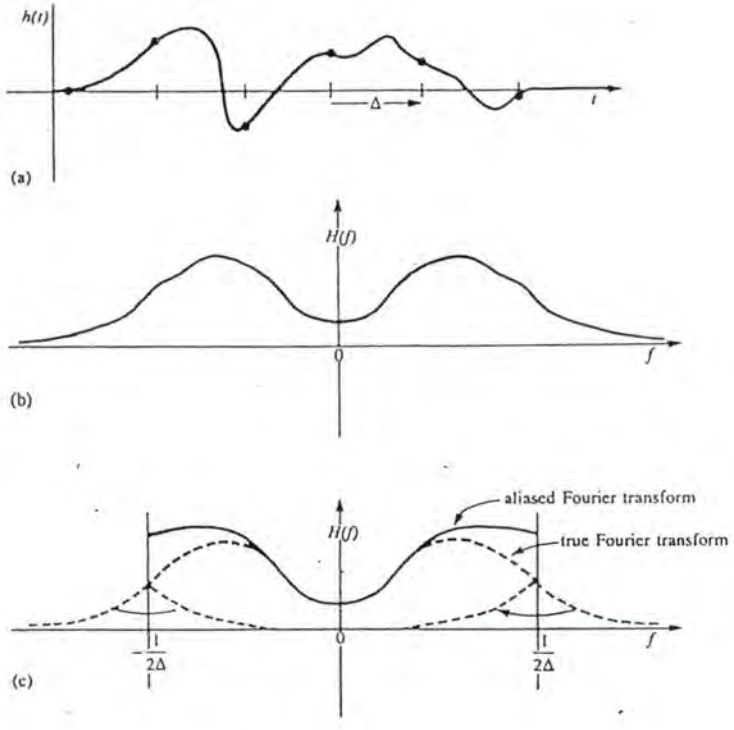
Si une fonction continue h(t), discrétisée avec un intervalle Δ, est limitée à une largeur de bande de fréquence inférieure en grandeur à f_c (i.e. $H(f) = 0 \ \forall f$ tq $|f| > f_c$), alors la fonction h(t) est entièrement déterminée par l'ensemble $\{ h_n \}$, résultat de la discrétisation. En fait, il est montré que :

$$h(t) = \Delta \sum_{n=-\infty}^{+\infty} h_n \frac{\sin[2\Pi f_c (t - n\Delta)]}{\Pi(t - n\Delta)}$$

De ce théorème, on déduit que le contenu d'information d'une fonction limitée en largeur de bande est infiniment plus petit que celui d'une fonction continue générale.

B) Le phénomène d'***Aliasing*** :

Ce phénomène apparaît lors de la discrétisation d'une fonction continue qui n'est pas limitée dans la largeur de bande définie par la fréquence de Nyquist f_c. L'effet provoqué par la discrétisation consiste en ce que chaque composant de fréquence extérieure à l'intervalle $[f_c,-f_c]$ est translaté de manière aléatoire dans cet intervalle. Afin d'éviter ce phénomène, il faut déterminer la largeur de bande naturelle du processus et effectuer une discrétisation suffisamment rapide pour avoir deux points par cycle de la fréquence la plus élevée. Les graphiques suivants illustrent le phénomène d'aliasing :



L'utilisation de ce phénomène dans l'étude du rythme cardiaque consistera à vérifier l'efficacité de la discrétisation du signal. En effet, le rythme cardiaque est une fonction continue et nous ne possédons de celle-ci qu'un ensemble de valeurs discrètes. Dès lors en examinant le spectre de Fourier de cet ensemble, nous pourrons déterminer l'efficacité de la discrétisation. Pour ce faire, il suffit de vérifier que le spectre de Fourier approche la valeur 0 dès que la fréquence f s'approche de $-f_c$ en décroissant ou de f_c en croissant. Le cas contraire signifierait que l'ensemble des valeurs numériques ne représentent pas exhaustivement le signal émis par les battements du coeur.

3.2. Transformée de Fourier d'un ensemble de valeurs discrètes :

Nous supposons que le nombre de points est limité et est égal à N :

$$h_k = h(k\Delta) \qquad k = 0, 1, 2, ..., N-1 \qquad (3)$$

Nous émettons l'hypothèse suivante : N est pair et la fonction est non nulle sur un intervalle de temps fini. Cet intervalle contient les N points reçus. Nous allons estimer la transformée de Fourier pour :

$$f_n \equiv n / N\Delta \qquad \text{avec } n = -N/2, ... N/2 \qquad (4)$$

où les deux limites correspondent à $-f_c$ et f_c.

Remarquons encore, que de N points initiaux, nous comptons en obtenir N+1 finaux. Ceci est possible car les deux valeurs limites sont équivalentes.

Utilisant (3) et (4), nous allons exprimer l'intégrale de (1) à l'aide d'une sommation :

$$H(f_n) = \int_{-\infty}^{+\infty} h(t)\, e^{2\Pi i f_n t}\, dt$$

$$\approx \sum_{k=0}^{N-1} h_k e^{2\Pi i f_n t_k} \Delta$$

$$= \Delta \sum_{k=0}^{N-1} h_k e^{2\Pi i \frac{kn}{N}}$$

$$= \Delta\, H_n \qquad (5)$$

En examinant (5), nous constatons que H_n est périodique en n de période N (i.e. $H_{-n} = H_{N-n}$ n = 1, 2, ...) :

$$H_{N-n} = \sum_{k=0}^{N-1} h_k e^{2\Pi ik\frac{(N-n)}{N}}$$

$$= \sum_{k=0}^{N-1} h_k e^{2\Pi ik} e^{-2\Pi ik\frac{n}{N}}$$

$$= \sum_{k=0}^{N-1} h_k e^{-2\Pi ik\frac{n}{N}} = H_{-n}$$

où nous avons utilisé la relation suivante : $e^{2\Pi ik} = 1$.

Nous considérons alors que n varie de 0 à N-1 en tenant compte du fait que :

- la fréquence nulle correspond à n = 0,
- les fréquences positives correspondent à $1 \leq n \leq N/2 - 1$,
- les fréquences négatives correspondent à $N/2 + 1 \leq n \leq N-1$,
- Les fréquences de Nyquist correspondent à N/2.

Les propriétés pour les transformées continues exprimées dans le tableau de la page 12 restent valables pour les transformées discrètes. Il suffit de remplacer :

- $h(t) \rightarrow h_k$
- $H(f) \rightarrow H_n$
- $H(-f) \rightarrow H_{N-n}$
- h(t) paire $\rightarrow h_k = h_{N-k}$
- h(t) impaire $\rightarrow h_k = -h_{N-k}$

La formule de la transformée inverse est le dernier renseignement à fournir en ce qui concerne les transformées de Fourier de fonctions discrétisées :

$$h_k = \frac{1}{N} \sum_{n=0}^{N-1} H_n e^{-2\Pi ik\frac{n}{N}} \qquad (6)$$

§4 Algorithme complexe de calcul des transformées de Fourier : FFT

Jusqu'au milieu des années 60, la transformée de Fourier était implémentée d'une façon particulière. Le nombre $\mathbf{W} \equiv e^{2\Pi i/N}$ était défini et l'on écrivait

$$H_n = \sum_{k=0}^{N-1} W^{nk} h_k$$

Ce qui revenait à multiplier le vecteur des h_k par une matrice dont l'élément situé en position (n,k) avait pour valeur W à la puissance n*k :

$$\left(H_n\right)_{n\times 1} = \left[W^{nk}\right]_{n\times k}\left(h_k\right)_{k\times 1}$$

Cette multiplication matricielle nécessite N^2 multiplications complexes, ainsi qu'un plus petit nombre d'opérations nécessaires au calcul de la puissance de W. Au total, une telle implémentation réclame $O(N^2)$ opérations, ce qui est énorme !

Nous allons montrer comment réduire ce nombre d'opérations et passer de $O(N^2)$ opérations à $O(N\log_2 N)$ opérations. Pour ce faire, nous allons appliquer le raisonnement suivi par ***Danielson et Lanczos***.

4.1. Calcul de la transformée de Fourier de longueur N à l'aide de deux transformées de longueur N/2.

$$H_n = \sum_{k=0}^{N-1} h_k e^{2\Pi i k\frac{n}{N}}$$

$$= \sum_{k=0}^{N/2-1} h_{2k} e^{2\Pi i(2k)\frac{n}{N}} + \sum_{k=0}^{N/2-1} h_{2k+1} e^{2\Pi i(2k+1)\frac{n}{N}}$$

$$= \sum_{k=0}^{N/2-1} h_{2k} e^{2\Pi i k \frac{n}{(N/2)}} + W^n \sum_{k=0}^{N/2-1} h_{2k+1} e^{2\Pi i k \frac{n}{(N/2)}}$$

$$= H_n^p + W^n H_n^i \qquad (7)$$

Dans la formule (7), H_n^p est la n^e composante de la transformée de Fourier de longueur N/2 formée des composantes paires de l'ensemble $\{h_k\}$ des données initiales.

H_n^i est semblable à H_n^p excepté que c'est l'ensemble des composantes impaires de $\{h_k\}$ qui est utilisé.

Il faut remarquer que n varie de 0 à N alors que H_n^p et H_n^i sont des transformées de longueur N/2 (i.e. périodique en n de période N/2). Dès lors chacune des transformées est répétée à travers deux cycles pour obtenir la transformée finale H_n.

Utilisant la même technique pour réduire H_n^p et H_n^i, nous pouvons exprimer H_n à l'aide de quatre transformées de longueur N/4 :

$$H_n^{pp},\ H_n^{pi}, H_n^{ip}, H_n^{ii}$$

où H_n^{pp} est la notation représentant la transformée utilisant les nombres qui se trouvent en position paire dans l'ensemble des valeurs servant à déterminer H_n^p.

$H_n^{pi}, H_n^{ip}, H_n^{ii}$ se définissent de manière analogue.

Cette technique requiert un nombre d'éléments initiaux dont la valeur égale une puissance de deux. Cette exigence peut être facilement atteinte si l'ensemble initial est trop petit. Il suffit de lui ajouter des valeurs nulles jusqu'au moment où l'on atteint la puissance de deux supérieure en valeur et la plus proche. Cette précondition permet d'appliquer la récursivité pour obtenir à la limite des transformées de Fourier de longueur unitaire. Or ces transformées unitaires ne sont rien d'autre que des identités. De cette manière, pour chaque composition possible de p et de i, il existe un élément de départ h_k :

$$\exists k: 0 \le k \le N-1, H_n^{pippiii..pp} = h_k$$

Il existe $\log_2 N$ compositions possibles.

4.2. Association d'un élément de l'ensemble de départ à une composition ppiii....pipp :

Ayant découvert ce mécanisme récursif, il suffit, maintenant de découvrir quel est l'élément de départ h_k qui correspond à la transformée unitaire $H_n^{pippiii...pp}$. Nous utilisons une petite astuce pour résoudre ce problème :

1. Inverser les éléments de la composition :
   ppip → pipp
2. Associer à chaque élément p de la composition nouvellement obtenue la valeur 0 et à chaque élément i, la valeur 1.
3. La suite de 0 et de 1 que nous obtenons au pas 2 correspond à la valeur binaire k de l'élément h_k correspondant.

4.3. Réarrangement des valeurs en entrée selon l'inversion des bits:

Afin de rendre l'algorithme plus performant, nous plaçons les éléments de départ selon l'ordre ***inverse des bits*** défini à l'étape 4.3. Ayant placé les éléments de cette manière, nous avons rangé les transformées unitaires dans l'ordre d'utilisation. Cet arrangement permet, en combinant les paires d'éléments adjacents, d'obtenir les transformées de Fourier de longueur 2. Ensuite, combiner les paires adjacentes de paires permet de trouver les transformées de Fourier de longueur 4 et ainsi de suite jusqu'à l'obtention de la transformée finale. Etant donné que chaque combinaison requiert ± N opérations et qu'il y a $\log_2 N$ combinaisons, nous remarquons que l'algorithme sera de l'ordre de $N\log_2 N$. Nous supposons que l'opération de réarrangement des données est d'un ordre inférieur ou égal à $N\log_2 N$.

4.4. Un exemple pour éclairer la situation :

Soit un ensemble de 8 données numériques h_k k = 0..7, examinons le calcul de la n^e composante de la transformée de Fourier :

$$H_n = H_n(h_0, h_1, h_2, h_3, h_4, h_5, h_6, h_7)$$

$$H_n = H_n^p(h_0, h_2, h_4, h_6) + W^n H_n^i(h_1, h_3, h_5, h_7)$$

$$H_n = H_n^{pp}(h_0, h_4) + W^n H_n^{pi}(h_2, h_6)$$

$$+ W^n \left[H_n^{ip}(h_1, h_5) + W^n H_n^{ii}(h_3, h_7) \right]$$

$$H_n = H_n^{ppp}(h_0) + W^n H_n^{ppi}(h_4) + W^n H_n^{pip}(h_2)$$
$$+ W^n W^n H_n^{pii}(h_6) + W^n\left[H_n^{ipp}(h_1) + W^n H_n^{ipi}(h_5)\right]$$
$$+ W^n\left[W^n H_n^{iip}(h_3) + W^n W^n H_n^{iii}(h_7)\right]$$

En appliquant la règle du point 4.2., nous remarquons :

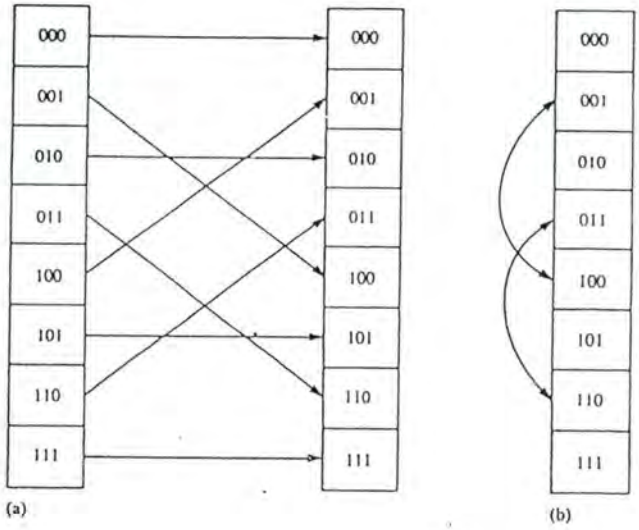
- $H_n^{ppp}(h_0)$: ppp → ppp → 000 valeur binaire de 0
- $H_n^{ppi}(h_4)$: ppi → ipp → 100 valeur binaire de 4
- $H_n^{ipi}(h_5)$: ipi → ipi → 101 valeur binaire de 5

Et ainsi de suite pour les autres éléments du calcul.

4.5. Structure de l'algorithme FFT :

L'algorithme FFT se compose de deux sections :

* Une première section a pour but d'arranger les données dans le tableau dans l'ordre inverse des bits. Ce réarrangement ne nécessite aucun élément de stockage supplémentaire.En effet, il implique seulement des permutations de paires d'éléments. Dans l'exemple, nous constatons que $h_1$permute avec h_4 et h_3 avec h_6.



* Une seconde section s'occupe du calcul de la transformée de Fourier. Elle se compose d'une boucle principale exécutée $\log_2 N$ fois et qui fournit les transformées d'ordre 2, 4, 6,... successivement. A chaque étape de ce processus, deux boucles internes utilisent les sous-transformations déjà calculées et implémente la récursivité définie par Danielson-Lanczos.

4.6. Routine FFT utilisée dans ce mémoire :

Cette routine exige trois paramètres en entrée :

* **N** le nombre de valeurs numériques complexes pour lesquelles on désire calculer la transformée de Fourier.

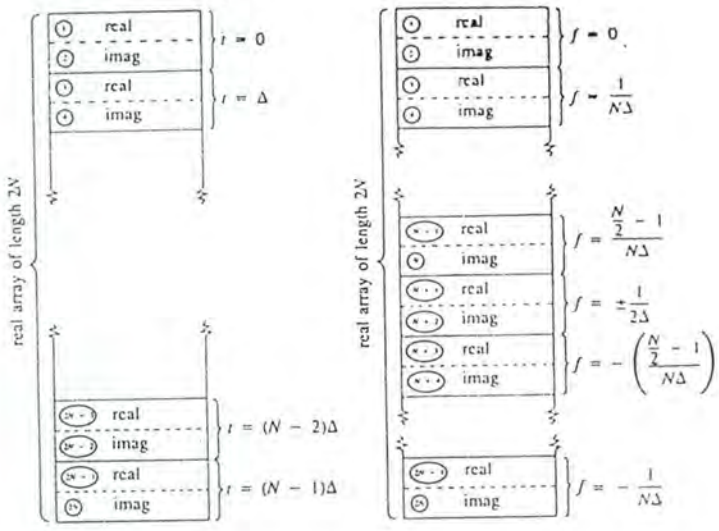
* **data** le tableau contenant les valeurs numériques de départ. Il s'agit d'un tableau de nombres réels de dimension égale au double du nombre N de valeurs. Chaque paire d'éléments successifs de data constitue un nombre complexe. Le premier membre de chaque paire joue le rôle de la partie réelle du nombre tandis que le second membre constitue la partie imaginaire. Par exemple data[0] est la partie réelle de h_0 et data[1] est sa partie imaginaire. Data[2N-1] et data[2N] jouent le même rôle pour h_{N-1}.

* **Signe** prend la valeur $\pm$ 1 et constitue le signe de l'exposant de W. De cette manière, lorsque Signe = +1 la routine retourne la transformée de Fourier de la fonction discrétisée. Tandis que lorsque Signe vaut -1, la routine retourne la transformée inverse de Fourier.

Le résultat de la routine est renvoyé dans le même tableau data et sous le même format. Cependant, l'ordre des H_n est différent :

* H_0 est stocké dans data[1] et data[2].
* H_i relatif à la plus petite fréquence positive non nulle est stockée dans data[3] et data[4].
* H_i relatif à la plus petite fréquence négative non nulle est stockée en data[2N-1] et data[2N-2].
* Les éléments data[5] à data[N] contiennent les H_i correspondant aux fréquences positives rangées par ordre croissant de grandeur.
* Les éléments data[2N-3] jusque data[N+3] contiennent les H_i correspondant aux fréquences négatives rangées par ordre croissant de grandeur.
* Enfin, data[N+1] et data[N+2] contiennent le H_i relatif à la fréquence de Nyquist $1/2\Delta$.

Le schéma suivant va permettre de mieux comprendre l'ordre des éléments résultant du calcul de la transformée.



§5 Algorithme réel :

Les données numériques reçues, pour lesquelles on désire obtenir la transformée de Fourier, sont souvent réelles et non complexes. L'utilisation de l'algorithme complexe du paragraphe précédent nécessite que toutes les parties imaginaires soient de valeur nulle. Le résultat obtenu possède alors les propriétés suivantes :

* $(H_{N-n})^* = H_n$
* H_0 et $H_{N/2}$ sont deux valeurs réelles.
* H_1 ..... $H_{N/2-1}$ sont des valeurs indépendantes.

De ces propriétés, nous déduisons que le degré de liberté du résultat, qui vaut 2(N/2-1)+2 = N, est identique à celui de l'ensemble initial des données réelles. Nous constatons, de cette manière, qu'appliquer l'algorithme complexe tel quel à un ensemble de données réelles est une méthode inefficace, aussi bien du point de vue temps d'exécution que du point de vue espace de stockage exigé.

Deux méthodes plus efficaces existent :

* Soit stocker dans le tableau de départ deux *fonctions* réelles distinctes de manière à pouvoir tirer du résultat leurs transformées respectives. Cette méthode est expliquée dans 10°).

* Soit stocker les données intelligemment, sans valeurs nulles supplémentaires, dans un tableau de moitié de longueur. C'est-à-dire dont la

longueur est égale au nombre de données réelles. Cette méthode est développée dans la section suivante.

5.1. FFT appliqué à une *fonction* réelle (discrétisée) :

L'astuce utilisée dans cet algorithme consiste à éclater le tableau des données initiales en deux ensembles. Le premier ensemble reprend tous les éléments pairs du tableau tandis que le second ensemble reprend tous les éléments impairs. Nous considérons alors le tableau comme un tableau de nombres complexes de moitié de longueur. Les éléments du premier ensemble forment la partie réelle du tableau et les éléments du second ensemble forment sa partie complexe :

$$h_j = f_{2j} + if_{2j+1} \qquad j = 0..N/2 - 1 \qquad (8)$$

où N est le nombre de valeurs réelles initiales.

Nous appliquons ensuite l'algorithme complexe pour obtenir :

$$H_n = H_n^p + iH_n^i \qquad n = 0..N/2 - 1 \quad (9)$$

où nous définissons

$$H_n^p = \sum_{k=0}^{N/2-1} f_{2k} e^{2\Pi ikn/(N/2)}$$

$$H_n^i = \sum_{k=0}^{N/2-1} f_{2k+1} e^{2\Pi ikn/(N/2)}$$

Ceci s'obtient directement de l'équation de la transformée de Fourier :

$$H_n = \sum_{k=0}^{N/2-1} h_k e^{2\Pi ikn/(N/2)}$$

$$= \sum_{k=0}^{N/2-1} (f_{2k} + if_{2k+1}) e^{2\Pi ikn/(N/2)}$$

$$= \sum_{k=0}^{N/2-1} f_{2k} e^{2\Pi ikn/(N/2)} + i \sum_{k=0}^{N/2-1} f_{2k+1} e^{2\Pi ikn/(N/2)}$$

De ce résultat, il faut tirer la véritable transformée de Fourier F_n des éléments de départ f_i. Se référant au mécanisme de Danielson-Lanczos décrit dans la formule (7), nous pouvons écrire que

$$F_n = F_n^p + e^{2\Pi in/N} F_n^i \qquad n = 0,...N-1$$

Or en examinant les expressions de F_n^p et F_n^i dans (7), nous constatons que $F_n^p = H_n^p$ et $F_n^i = H_n^i$. Dès lors, nous pouvons déduire F_n à partir de H_n grâce à

$$F_n = \frac{1}{2}\left(H_n + H^*_{N/2-n}\right) - \frac{i}{2}\left(H_n + H^*_{N/2-n}\right)e^{2\Pi in/N} \qquad n = 0..N-1$$

En effet, il suffit de vérifier que

$$\left[H_{N/2-n}\right]^* = \left[\sum_{k=0}^{n/2-1} f_{2k} e^{2\Pi \frac{N/2 - n}{N/2}} + i\sum_{k=0}^{n/2-1} f_{2k+1} e^{2\Pi \frac{N/2 - n}{N/2}}\right]^*$$

$$= \left[\sum_{k=0}^{N/2-1} f_{2k} e^{-2\Pi ikn/(N/2)} e^{2\Pi ik} + i\sum_{k=0}^{N/2-1} f_{2k+1} e^{-2\Pi ikn/(N/2)} e^{2\Pi ik}\right]^*$$

$$= \left[\sum_{k=0}^{N/2-1} f_{2k} e^{-2\Pi ikn/(N/2)}\right]^* - i\left[\sum_{k=0}^{N/2-1} f_{2k+1} e^{-2\Pi ikn/(N/2)}\right]^*$$

$$= F_n^p - i\,F_n^i$$

Dans le tableau reçu en résultat, nous avons à notre disposition seulement N/2-1 valeurs. Or la transformée de Fourier F_n des données initiales f_j doit être composée de N-1 valeurs. Donc, nous n'avons pas suffisamment de valeurs H_n pour obtenir l'entièreté des F_n. Néanmoins, ce résultat nous satisfait. En effet, la transformée F_n possède la symétrie suivante $F^*_{N-n} = F_n$ du fait que les données initiales sont réelles. De cette symétrie, nous concluons que la connaissance de la moitié de la transformée de Fourier F_n est suffisante. Il suffit d'appliquer la symétrie pour obtenir la seconde moitié.

En appliquant cette vision des choses, nous pouvons stocker le résultat, i.e. la demi-transformée, dans le tableau contenant les données initiales. De cette façon, nous réduisons l'espace de stockage utilisé. Cette économie de l'espace de stockage utilisé est rendue possible par le fait que F_0 et $F_{N/2}$ sont réelles et peuvent être stockées en data[0] et data[1] respectivement. Le tableau final contient donc N/2+1 valeurs de la transformée :

* data[0] = F_0
* data[1] = $F_{N/2}$
* data[2] et data[3] forment F_1
* ....

1	réel	F_0
2	réel	$F_{N/2}$
3	partie réelle	F_1
4	partie imaginaire	
5	partie réelle	F_2
6	partie imaginaire	
.	.	.
.	.	.
.	.	.
N/2-1	partie réelle	$F_{N/2-1}$
N/2	partie imaginaire	

§6 Spectre de Fourier :

Ayant appliqué l'algorithme du paragraphe précédent et sachant que le spectre de Fourier est défini par le graphe représentant

$$|F_n| = \sqrt{F_n * F_n^*}$$

en fonction de la fréquence f, il nous reste à calculer à partir des données du tableau data les valeurs de la norme des F_n's :

$$|F_0| = |data(1)|$$

$$|F_1| = |data(2)|$$

$$|F_2| = \sqrt{data^2(3) + data^2(4)}$$

.....

Une fois le spectre de Fourier représenté, une analyse de la dynamique du processus est possible. En effet, si le phénomène est périodique de période T, alors son spectre contiendra un pic aigu en la fréquence 1/T. Des pics secondaires beaucoup moins prononcés correspondent à des bruits périodiques présents dans le signal. Un exemple concret est présenté dans le chapitre consacré au développement du logiciel. Si le processus est apériodique, une analyse du spectre de Fourier est inutile voire impossible. Il suffit pour s'en rendre compte d'observer le résultat fourni par l'analyse d'un signal obtenu à l'aide d'une fonction fournissant des nombres aléatoirement et qui se trouve également dans le chapitre 3.

Chapitre 2 : Dynamique chaotique et théorie fractale : un outil d'analyse du rythme cardiaque.

§1 Introduction :

Mon mémoire étant la continuation de celui de Thierry Reniers, je me contenterai de donner un résumé des différentes définitions et méthodes concernant les fractales et la dynamique chaotique. En effet, développer ici l'étude de ces deux domaines reviendrait en quelque sorte à recopier le mémoire de monsieur Reniers, ce qui n'est pas le but de ce mémoire. De fait, mon travail consiste plus en une application de ces deux théories. Pour les personnes désireuses d'avoir de plus amples renseignements concernant les définitions, théorèmes et autres concepts de ces domaines non développés dans ce travail-ci, je leur conseille de parcourir le mémoire de monsieur Reniers.

Ce chapitre est composé de trois parties distinctes :

- La première partie reprend les points importants à se rappeler en ce qui concerne les fractales en relation avec le chaos déterministe et les attracteurs.

- La seconde partie contient les rappels concernant les diverses dimensions fractales existantes. Elle explique aussi une méthode de calcul de la dimension d'une fractale. Plusieurs méthodes existent, mais je ne développe que celle utilisée dans le cadre de ce travail. Les développements concernant les autres méthodes sont clairement donnés chez T. Reniers.

- La troisième partie explique l'utilisation pratique de ces notions dans le cadre de l'étude du rythme cardiaque.

§2 Rappels concernant la dynamique et les fractales :

Les structures fractales résultent souvent d'une dynamique non linéaire chaotique : là où des processus chaotiques façonnent l'environnement (le bord de mer, l'atmosphère, une faille géologique), des fractales apparaissent (la ligne cotière, les nuages, les formations rocheuses). Au début les mathématiques des fractales furent développées indépendamment de la dynamique non linéaire et, même de nos jours, les connexions entre ces disciplines ne sont pas entièrement élucidées.

Le développement topologique concernant ces notions ne sera pas repris dans ce paragraphe. Je préfère me concentrer sur une approche dynamique et qualitative plutôt que sur une approche topologique et formelle.

Une fractale, selon Mandelbrot, est une figure géométrique ou un objet naturel constitué par des parties de taille et d'orientation diverse mais de forme ou de structure semblable au tout. Elle est très irrégulière, interrompue ou fragmentée, et le reste quelle que soit l'échelle d'observation.

Un système dynamique est un processus qui évolue dans le temps de manière déterministe. Il faut comprendre par déterministe que si l'on connaît avec exactitude l'état initial du processus, alors le futur de celui-ci est prévisible avec certitude. Parmi ces systèmes dynamiques, certains, très complexes, dépendent sensiblement des conditions initiales. Pour de tels systèmes, une imprécision infime concernant la connaissance de l'état initial conduit à des prédictions sur le futur entachées d'une importante erreur. Ces systèmes conduisent au chaos.

Ayant rappelé les deux concepts de base de ce chapitre, il me reste à montrer quel est le lien qui existe entre les deux. Dans ce but, je vais décrire brièvement l'étude d'un système dynamique. De cette étude, je vais faire apparaître l'élément qui servira de lien avec la théorie fractale.

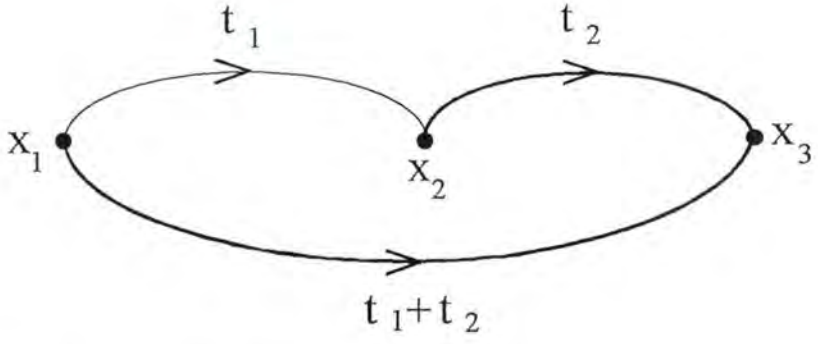
2.1. Etude d'un système dynamique :

A) Définition :

Un système dynamique est un processus d'évolution temporelle que l'on peut définir par un triplet (X, $\Re$, Π) où (X, d) est un espace métrique et Π est une application $X \times \Re \rightarrow X$ satisfaisant les trois propriétés suivantes :

* $\Pi(x,0) = x \quad \forall x \in X$ (axiome de l'identité)
  Elle signifie qu'au temps 0 le système n'a pas encore évolué.

* $\Pi(\Pi(x,t_1),t_2) = \Pi(x,t_1+t_2) \quad \forall x \in X , \forall t_1,t_2 \in \Re$
  (axiome du groupe). La signification de cette propriété est clairement expliquée par le schéma suivant :



* Π est continue.

Dans l'optique suivie pour ce travail, le système dynamique est représenté à l'aide d'une équation différentielle ordinaire du premier ordre dans l'espace métrique des réels :

$$\frac{dx(t)}{dt} = f(x(t))$$

où $x(t) \in \Re^n$ ***et*** $f : \Re^n \rightarrow \Re^n$ ***est une fonction continue.***

Si cette équation différentielle possède une solution unique φ (t,x), appelée flot et telle que φ (0,x) = x, alors on peut montrer que φ satisfait aux trois propriétés citées ci-dessus. Dès lors, l'existence de φ est équivalente à la donnée de l'application Π : $\Re^n \times \Re \rightarrow \Re^n$, avec Π (x,t) = φ (t,x) et correspond donc bien à un système dynamique sur $\Re^n$.

Une définition analogue est possible lorsque f dépend explicitement du temps :

$$\frac{dx(t)}{dt} = f(x(t),t)$$

où $x(t) \in \Re^n$ et $f : \Re \times \Re^n \rightarrow \Re^n$ continue.

Alors $\forall\ (t_0, x_0) \in \Re \times \Re^n$, $\varphi(t,t_0, x_0)$ solution unique permet de définir un système dynamique sur $X = \Re \times \Re^n$ par $\Pi : \Re \times \Re^n \times \Re \rightarrow \Re \times \Re^n$ avec $\Pi((t,x),s) = (s+t,\ \varphi(s+t, t, x))$.

B) Espace des phases :

Un système dynamique possède deux aspects,

* sa dynamique, qui consiste en la loi qui régit l'évolution de l'état du système.
* son état, qui contient les informations caractérisant le système.

L'état du système existe par rapport à un espace abstrait appelé ***espace des phases*** dans lequel il est représentable. En effet, dans cet espace des phases, l'état correspond aux coordonnées d'un point correspondant aux valeurs prises par les variables intervenant dans le système. La dimension de l'espace des phases est égale au nombre de variables servant à décrire le système. L'espace des phases est utilisé afin de représenter l'évolution du système dynamique.

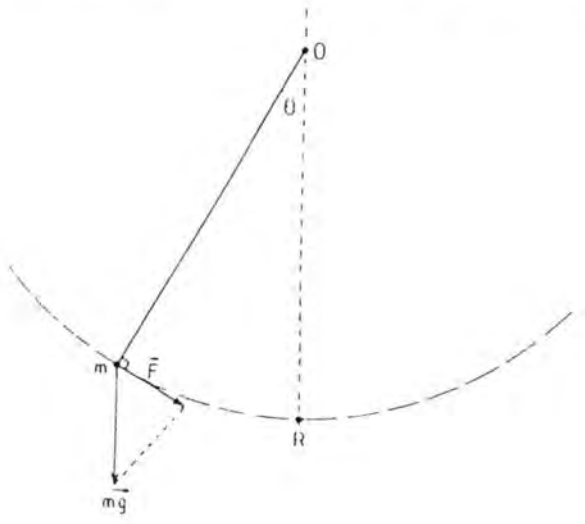
En utilisant les notations définies en A), nous pouvons définir l'espace des phases comme étant l'espace X sur lequel a été définie l'application Π du système dynamique $(X,\Re,\Pi)$.

L'espace des phases devient un outil d'analyse très efficace lorsque le système dynamique n'est pas intégrable. En effet, grâce à la représentation géométrique des trajectoires, nous pouvons déduire le comportement du système.

Exemple :

Afin d'illustrer ces notions concernant l'espace des phases, visualisons, sans développements théoriques (que l'on peut trouver dans

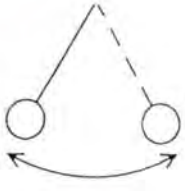
le mémoire de T. Reniers), les évolutions des trajectoires pour un pendule simple :

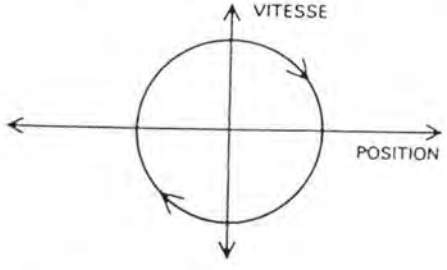


où m est la masse du pendule, g est l'accélération gravifique et $\overrightarrow{OR}$ la verticale.

Premier cas :

Nous considérons un pendule idéal, c'est-à-dire sur lequel n'agit aucune force de frottement. Le pendule va et vient sans arrêt et sans variation d'amplitude des oscillations :

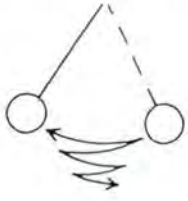


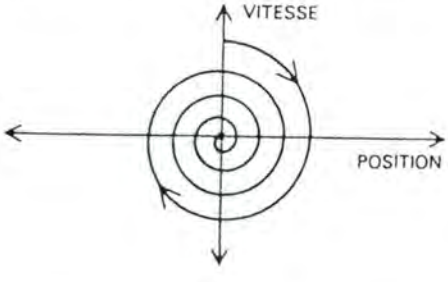


La trajectoire dans l'espace (vitesse, position) est une courbe fermée.

Second cas :

Le pendule est soumis à des forces de frottement qui ont pour effet de diminuer l'amplitude des oscillations :





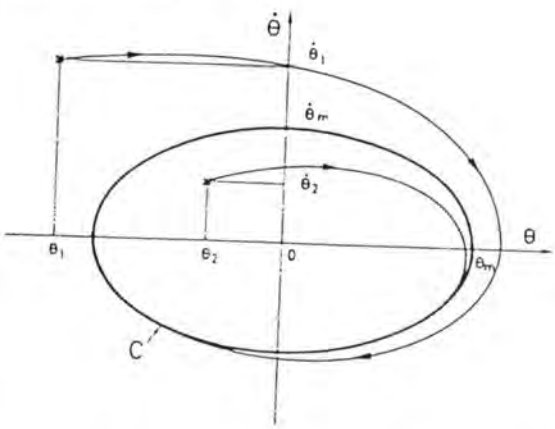
Ici, dans l'espace (vitesse, position), la trajectoire du pendule est une spirale dont le centre correspond à la position finale du pendule, laquelle est une position d'équilibre.

C) Attracteur :

Basant l'étude du système sur la représentation de l'état dans l'espace des phases, nous devons chercher géométriquement l'état vers lequel le système convergera. Ainsi, dans le second cas de notre exemple, nous remarquons que la spirale converge toujours vers le point final de la trajectoire qui est une position d'équilibre stationnaire (i.e. x(t) pour lequel dx/dt = 0). Ce point fixe est appelé ***Attracteur.***

La notion d'attracteur se retrouve dans les systèmes dynamiques comportant des phénomènes de frottement. Un attracteur est une région de l'espace des phases vers laquelle convergent les trajectoires et qui possède une dimension inférieure à celle de l'espace des phases. Plusieurs types d'attracteurs existent, comme :

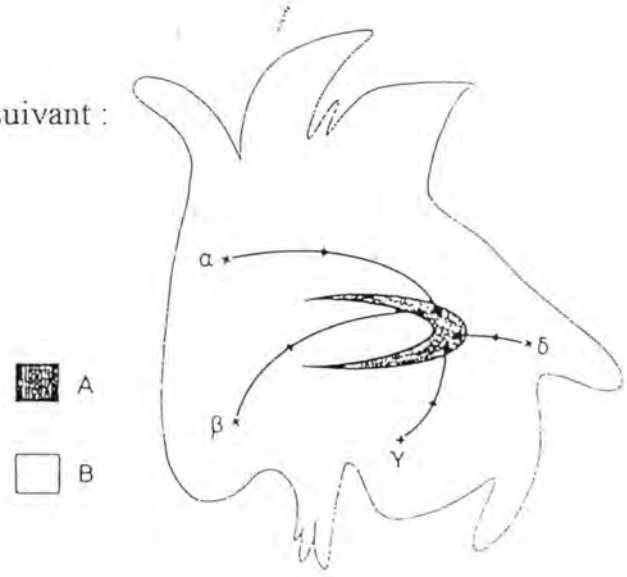
* l'attracteur correspondant à un point d'équilibre, comme ci-dessus.
* l'attracteur correspondant à un cycle limite qui est représenté par une courbe fermée vers laquelle convergent les trajectoires d'un système dynamique périodique.



Si une condition initiale décrit un point appartenant au cycle C, alors ce point ne quittera jamais ce cycle. Rappelons que les trajectoires obtenues dans l'espace des phases sont déterminées à partir de conditions initiales précises.

Plusieurs attracteurs peuvent exister pour un même système. Chacun de ces attracteurs correspond à un ensemble de points de l'espace des phases, déterminés géométriquement par les conditions initiales et qui sont à l'origine de trajectoires évoluant vers l'attracteur considéré. Cet ensemble de points constitue le ***bassin d'attraction*** de l'attracteur :

Ainsi, dans l'exemple suivant :



A est un attracteur et B son bassin d'attraction constitué des points α, β, γ, δ, ... qui sont origines de trajectoires convergeant vers A.

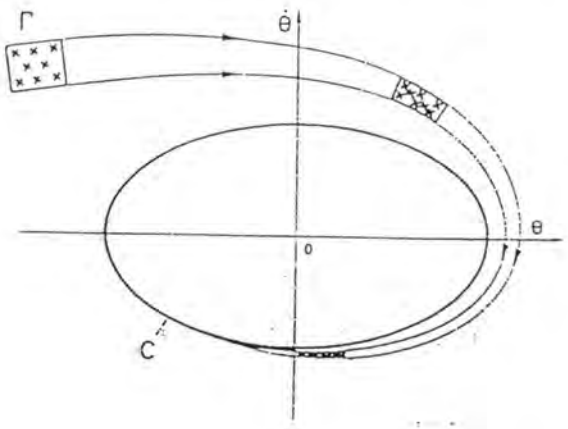
C'est ici que va apparaître pour la première fois la notion de fractales dans les systèmes dynamiques. En effet, des études ont fait remarquer que les frontières entre bassins d'attraction de divers attracteurs forment souvent des objets fractals.

D) Contraction des aires ou des volumes :

L'existence des attracteurs est généralement liée à l'aspect dissipatif des systèmes dynamiques. Les systèmes pour lesquels l'énergie se dissipe présentent des trajectoires qui convergent asymptotiquement vers un cycle limite ou un point fixe. Pour de tels systèmes, dans l'espace des phases, l'aire de tout un ensemble de conditions initiales diminue au cours du temps.

Si la dimension de l'espace est trois, alors cette propriété s'applique au volume. Une telle règle ne s'applique pas dans le cas des systèmes conservatifs. L'aire et le volume s'y conservent.

Le phénomène de contraction des aires ou des volumes est illustré par la figure suivante où la région Γ de conditions initiales se réduit à une ligne lorsque l'attracteur C est atteint.



L'observation de cet exemple nous permet de tirer quelques conclusions. Tout d'abord, la position des points initiaux devient de plus en plus floue au fur et à mesure que l'on se rapproche de l'attracteur. Cette information est irrémédiablement perdue une fois l'attracteur atteint. Deuxièmement, la dimension de l'attracteur doit être strictement inférieure à la dimension de l'espace des phases. Cette dimension de l'espace des phases est égale au nombre de variables utilisées par le système dynamique. Dans l'exemple ci-dessus, nous passons d'une région Γ possédant une aire non nulle à un cycle C possédant une aire nulle.

De toutes ces conclusions, nous pouvons tirer une définition plus formelle de la notion d'attracteur. Dans l'espace des phases, la solution d'un système d'équations différentielles ordinaires constitue un flot φ qui pour un système dissipatif, possède un attracteur.

Un attracteur A est un ensemble compact dans l'espace des phases satisfaisant aux propriétés suivantes :

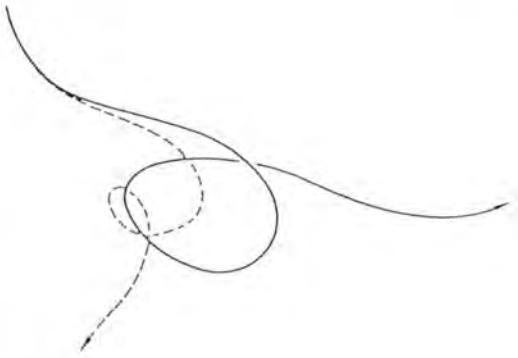
1. ***A est invariant sous l'action du flot : $\varphi(A) = A$.***
2. ***Le volume de A est nul dans l'espace des phases de dimension n***
3. ***A est contenu dans un domaine B, dont le volume est non nul et qui constitue son bassin d'attraction.***

E) Attracteurs étranges :

A ce point de la théorie, nous n'avons encore établi aucun lien entre les systèmes dynamiques et les fractales, si ce n'est par l'intermédiaire de la frontière des bassins d'attraction.Tous les attracteurs que nous avons esquissés, à titre d'exemples, possèdent une géométrie assez simple et caractérisent des systèmes dynamiques pas trop complexes. Nous allons, dorénavant, nous intéresser à des systèmes dont le comportement est apparemment aléatoire et pour lesquels aucun des attracteurs vus précédemment ne permet une description.

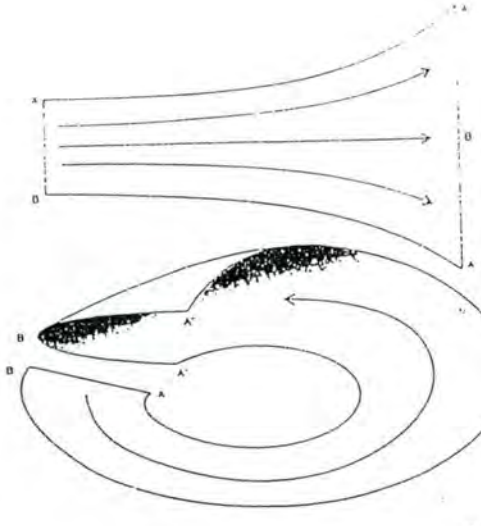
Il s'agit de systèmes soumis à des perturbations microscopiques qui s'amplifient et produisent rapidement des modifications importantes au niveau macroscopique. C'est dans ce cadre-ci, qu'apparaissent les systèmes dits sensibles aux conditions initiales. Cette propriété est caractérisée par le fait que deux trajectoires issues de conditions initiales voisines ne restent

proches l'une de l'autre que pendant un court instant et ensuite divergent de façon exponentielle :

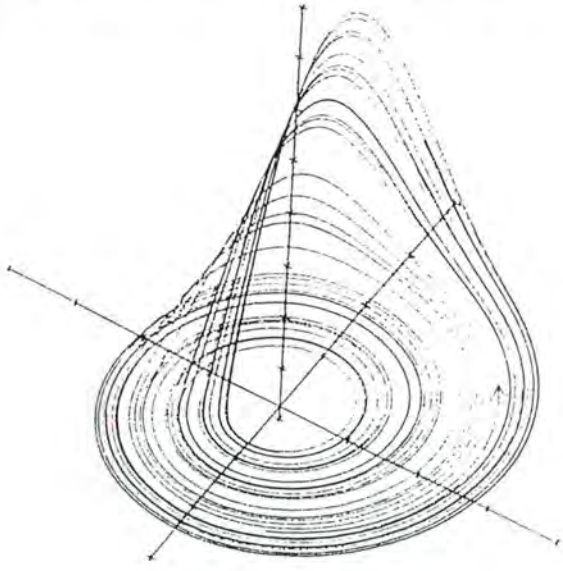


Les études de tels systèmes dynamiques conduisent aux *attracteurs étranges*.

Il n'est pas évident, de prime abord, de comprendre comment un système sensible aux conditions initiales et conduisant, donc, à une divergence exponentielle des trajectoires permet simultanément à ces mêmes trajectoires de converger asymptotiquement vers un attracteur. Nous comprendrons plus facilement cette possibilité en considérant comment se "dessine" un attracteur étrange. En effet, dans un premier temps, les trajectoires divergent et produisent un étirement latéral. Cette divergence est le résultat de la sensibilité aux conditions initiales et ne peut être que locale car la dimension de l'attracteur est finie. Ensuite, les trajectoires se rapprochent et restent dans un espace borné. A ce moment-là, l'objet se replie sur lui-même. Ce processus d'étirement et de repliement se répète à l'infini et fournit un attracteur étrange ressemblant à un nombre infini de plis imbriqués les uns dans les autres. Schématiquement, ce processus se présente comme suit :



et aboutit finalement à un attracteur semblable à celui-ci :



Attracteur étrange de Rossler.

Cette construction de l'attracteur étrange à l'aide du processus d'étirement et de repliement répété un nombre infini de fois, permet de constater que cet attracteur est un objet dont la structure apparaît d'autant plus détaillée mais identique ou semblable que le grossissement est important. Se référant à la définition de fractale fournie dans le paragraphe 2, nous constatons qu'un attracteur étrange est une fractale. Nous venons d'établir le lien entre les systèmes chaotiques et la géométrie fractale.

Le phénomène de contraction des aires ou des volumes nous informe que la dimension de l'attracteur est inférieure à la dimension de l'espace des phases. Donc, si le nombre de degrés de liberté du sytème est trois, la dimension d de l'attracteur doit être inférieure à trois. Or le phénomène de sensibilité aux conditions initiales, forçant la divergence des trajectoires, conduit l'attracteur à posséder une dimension d supérieure à deux. Donc, les attracteurs étranges forcent à envisager la définition d'une dimension non entière appelée dimension fractale. Cette constatation est la motivation du prochain paragraphe.

§3 Rappels concernant les dimensions fractales :

Une fractale est constituée par des figures de tailles et d'orientations diverses, mais qui sont toutes de formes similaires. Les détails d'une fractale à une certaine échelle sont semblables (mais pas nécessairement identiques) à ceux de la structure vue à une échelle différente. Cette propriété de ressemblance interne propre à toutes les fractales s'appelle la similitude interne.

Comme une fractale est formée de structures semblables dont les éléments sont de plus en plus petits, sa longueur peut être infinie.

Si nous mesurons la longueur d'une fractale avec une règle, nous constatons que certains composants seront plus petits que la longueur minimale que la règle permet d'évaluer. Lorsque la précision de l'instrument augmente, la longueur mesurée de la fractale croît. Parler de longueur d'une fractale est donc peu significatif, en sorte que les mathématiciens introduisent la ***dimension*** d'une fractale comme quantification de son aptitude à ***emplir*** l'espace.

Le concept familier de dimension s'applique aux objets de la géométrie classique, euclidienne. Les droites, les cercles sont de dimension un, les sphères sont de dimension deux et les boules de dimension trois. Certaines fractales ont des dimensions non entières (parfois fractionnaires). Alors qu'une ligne ordinaire emplit exactement un espace de dimension 1, une courbe fractale ***déborde*** sur l'espace à deux dimensions. Ainsi une courbe fractale comparable à la ligne côtière d'un pays, aura une dimension comprise entre un et deux. Plus la dimension d'une fractale est grande, plus la probabilité qu'une région donnée de l'espace contienne des parties de cette fractale est élevée.

Comme nous venons de le constater au paragraphe précédent, la dimension d'un attracteur étrange n'est pas entière. Donc celui-ci est une fractale.

3.1. Dimension de similitude :

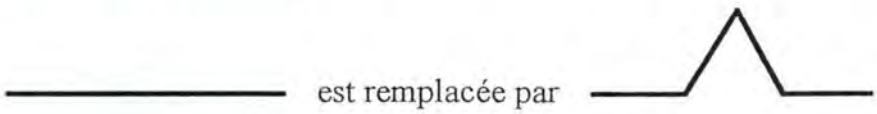
Si une figure se compose de a^D formes semblables de taille 1/a, alors l'exposant D peut être défini comme sa ***dimension de similitude*** et se note D_S. Elle ne peut être définie que pour des objets géométriquement simples et possédant une homothétie interne. De plus, elle peut prendre des valeurs non entières si la figure se compose de b formes semblables de taille 1/a. En effet, dans ce cas :

$$D_s = \frac{\log b}{\log a}$$

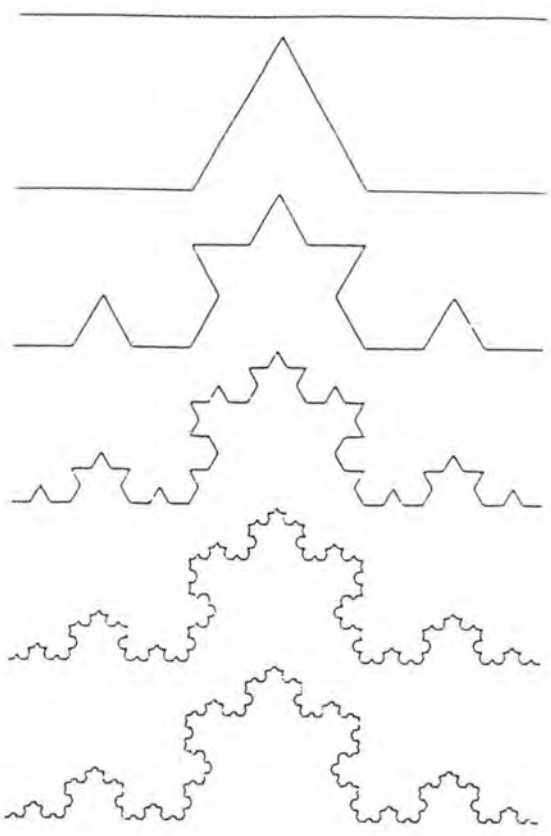
étant donné que b doit valoir a^b. Cette dimension est considérée comme un indice de complexité géométrique de la figure concernée.

Exemple :

Calculons la dimension de similitude de la courbe de Von Koch dont la construction consiste à remplacer un segment par quatre segments dont la longueur est égale au tiers du segment dont ils sont issus :



Nous trouvons $D_S = \log 4/\log 3 = 1.2618...$



Nous passons donc d'une simple ligne à une courbe dont la structure est plus compliquée, de telle sorte que la dimension de similitude reflète cette augmentation de complexité en prenant une valeur supérieure à 1.

Les restrictions d'application de cette dimension étant malheureusement, trop importantes, il a été nécessaire de proposer des définitions plus générales de la dimension fractale.

3.2. Les dimensions plus générales :

Les dimensions qui vont être définies dans cette partie du mémoire s'appliquent à des ensembles bornés d'un espace métrique X. Elles se définissent par une méthode de recouvrement et permettent de caractériser la complexité géométrique de ces sous-ensembles.

A) Dimension de Hausdorff-Bésicovitch :

Cette mesure s'applique aux sous-ensembles bornés de l'espace métrique $\Re^n$ muni de la norme euclidienne d. Cette dimension nécessite de définir auparavant la ***mesure de Hausdorff dans la dimension D :***

Soient D et ε deux nombres réels tels que $0 < D < \infty$ et $0 < \varepsilon < \infty$,
un ensemble borné $A \subset \Re^n$ et r le diamètre de A défini par
$r = \sup \{ d(x,y) : x,y \in A \}$.
Recouvrons A par un nombre infini dénombrable de sous-ensembles A_i de diamètres $r_i < \varepsilon$. Donc $A = \bigcup_{i=1}^{\infty} A_i$.
La mesure de Hausdorff dans la dimension D de A se définit comme étant :

$$M_D(A) = \lim_{\varepsilon \to 0} \inf_{r_i < \varepsilon} \sum_{i=1}^{\infty} (r_i)^D$$

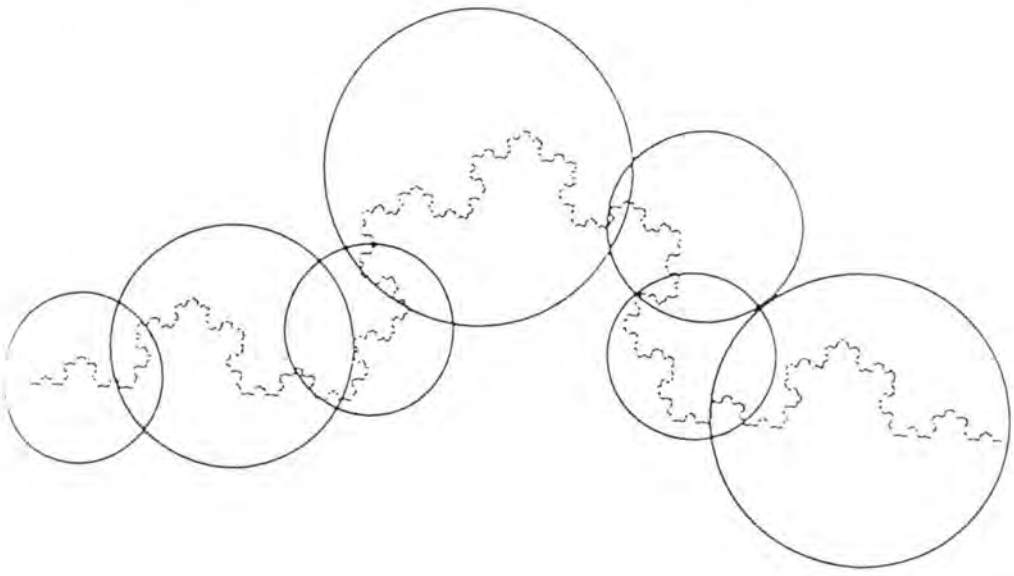
Le théorème suivant est à l'origine de la définition de la dimension de Hausdorff-Bésicovitch :

Théorème : Soient n un entier positif, et A un sous-ensemble borné de $(\Re^n, d)$,
Il existe un nombre réel unique $D_H \in [0,n]$ tq
pour $D \in [0,\infty]$:

$$M_D(A) = \begin{cases} \infty & \text{si } D < D_H \\ 0 & \text{si } D > D_H \end{cases}$$

La valeur D_H utilisée dans le théorème précédent est la dimension de Hausdorff-Bésicovitch de l'ensemble A. Cette dimension est généralement très difficile à déterminer rigoureusement.

Afin de comprendre intuitivement cette notion, considérons un recouvrement de la courbe de Von Koch par des cercles de diamètres r_i :



Si les cercles ne se chevauchent pas trop, la longueur totale de la courbe peut être approximée par la somme de leurs diamètres. L'infimum de cette somme (pour des diamètres inférieurs à une valeur positive ε) garantit la petitesse des intersections entre les cercles. De même, l'approximation de l'aire de la courbe de Von Koch par l'aire des cercles de recouvrement est proportionnelle à $r_1^2 + r_2^2 + \ldots + r_k^2$. Un même raisonnement est applicable au volume. Jusqu'à présent, nous venons de calculer les mesures usuelles associées aux dimensions 1, 2, 3. Nous pouvons généraliser cette discussion pour une dimension D quelconque où la mesure d'une sphère de diamètre r est proportionnelle à r^D.

La limite lorsque ε tend vers 0 dans la définition de la mesure de Hausdorff-Bésicovitch permet de réduire au maximum les intersections des cercles utilisés pour le recouvrement et donc d'obtenir une convergence de la mesure vers celle de l'ensemble initial considéré. Pour l'exemple de la courbe de Von Koch, il a été prouvé mathématiquement que la longueur diverge tandis que la surface est nulle. Par contre, la mesure de Hausdorff-Bésicovitch prend une valeur finie dans la dimension D = log4/log3.

B) Dimension de capacité :

Cette dimension-ci a été introduite pour pallier le problème de la complexité du calcul de la dimension de Hausdorff-Bésicovitch. Elle sera donc considérée comme plus pratique. La simplification introduite dans la définition de la ***dimension de capacité*** consiste à considérer que les boules utilisées pour le recouvrement ont toutes le même diamètre.

Définition :

Soit A un ensemble compact d'un espace métrique X.
Soient $\varepsilon > 0$ et B (x,ε) une boule fermée centrée en $x \in A$, de rayon ε .
Soit N (A,ε), le nombre minimum de boules fermées de rayon ε nécessaires pour recouvrir A. N (A,ε) est donc le plus petit entier positif M tel que $A = \bigcup_{n=1}^{M} B(x_n, \varepsilon)$ pour un certain ensemble de points distincts $\{ x_n : n = 1, 2, ...M\} \subset X$.
La dimension de capacité, ou de contenu, de l'ensemble A est la quantité obtenue par l'expression :

$$D_c = \lim_{\varepsilon \to 0} \frac{\log N(A, \varepsilon)}{\log \frac{1}{\varepsilon}}$$

La dimension de capacité peut être considérée comme un cas particulier de la dimension de Hausdorff-Bésicovitch. Ces deux dimensions coïncident souvent, mais lorsqu'elles diffèrent, il est possible de montrer que $D_C \geq D_H$.

3.3. La méthode "box-counting" :

Les dimensions définies précédemment ne posent aucun problème mathématiquement. Il est toujours possible, malgré la complexité du calcul de déterminer théoriquement leur valeur. Cependant, il est impossible pratiquement de réaliser le passage à la limite lorsque ε tend vers 0. En effet, une longueur nulle ne correspond pas à un concept physique. Dès lors, diverses méthodes de calcul de ces dimensions se sont développées :

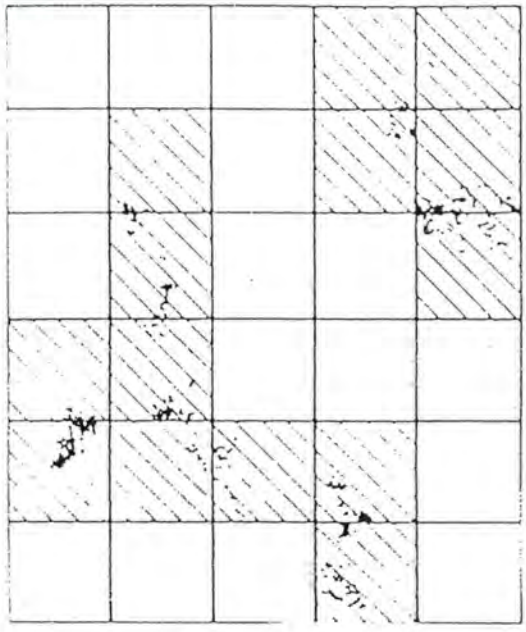
1. la méthode "box-counting"
2. la relation masse-rayon
3. la fonction de corrélation.

Dans le cadre de ce travail, je ne développe que la méthode utilisée dans le logiciel. C'est-à-dire la méthode "box-counting". Pour les personnes désireuses de connaître le fonctionnement des deux autres méthodes, je leur conseille de parcourir le mémoire de T. Reniers.

La méthode "box-counting" :

Elle se base sur la définition de la dimension de capacité. En effet, le principe utilisé consiste à recouvrir la figure de petits carrés (ou cubes pour un espace métrique de dimension trois) dont les côtés sont de même longueur. Expliquons la méthode sur un exemple concret afin de clarifier les principes de la technique utilisée.

Supposons un ensemble de points dans un plan et calculons sa dimension. Dans un premier temps, divisons le plan en carrés de longueur de côté égale à r :



Ensuite, comptons le nombre de carrés contenant au moins un point de l'ensemble. Soit N(r) ce nombre. Si N(r) vérifie la relation $N(r) \approx r^{-D}$ lorsque r varie, alors la distribution des points a une dimension fractale égale à D.

La validité de la méthode box-counting repose sur les théorèmes suivants, dont on peut trouver les démonstrations dans le mémoire de T. Reniers.

Théorème 1 :

Soit A, un ensemble compact d'un espace métrique (X,d),
Soit $\varepsilon_n = cr^n$ où c et r sont deux nombres réels tels que
$0 < r < 1$, $c > 0$ et $n = 1, 2, 3, ...$
Si

$$D = \lim_{n \to \infty} \frac{\ln N(A, \varepsilon_n)}{\ln \left(\frac{1}{\varepsilon^n}\right)}$$

alors D est la dimension fractale de capacité de A.

Théorème 2 :

Soit A un ensemble compact de l'espace métrique $\Re^n$ muni de la norme euclidienne. Recouvrons $\Re^n$ par des "boîtes" fermées et juxtaposées (des carrés si n = 2, des cubes si n = 3) dont les côtés ont une longueur $(1/2)^n$.
Soit N_n (A), le nombre de boîtes ayant un côté de longueur $(1/2)^n$ et qui ont une intersection non vide avec l'ensemble A.
Si

$$D = \lim_{n \to \infty} \frac{\ln N_n (A)}{\ln 2^n}$$

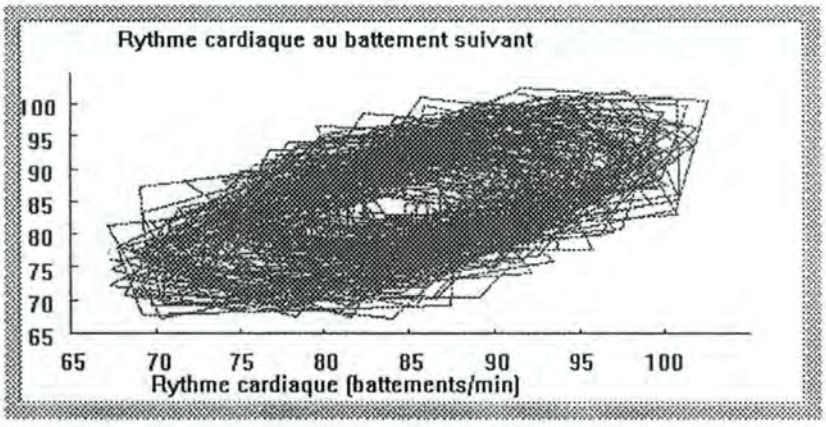
alors D est la dimension fractale de capacité de l'ensemble A.

L'utilisation que nous avons faite de la méthode "box-counting", plutôt qu'une autre, dans notre travail est motivée par le traitement d'images par ordinateur. En effet, il est efficace d'utiliser les pixels comme des carrés de longueur de côté très petite.

§4 Utilisation pratique des notions relatives aux fractales et à la dynamique chaotique :

Nous allons appliquer la théorie relative aux fractales et à la dynamique chaotique pour étudier le comportement du système dynamique obtenu à partir du rythme cardiaque. N'ayant à notre disposition que des données numériques exprimant les battements par minute par rapport au temps, nous allons effectuer en premier lieu un mapping.

Ce mapping consiste à représenter le rythme cardiaque dans son espace des phases. Autrement dit, nous cherchons à représenter l'évolution de la trajectoire définie par les données numériques. L'espace des phases aura pour coordonnées le rythme cardiaque (battements par minute) en abscisse et le rythme cardiaque suivant en ordonnée.



Le résultat de ce mapping permettra de savoir de quelle sorte est l'attracteur fourni par le rythme cardiaque du patient.

Si l'attracteur est un cycle, le patient possède un coeur qui bat périodiquement. Par contre, un attracteur étrange résulterait d'un coeur dont les battements ne sont pas réguliers et même très variables.

La dimension fractale sert d'indice de complexité d'un objet géométrique. Nous allons, ainsi, l'utiliser afin de connaître la complexité de l'attracteur obtenu par le mapping. Le calcul de cette dimension fractale utilisera la méthode "box-counting" sur l'image contenant le mapping du rythme cardiaque. Possédant la dimension fractale, nous serons au courant de l'importance de la variabilité que présente le coeur.

Chapitre 3 : Logiciel d'analyse du rythme cardiaque :

Ce chapitre explique les étapes qui m'ont permis de réaliser un logiciel d'analyse du rythme cardiaque. Ce logiciel utilise les notions décrites dans les deux chapitres précédents afin de déterminer les caractéristiques dynamiques du rythme cardiaque d'une personne au repos.

Dans un premier temps, le programme doit permettre d'obtenir le spectre de Fourier associé au signal envoyé par les battements du coeur. Dans un second temps, le programme doit permettre de représenter le mapping obtenu à partir du rythme cardiaque et donner la dimension fractale du résultat. Je rappelle que ce mapping consiste à représenter l'évolution de l'orbite décrite par les données numériques dans l'espace des phases associé au système dynamique que définissent les battements du coeur. Cet espace des phases est un espace à deux dimensions dont les abscisses représentent le rythme cardiaque (battements par minute) et les ordonnées le rythme cardiaque au battement suivant. Le résultat obtenu à partir du mapping consiste en un attracteur dont la dimension fractale détermine la complexité géométrique. La complexité géométrique est un facteur qui détermine l'importance de variabilité du rythme cardiaque.

A la base, le logiciel devait avoir à sa disposition des données numériques représentatives du rythme cardiaque d'une personne au repos. Nous espérions que ces données puissent être fournies par la clinique ***Sainte-Camille de Namur***. Malheureusement, il s'est avéré que les fichiers de valeurs numériques fournis par le logiciel ***Stratascan*** de la clinique possèdent un format classé top secret. Ce logiciel Stratascan est développé par la société américaine ***DELMAR AVIONICS*** de Californie. Sa filiale belge qui se situe à Zaventem n'ayant pas la permission de dévoiler le format des fichiers, je me suis adressé par fax à la firme américaine. Je n'ai reçu, à l'heure actuelle, aucune réponse de Delmar Avionics. De ce fait, afin de tester le logiciel d'analyse du rythme cardiaque, j'ai simulé des signaux et recueilli des données numériques dans des fichiers.

Le langage utilisé pour la réalisation du logiciel est le Borland C++ sous Windows orienté objet. Ce langage est très performant et permet d'utiliser toute la puissance de Windows. Son caractère orienté objet permet de profiter au maximum des relations d'héritage qui existent entre objets parents et enfants. Cet héritage est surtout utile au niveau des objets prédéfinis par le langage. En effet, travailler sous Windows

nécessite l'utilisation de fenêtres qu'il faut installer à l'écran, qu'il faut remplir, qu'il faut rafraîchir lorsque d'autres fenêtres se superposent,.... Toutes ces fonctions utilitaires sont déjà définies dans les objets prédéfinis par le langage. Dés lors afin de profiter de celles-ci par héritage, nous définissons les objets du programme comme des enfants des objets prédéfinis.

Ce chapitre se compose de quatre parties. Une première partie est consacrée à la spécification des fonctions nécessaires à l'analyse du rythme cardiaque. Une seconde partie reprend les diverses décisions de conception envisagées pour la réalisation du programme. La troisième partie dévoile les moyens pratiques utilisés afin de réaliser le logiciel sans s'éloigner trop de l'idée proposée lors de la conception. Enfin, la dernière partie dévoile les résultats obtenus à partir de divers tests. Ces tests sont effectués sur des fichiers de données numériques créés personnellement et reflétant de la meilleure manière possible des rythmes soit périodiques, soit très variables.

§1 Spécification des fonctions nécessaires à l'analyse du rythme cardiaque :

A) Calcul du spectre de Fourier :

- Arguments : Le calcul du spectre de Fourier s'effectue sur un ensemble de données numériques réelles $\{x_1,...,x_n\}$ situé dans un fichier.

- Résultats : Le spectre de Fourier $\{\tilde{x}_1,...,\tilde{x}_n\}$ de l'ensemble de données initiales $\{x_1,...,x_n\}$.

- Règles de traitement :

  * Préconditions : Les données numériques réelles de départ doivent être entières. En effet, le nombre de battements de coeur par minute ne peut avoir de valeurs décimales.

  * Postconditions : La relation qui existe entre $\tilde{x}_i$ et x_i est déterminée par l'expression suivante :

$$\tilde{x}_k = \left| \sum_{j=1}^{n} x_j \, e^{2\Pi ijk/n} \right|$$

En effet, le spectre de Fourier est le module de la transformée de Fourier de l'ensemble de départ.

B) Calcul de la dimension fractale du mapping :

- Arguments : Un mapping $\{m_1,...,m_{n-1}\}$ calculé à partir des valeurs numériques $\{x_1,...,x_n\}$.

- Résultats : La dimension fractale du mapping. Cette dimension est un nombre réel reflétant la complexité géométrique du mapping.

- Règles de traitement :

  * Préconditions : Le mapping doit être affiché à l'écran.

  * Postconditions : La dimension fractale du mapping est obtenue en utilisant la méthode "box-counting" sur la représentation du mapping à l'écran en considérant les pixels comme des carrés de très petite longueur.

C) Affichage du mapping :

- Arguments : Le calcul du spectre de Fourier s'effectue sur un ensemble de données numériques réelles $\{x_1,...,x_n\}$ situé dans un fichier.

- Résultats : Le mapping $\{m_1,...,m_{n-1}\}$ est affiché à l'écran dans un repère d'axes représentant l'espace des phases associé au rythme cardiaque. En abscisse se trouve le rythme cardiaque et en ordonnée le rythme cardiaque à l'instant suivant.

- Règles de traitement :

  * Préconditions : Les données numériques réelles de départ doivent être entières. En effet, le nombre de battements de coeur par minute ne peut avoir de valeurs décimales.

  * Postconditions : Le mapping est obtenu à partir de l'ensemble des données initiales de la manière suivante :

$$m_1 = (x_1 , x_2)$$
$$m_2 = (x_2 , x_3)$$
$$\vdots$$
$$m_{n-1} = (x_{n-1} , x_n)$$

D) Affichage du rythme cardiaque :

- Arguments : Le calcul du spectre de Fourier s'effectue sur un ensemble de données numériques réelles $\{x_1,...,x_n\}$ situé dans un fichier.

- Résultats : L'affichage du rythme cardiaque à l'écran.

- Règles de traitement :

  * Préconditions : Les données numériques réelles de départ doivent être entières. En effet, le nombre de battements de coeur par minute ne peut avoir de valeurs décimales.

  * Postconditions : Le repère d'axes dans lequel on représente le rythme cardiaque a pour abscisse le temps et pour ordonnées le rythme cardiaque. L'ensemble des couples de nombres à afficher à l'écran est obtenu de la manière suivante :

$$p_1 = (1 , x_1)$$
$$\vdots$$
$$p_n = (n , x_n).$$

E) Affichage du spectre de Fourier :

- Arguments : Le spectre de Fourier $\{\tilde{x}_1,...,\tilde{x}_n\}$ calculé à partir des valeurs numériques $\{x_1,...,x_n\}$.

- Résultats : Le spectre de Fourier est affiché à l'écran.

- Règles de traitement :

  * Préconditions : Les valeurs numériques $\{x_1,...,x_n\}$ à la base du calcul du spectre doivent représenter le rythme cardiaque à chaque seconde.

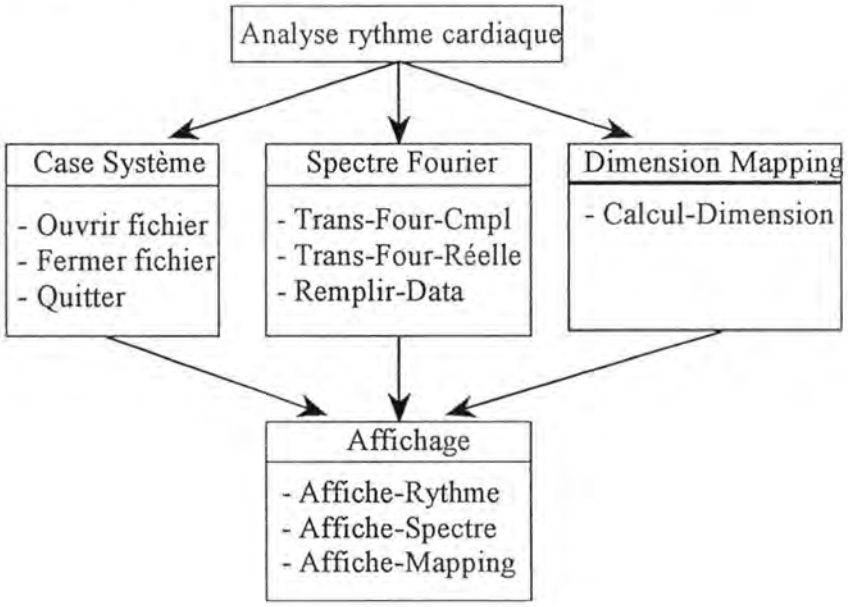
  * Postconditions : Le spectre de Fourier est affiché dans un repère d'axes où l'amplitude est représentée en fonction de

la fréquence. Les couples de nombres qui représentent les points à afficher sont obtenus de la manière suivante :

$$f_1 = (1/n , \tilde{x}_1)$$
$$f_2 = (2/n , \tilde{x}_2)$$
$$\vdots$$
$$f_n = (n/n , \tilde{x}_n)$$

§2 Décisions de conception :

La conception va être basée sur l'arbre des modules suivant :



La modularisation présentée dans ce graphe a pour objectif d'éviter les redondances dans le travail à effectuer et de créer des ensembles de fonctions qui pourront être utilisées indépendamment du logiciel moyennant un nombre minimal de modifications. Présentons l'utilité de chaque module indépendamment des autres.

A) Module "Case-Système" :

Ce module joue un peu le rôle des cases système que l'on retrouve dans le coin supérieur gauche des fenêtres de Windows. En effet, dès le lancement du programme, il permet à l'utilisateur de choisir entre trois possibilités :

→ soit ouvrir un fichier : 1
→ soit fermer le fichier ouvert précédemment : 2
→ soit quitter l'application : 3.

• Choix 1 :

Le choix 1 lance l'exécution de la fonction ouvrir-fichier dont voici la spécification :

* Préconditions : Il ne faut pas qu'un fichier soit déjà ouvert par l'application. Cette restriction n'a qu'un seul objectif qui est d'éviter d'oublier de fermer un fichier lorsque l'on quitte l'application. L'idée de base est : un seul fichier ouvert ⇒ un seul fichier à fermer.

* Postconditions : L'effet de la fonction est d'ouvrir le fichier désigné par l'utilisateur. Ce fichier contient les données numériques représentant le rythme cardiaque. Ensuite, elle appelle la fonction affiche-rythme (qui sera spécifiée dans le module affichage). Enfin, elle empêche l'utilisateur, dès l'ouverture du fichier, d'ouvrir un second fichier ou de quitter l'application sans passer par le choix 2.

• Choix 2 :

Le choix 2 revient à demander l'exécution de la fonction fermer-fichier dont voici la spécification :

* Préconditions : Cette fonction n'est pas exécutable si le choix 1 n'a pas été sélectionné avant. Elle nécessite, en effet qu'un fichier soit déjà ouvert pour être exécutée.

* Postconditions : Cette fonction ferme le fichier ouvert par l'utilisateur lors du choix 1. Elle permet, ensuite, à l'utilisateur de pouvoir choisir à nouveau parmi les choix 1 et 3.

• Choix 3 :

Enfin le choix 3 permet à l'utilisateur de quitter l'application. Il n'est possible de quitter l'application que suite à des séquences (choix 1 , choix 2) ou en choisissant le choix 3 dès le lancement de l'application.

B) Module Spectre-Fourier :

Ce module englobe tous les traitements à effectuer sur les données numériques afin d'en obtenir le spectre de Fourier. Il est basé sur les algorithmes décrits dans le chapitre 1. Dans un premier temps, la fonction remplir-data est exécutée :

• remplir-data :

* Préconditions : L'ensemble des données numériques $\{x_1,...,x_n\}$ sur lequel on doit effectuer le traitement pour obtenir le spectre doit être fourni.

* Postconditions : Cette fonction a pour but de créer un tableau data qui possède les caractéristiques exigées par les algorithmes du chapitre 1. Elle doit en premier lieu déterminer la puissance de 2, soit m, qui est la plus proche du nombre n de données et qui soit supérieure ou égale à n. Ensuite, elle doit créer un tableau de taille m et le remplir avec les valeurs numériques. Enfin, elle doit, éventuellement, le compléter avec des valeurs nulles si la taille exigée m est supérieure au nombre n de valeurs.

Une fois le tableau data créé et rempli avec les données, il suffit d'appliquer les algorithmes de calcul de la transformée de Fourier. Ce calcul exige l'exécution de la fonction trans-Four-réelle qui, elle-même, fait appel à trans-Four-cmpl.

• trans-Four-réelle (data , n , signe) :

Cette fonction correspond à la routine FFT appliquée à une fonction réelle discrétisée. Cette routine est décrite au chapitre 1.

* Préconditions : - data doit être un tableau de données réelles contenant les valeurs de discrétisation de la fonction pour laquelle on désire calculer la transformée de Fourier.

- n est la demi-dimension du tableau data.

- signe vaut 1 pour le calcul de la transformée de Fourier et -1 pour calculer la transformée inverse de Fourier.

* Postconditions : Cette fonction retourne dans data la demi-transformée de Fourier rangée selon l'ordre expliqué au chapitre 1.

• trans-Four-cmpl (data , n , signe) :

Cette fonction correspond à la routine FFT appliquée à une fonction complexe discrétisée. L'algorithme de cette fonction est analysé au chapitre 1.

* Préconditions : - data doit être un tableau de données réelles contenant les valeurs de discrétisation de la fonction pour laquelle on désire calculer la transformée de Fourier.
- n est la demi-dimension du tableau data.
- signe vaut 1 pour le calcul de la transformée de Fourier et -1 pour calculer la transformée inverse de Fourier.

* Postconditions : Cette fonction retourne dans data la transformée de Fourier dans son entièreté et rangée selon l'ordre décrit au chapitre 1.

Une fois la transformée calculée, le module Spectre-Fourier fait appel à la fonction affiche-spectre du module Affichage qui calcule le spectre de Fourier et l'affiche automatiquement. Cette fonction est spécifiée dans le module en question.

C) Module Dimension-Mapping :

En premier lieu, ce module fait appel à la fonction affiche-mapping du module Affichage. Une fois l'exécution de affiche-mapping terminée, il calcule la dimension fractale de l'image qui se trouve à l'écran. Pour ce faire, il suffit de lancer l'exécution de la fonction calcul-dimension. Cette fonction est basée sur la méthode "box-counting" décrite au chapitre 2 et

utilise l'algorithme défini par T. Reniers dans son mémoire. Il sera re-programmé au sein du programme.

- calcul-dimension :

  * Préconditions : Cette fonction requiert une image à l'écran représentant le mapping. Les axes de l'espace des phases, ainsi que les légendes doivent posséder une couleur différente de celle du graphe proprement dit.

  * Postconditions : Cette fonction applique la méthode "box-counting" en considérant un pixel comme un carré dont la longueur des côtés est très petite. Elle est basée sur l'idée de comptabiliser les pixels de l'image correspondant à la région occupée par le mapping. Sachant que la couleur des axes et des légendes est bleue et que celle du mapping est rouge, il suffit d'appliquer la technique suivante :

    → lire chaque pixel de l'image et compter le nombre de pixels rouges (count).
    → déterminer la longueur du côté du carré de recouvrement en nombre de pixels (cote).
    → déterminer la dimension fractale (dim = log (count) / log (cote)).

L'algorithme plus précis qui calcule la dimension fractale est décrit clairement ci-dessous :

```
count, cote, i, j, imin, imax, jmin, jmax, rr, rb  :  integer ;
dim : real ;
Fonction calcul-dimension ;
   count : = 0 ;
   imin : = 1000 ;
   jmin : = 1000 ;
   imax : = 0 ;
   jmax : = 0 ;
   pour i = 1 à rr faire
     pour j = 1 à rb faire
        lire le pixel de coordonnées ( i , j ) ;
        si la couleur du pixel est rouge alors
           count : = count + 1 ;
           imin : = min ( i , imin ) ; jmin : = min ( j , jmin ) ;
           imax : = max ( i , imax ) ; jmax : = max( j , jmax ) ;
     fin de faire ;
   fin de faire ;
```

cote : = max (imax - imin , jmax - jmin) + 1 ;
dim : = log (count) / log (cote) .

où

- rr et rb sont respectivement la dernière abscisse et ordonnée de la fenêtre de Windows.
- imin et imax sont respectivement la plus petite et la plus grande abscisse du mapping.
- jmin et jmax sont respectivement la plus petite et la plus grande ordonnée du mapping.

D) Module Affichage :

Ce module s'occupe des divers affichages à l'écran qu'exige le logiciel. Selon le cas, il affiche le mapping, le spectre ou encore le rythme cardiaque.

- affiche-rythme :

  * Préconditions : Les données numériques $\{x_1,...,x_n\}$ contenues dans le fichier choisi par l'utilisateur sont les uniques valeurs nécessaires pour l'affichage du rythme cardiaque.

  * Postconditions : L'affichage du rythme cardiaque en fonction du temps est le résultat fourni par affiche-rythme. Les axes et les légendes sont affichés en couleur bleue tandis que le rythme cardiaque proprement dit est affiché en rouge.

- affiche-spectre (data) :

  * Préconditions : Le tableau data fourni par le module Spectre-Fourier contient les valeurs numériques nécessaires à l'affichage du spectre.

  * Postconditions : - Les coordonnées des points à afficher doivent être calculées à partir des valeurs du tableau data.
  - Les ordonnées des points obéissent à la relation suivante :
  → ordonnée du premier point : | data (1) |
  → le $i^{ème}$ point a pour ordonnée :

$$\forall i : 1 < i < \frac{n}{2} \sqrt{data^2(2i - 1) + data^2 (2i)}$$

$$\rightarrow \text{ordonnée du } \frac{n}{2}^{ème} \text{ point : } | data (2) |$$

- Ces règles de calcul sont définies à partir de l'ordre de rangement des résultats de la transformée de Fourier de $\{x_1,...,x_n\}$ dans le tableau data. Cet ordre a été décrit dans le chapitre 1.
- Le spectre vérifie la symétrie suivante : $\tilde{x}_i = \tilde{x}_{n-i}$ $\forall i$ tel que $1 \leq i \leq n$. Grâce à cette symétrie, l'affichage du demi-spectre est suffisant pour effectuer une analyse de la dynamique du rythme cardiaque.
- L'abscisse du $i^{ème}$ point vaut (i - 1) / n. Ces valeurs correspondent aux valeurs des fréquences que l'on obtient en calculant la transformée de Fourier. Les règles de calcul sont décrites dans le chapitre 1.
- Les valeurs de l'amplitude du processus physique auquel correspond le rythme cardiaque n'ont pas une importance primordiale dans l'analyse du rythme cardiaque. Pour faciliter l'affichage du spectre, nous préférons considérer que cette amplitude varie de 0 (pour le minimum) à 1 (pour le maximum).
- Enfin, les légendes et les axes du repère sont de couleur bleue tandis que le spectre est de couleur rouge.

• affiche-mapping :

* Préconditions : Les données numériques $\{x_1,...,x_n\}$ contenues dans le fichier choisi par l'utilisateur sont les uniques valeurs nécessaires pour l'affichage du mapping.

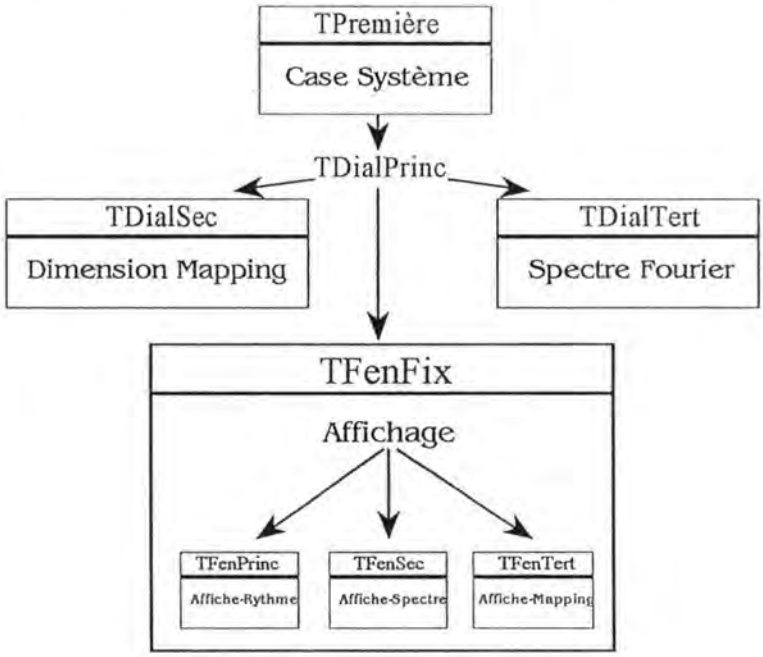
* Postconditions : - Une première étape dans l'affichage du mapping consiste à déterminer les valeurs maximales et minimales du rythme cardiaque. En effet, elles seront respectivement les valeurs maximales et minimales des abscisses et des ordonnées.
- Une seconde étape consiste en l'affichage du mapping. Les coordonnées des points sont obtenues directement à partir des valeurs numériques du rythme cardiaque. En effet, pour le $i^{ème}$ point, les coordonnées sont (x_i , x_{i+1}), avec $1 \leq i \leq n - 1$.

- La couleur des axes et des légendes est le bleu tandis que celle du mapping est rouge. Cette différence de couleur permet le calcul de la dimension fractale du mapping.

§3 Implémentation du logiciel : Utilisation du Borland C+ + sous Windows :

Borland C++ sous Windows orienté objet est un langage très performant et permet de profiter au maximum des services offerts par Windows. Je vais expliquer le choix des différents objets créés dans le cadre de ce projet et montrer l'analogie avec la modularisation envisagée au paragraphe précédent.

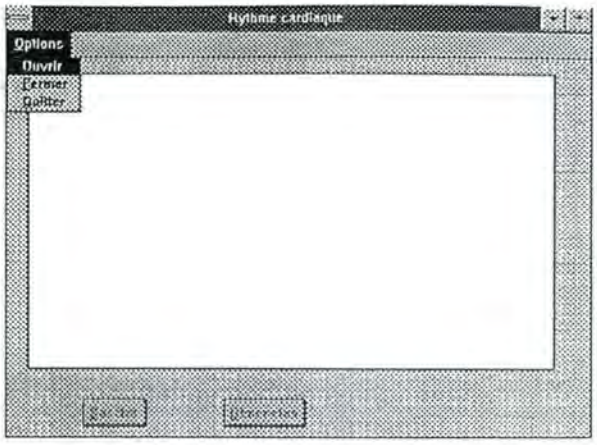
Le graphique suivant représente les différents objets utilisés dans le programme et montre la similitude avec la modularisation de la conception.



Expliquons, à présent, les divers objets repris dans ce graphique ainsi que leur utilité dans le programme.

A) La fenêtre TPremière :

C'est la fenêtre principale du projet. Sa création permet l'initialisation de l'application. Elle possède un menu déroulant. Ce menu offre les options suivantes :



Chaque option joue un rôle analogue au rôle imposé aux choix dans le module Case-Système. En effet, choisir **ouvrir** a pour résultat :

1°) Choisir le fichier à ouvrir grâce à la boîte de dialogue prédéfinie par Windows.
2°) Ouvrir le fichier choisi.
3°) Griser les options **ouvrir** et **quitter** pour forcer l'utilisateur à fermer le fichier qu'il vient d'ouvrir grâce à l'option **fermer**.
4°) Provoquer l'affichage du rythme cardiaque dans la fenêtre TFenPrinc.

Tandis que le choix de l'option **fermer** permettra de :

1°) Fermer le fichier qui a été ouvert par l'option ouvrir précédemment activée.
2°) Rafraîchir la fenêtre TFenPrinc.
3°) Dégriser les options ouvrir et quitter pour les rendre de nouveau accessibles à l'utilisateur.
4°) Détruire le tableau dans lequel les valeurs numériques représentant le rythme cardiaque ont été stockées. L'explication de l'existence de ce tableau est donnée dans la description de TDialprinc.

Enfin l'option **quitter** permet de détruire les fenêtres et de sortir de l'application.

Cette fenêtre principale cède la main à une boîte de dialogue TDialprinc dès que l'on a choisi l'option ouvrir. Cette boîte de dialogue va permettre à l'utilisateur d'effectuer l'analyse de son choix du rythme cardiaque.

B) La boîte de dialogue TDialprinc :

Cette boîte de dialogue a la structure de la boîte de dialogue présente sous le menu principal de TPremière.

Elle est composée :

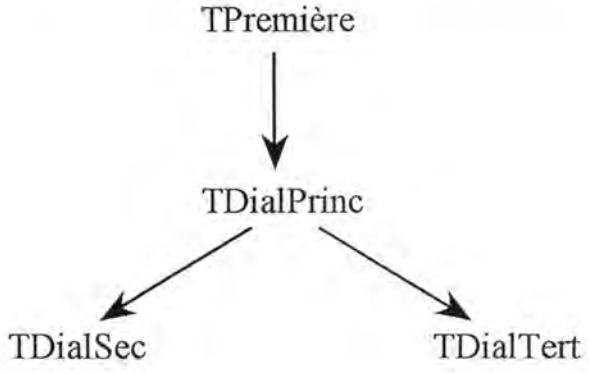
- d'une fenêtre sur laquelle s'effectue l'affichage du rythme cardiaque choisi par l'utilisateur.
- d'un bouton de commande **spectre** qui permet d'obtenir le spectre de Fourier du rythme cardiaque.
- d'un bouton de commande **dimension** qui permet d'obtenir le mapping relatif au rythme cardiaque et la dimension fractale de ce mapping.

TDialprinc est affichée dès la création de TPremière, mais ses boutons de commande ne peuvent être utilisés que lorsque l'option **ouvrir** du menu principal a été choisie. En effet, les boutons de commande sont grisés dès leur création et dégrisés par l'option **ouvrir**.

Lorsque l'utilisateur ferme le fichier contenant le rythme cardiaque, l'utilisation des boutons de commande n'a plus de sens. Ils sont grisés par l'option **fermer** du menu principal jusqu'au prochain choix de l'option **ouvrir**.

TDialprinc permet à l'utilisateur d'obtenir l'analyse dynamique de son choix du rythme cardiaque. Si le spectre est exigé, alors elle fait appel à une boîte de dialogue TDialtert qui fournira un affichage du spectre de Fourier pour le rythme cardiaque en question. Tandis que si l'utilisateur désire visualiser le mapping et connaître sa dimension fractale, TDialprinc fait appel à la boîte de dialogue TDialsec qui s'exécutera.

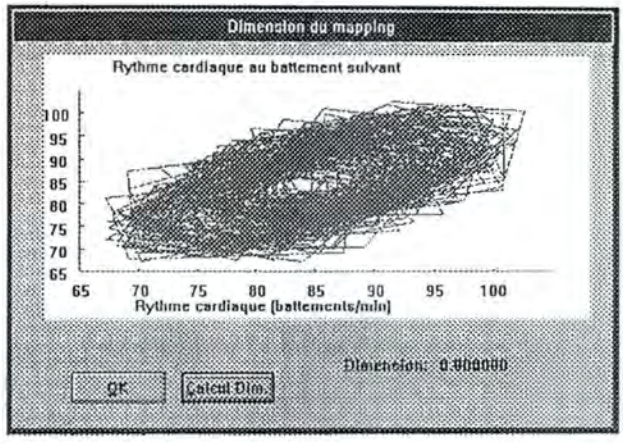
Tous ces objets sont reliés entre eux par des pointeurs :



De ce fait, les deux boîtes de dialogue TDialsec et TDialtert ne savent pas profiter des éléments de TPremière tels que le fichier contenant les données du rythme cardiaque. Pour résoudre ce problème, TDialprinc possède un tableau dans lequel elle recopie les données numériques. Elle possède, aussi, une variable entière où l'on stocke le nombre de données du fichier. TDialsec et TDialtert ont accès aux valeurs numériques grâce aux pointeurs les reliant à TDialprinc.

C) La boîte de dialogue TDialsec :

Cette boîte de dialogue possède la structure suivante :



Elle se compose de :

- une fenêtre TFensec sur laquelle le mapping est affiché.
- un bouton de commande OK qui permet à l'utilisateur de quitter la boîte TDialsec et revenir enTDialprinc.
- un bouton de commande **calcul-dim** qui permet à l'utilisateur d'avoir la dimension fractale du mapping.
- une zone statique qui reçoit le résultat du calcul de la dimension fractale du mapping.
- une zone d'édition contenant "dimension" donne la signification du résultat qui s'affiche dans la zone statique.

TDialsec est créée dès que l'utilisateur appuie sur le bouton de commande dimension de TDialprinc. C'est une boîte de dialogue modale, ce qui signifie que pour la quitter et revenir à TDialprinc, il est obligatoire d'appuyer sur le bouton OK.

L'affichage du mapping dans la fenêtre TFenSec se fait automatiquement au moment de la création de la boîte TDialsec. Par contre, le calcul de la dimension fractale est laissé au bon vouloir de l'utilisateur. S'il désire calculer la dimension, il lui suffit d'appuyer sur le bouton de commande calcul-dim. Le résultat de ce calcul sera automatiquement affiché dans la zone statique se situant à droite de la zone d'édition intitulée "dimension".

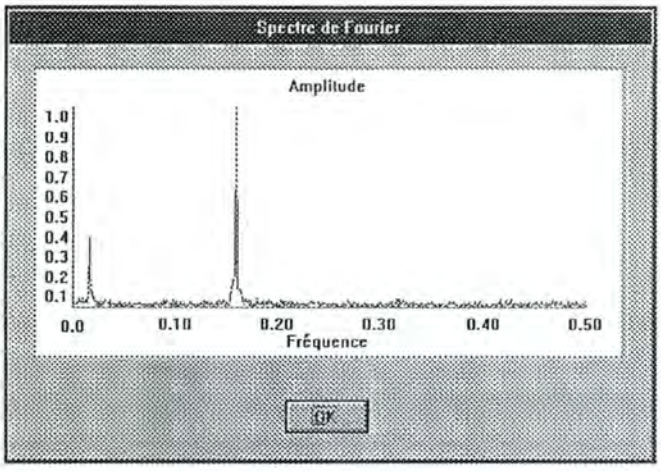
L'affichage est d'abord effectué sur un bitmap en mémoire. Ensuite, ce bitmap est affiché sur la fenêtre. Seul un gain de temps lors de l'affichage a motivé l'utilisation d'un bitmap. Le graphe du mapping dans le bitmap est réalisé par TDialsec en utilisant le tableau et la variable réelle de TDialprinc

qui contiennent respectivement les données numériques du rythme cardiaque et le nombre de ces données.

Nous remarquons que TDialsec est très semblable au module Dimension-Mapping du paragraphe précédent.

D) La boîte de dialogue TDialtert :

Cette boîte de dialogue possède la structure suivante :



Elle se compose de :

- une fenêtre TFenTert qui reçoit le graphe du spectre de Fourier.
- un bouton de commande OK qui permet de quitter TDialtert pour revenir à TDialprinc.

C'est une boîte de dialogue modale. Elle est créée lorsque l'utilisateur choisit d'appuyer sur le bouton de commande Spectre de TDialprinc. Lors de sa mise en place à l'écran, elle remplit un tableau **data** à partir du tableau de TDialprinc. Ce tableau **data** est rempli de sorte à respecter les exigences posées par l'algorithme de calcul de la transformée de Fourier décrit au chapitre 1. Ensuite, elle calcule la transformée de Fourier du rythme cardiaque en se servant de **data**. Enfin, elle installe la fenêtre TFenTert et lui impose de dessiner le spectre à partir de la transformée de Fourier stockée dans **data**.

Cette description de TDialtert permet de vérifier le similitude qui existe entre TDialtert et le module Spectre-Fourier de la conception.

E) La fenêtre TFenFix :

Grâce à l'objet TFenFix, nous possédons des fenêtres sans titre et sans bords. Ces fenêtres n'ont pour utilité, dans mon programme, que de servir de support pour les divers affichages à réaliser. Nous pouvons associer TFenFix au module Affichage de la conception.

Nous définissons trois fenêtres filles de TFenFix; chacune de ces fenêtres sert à afficher un graphe spécifique :

- **La fenêtre TFenPrinc :**

C'est la fenêtre que nous retrouvons dans la boîte de dialogue TDialprinc. Son rôle est de recevoir la représentation du rythme cardiaque choisi par l'utilisateur chaque fois qu'il sélectionne l'option **ouvrir** du menu principal de TPremière.

Pour tracer le graphe, elle utilise le tableau que remplit TDialprinc avec les valeurs du fichier contenant le rythme cardiaque. Il faut noter que le fichier possède une valeur numérique par ligne ! De plus, les données doivent correspondre au rythme cardiaque d'une personne au repos. Pour cette raison, nous considérons que le rythme le plus faible est de 60 battements / minute et le rythme le plus fort de 150 battements / minute. De plus, les données doivent correspondre à un rythme cardiaque mesuré sur une période de 15 minutes.

TFenPrinc est rafraîchie chaque fois que l'utilisateur décide de fermer le fichier contenant les valeurs numériques qui sont à l'origine de l'analyse en cours. Je considère qu'à partir de ce moment-là, l'analyse est terminée et que nous pouvons passer à l'analyse suivante, relative à un autre rythme cardiaque.

- **La fenêtre TFenSec :**

Elle se situe sur la boîte de dialogue TDialsec et contiendra la représentation du mapping. En fait, elle reçoit le bitmap sur lequel nous avons représenté le mapping.

Lorsque l'utilisateur choisit de calculer la dimension fractale de l'attracteur, la fenêtre TFenSec est parcourue de long en large pour compter le nombre de pixels rouges et déterminer la grandeur du côté du carré dans lequel se situe l'attracteur. Cette façon de procéder a été expliquée dans la partie conception.

• **La fenêtre TFenTert :**

Cette fenêtre appartient à la boîte de dialogue TDialtert. C'est la fenêtre sur laquelle le spectre de Fourier est affiché.

Les données utilisées pour le graphique sont contenues dans la tableau **data** de TDialtert. Elles subissent certaines modifications afin d'obtenir les valeurs correspondant au spectre. Les opérations à effectuer sont celles qui sont décrites dans la partie conception.

Il faut rappeler que grâce à la symétrie du spectre de Fourier, $\tilde{x}_i = \tilde{x}_{n-i}$, la représentation du demi-spectre est suffisante pour l'analyse du rythme cardiaque.

Afin d'obtenir une représentation significative du spectre, nous traçons le spectre moins sa valeur moyenne. Cette décision a été motivée par l'utilisation du Borland C++. En effet, représenter un graphique sur une fenêtre consiste à faire correspondre une dimension physique avec une dimension logique :

→ si la fenêtre a une longueur de 400 pixels, nous lui faisons correspondre une longueur logique de 1.
→ si la fenêtre a une hauteur de 200 pixels, nous lui faisons correspondre une hauteur logique de (max(spectre) - min(spectre)).

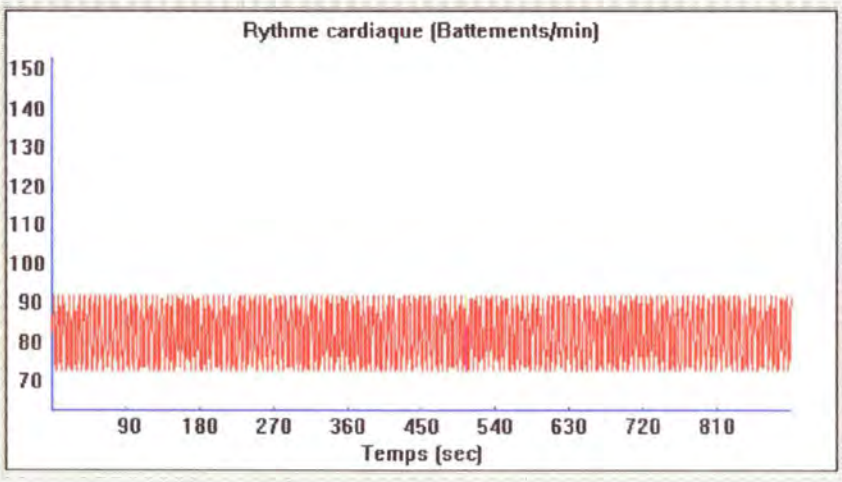
Malheureusement, la valeur de la hauteur logique est trop grande que pour avoir une représentation physique significative. Seules les valeurs extrêmes, i.e. le maximum et le minimum, sont perceptibles sur le graphique. Une représentation, dans de telles conditions, ne permet pas de visualiser les harmoniques provoquées par de petites perturbations. En représentant le spectre moins sa valeur moyenne, il existe une meilleure harmonie entre dimensions physique et logique.

§4 Les tests effectués sur le produit fini :

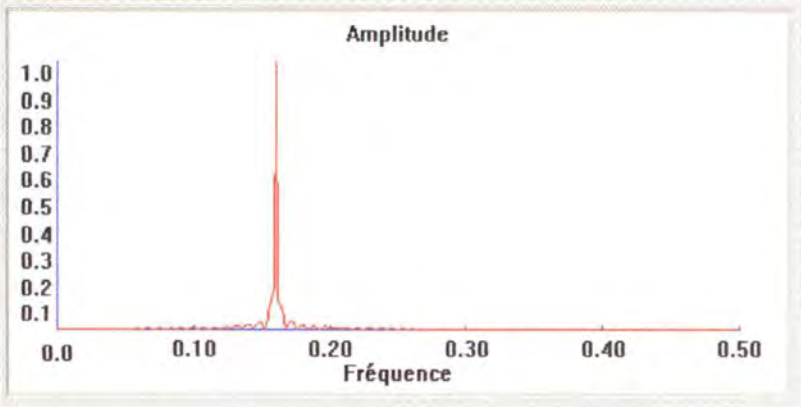
Afin de tester l'efficacité et l'exactitude du logiciel, j'ai préparé quelques fichiers de données représentant un rythme périodique,un rythme périodique subissant une faible perturbation ou encore un rythme aléatoire.

A) Analyse d'un rythme périodique :

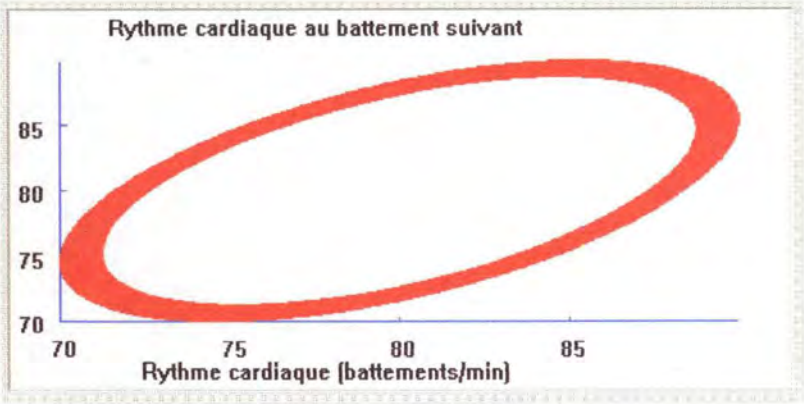
La simulation d'un rythme cardiaque périodique a été possible grâce à la fonction sin (x) * 10 + 80. Faisant varier x de 1 à 900 et stockant les résultats de la fonction dans un fichier avec une valeur par ligne, nous obtenons les valeurs numériques d'un coeur périodique.



Cette fonction est périodique de période 2Π. Son spectre doit, donc, posséder un pic dans les environs de 1/ 2Π. Le résultat obtenu avec le logiciel est très satisfaisant :



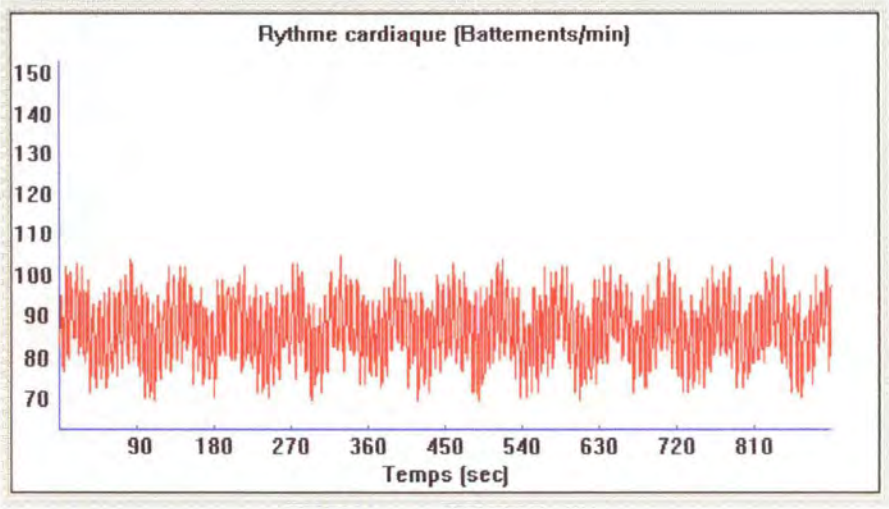
Le mapping résultant de l'analyse d'un tel signal doit ressembler à un cycle. L'attracteur, résultat du mapping dans le logiciel, en est très proche :



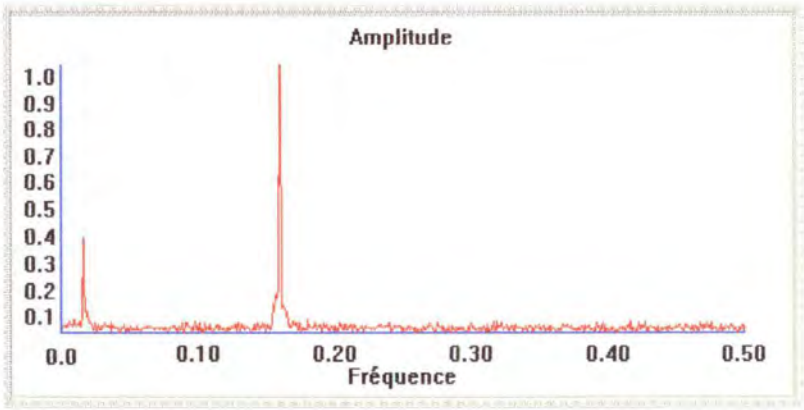
Sa dimension fractale est de 1.546157.... Elle reflète la complexité géométrique de l'attracteur. Elle est supérieure à la dimension d'une ligne, mais elle ne démontre pas une variabilité excessive dans le rythme.

B) Analyse d'un rythme périodique avec perturbation :

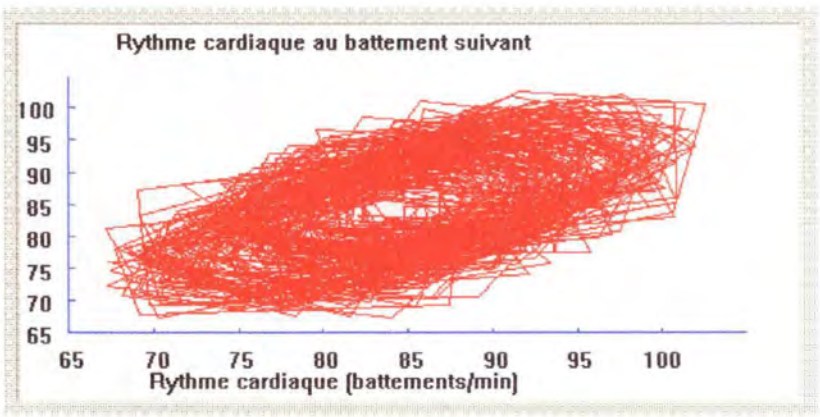
La simulation d'un tel rythme est réalisable à l'aide de la fonction suivante: sin (x / 10) * 5 + sin (x) * 10 + 80 + random (10) où la fonction random permet d'obtenir aléatoirement un nombre entre 1 et 10. Faisant varier x de 1 à 900 et stockant les résultats de la fonction dans un fichier, nous obtenons le rythme cardiaque suivant :



Le spectre de Fourier d'un tel rythme doit posséder un pic dans les environs de 1 / 2Π correspondant à la période principale du signal et un second pic moins important correspondant à l'harmonique provoquée par sin(x / 10). Le résultat pratique vérifie les conclusions théoriques :



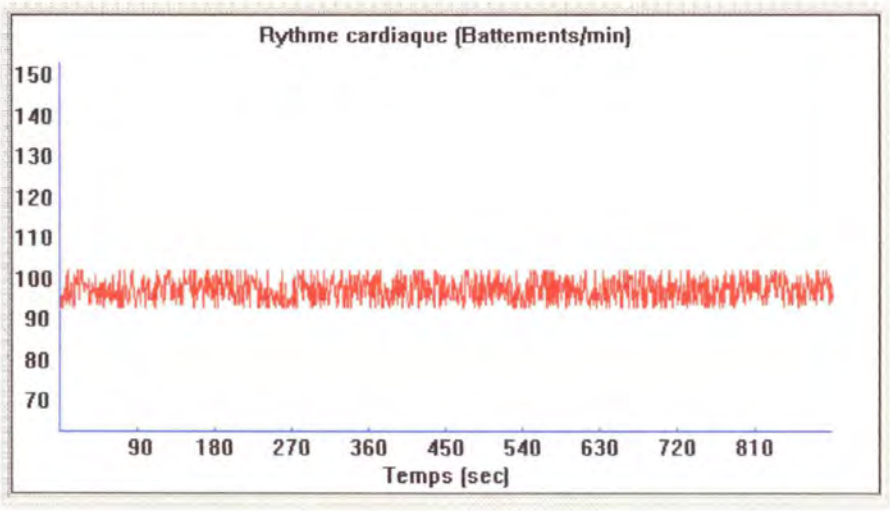
Le mapping résultant possède une structure géométrique plus complexe reflétant la plus grande variabilité du rythme cardiaque :



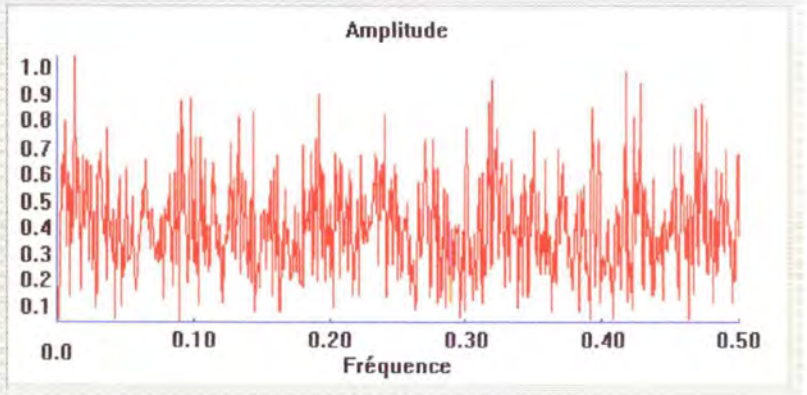
La dimension fractale confirme cette affirmation : dimension = 1.717869.

C) Analyse d'un rythme aléatoire :

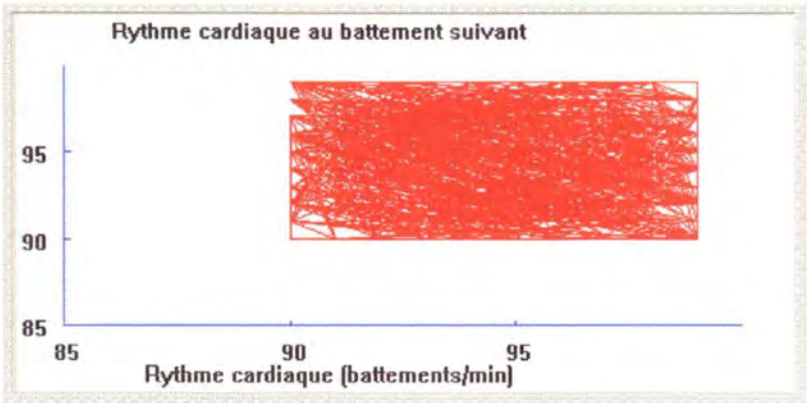
La fonction permettant la simulation d'un tel rythme est la suivante : 90 + random (10). La fonction random est à l'origine de l'aléatoire qui existe dans le rythme :



Ce rythme ne possède aucune périodicité. Son spectre de Fourier est très large et son analyse ne fournit aucun renseignement :



Son mapping fournit un attracteur étrange très complexe qui confirme la grande variabilité du rythme cardiaque :



La confirmation de cette complexité est apportée par la dimension fractale de l'attracteur : 1.797640....Cette dimension est supérieure à celle de l'attracteur du rythme périodique perturbé dont la valeur est de 1.717869. En effet, dans ce rythme-ci, aucune période ne peut être décelée, d'où aucune présence de pseudo-cycle n'est possible dans l'attracteur. Les pseudo-cycles permettent de limiter l'espace utilisé par l'attracteur.

Conclusion :

De récentes analyses des représentations dans l'espace des phases d'un rythme cardiaque normal ont fourni des résultats qui ressemblent plus à un attracteur étrange qu'à l'attracteur périodique caractéristique d'un processus véritablement régulier. La dynamique du rythme cardiaque normal doit être chaotique.

Le mécanisme amenant le chaos du rythme cardiaque sain réside probablement dans le système nerveux. Le sinus du coeur (son régulateur naturel) reçoit des signaux de la partie non volontaire (végétatif) du système nerveux. Le système nerveux végétatif se décompose en sympathique et parasympathique. Le parasympathique diminue la fréquence de décharge des cellules du sinus cardiaque alors que le sympathique a l'effet inverse. Ces influences opposées, qui contrôlent le rythme cardiaque, expliquent ses variations chez les sujets sains. De telles conclusions furent confirmées lorsqu'on a constaté la réduction de la variabilité du rythme cardiaque survenant après une transplantation cardiaque, opération dans laquelle les fibres nerveuses végétatives sont coupées.

Divers travaux ont prouvé l'existence du chaos dans d'autres composants du système nerveux. Une dynamique chaotique offre plusieurs avantages fonctionnels : les systèmes chaotiques acceptent des environnements plus variés que les systèmes périodiques rigides, ils sont plus adaptables et plus souples. Cette plasticité est une adaptation favorable, car elle accepte l'imprévisible.

De nombreuses pathologies se traduisent par un comportement de périodicité plus marquée et une perte de variabilité. L'analyse de Fourier des enregistrements éléctrocardiographiques lors d'arythmies cardiaques très rapides qui amènent généralement l'arrêt du coeur, montre que même le coeur d'un homme mourant peut battre avec une grande régularité. En 1988, Ary Golberger et son équipe ont étudié les électrocardiogrammes de patients atteints d'une maladie cardiaque grave. Les résultats de cette analyse firent découvrir que la structure des battements de ces malades était, quelques minutes ou quelques mois avant l'accident cardiaque mortel, moins fluctuante que la normale. Dans certains cas, la variabilité de l'intervalle entre les battements diminuait; dans d'autres, des oscillations rapides et périodiques du rythme cardiaque apparaissaient puis disparaissaient brusquement.

Donc, certaines pathologies cardiaques se traduisent par des oscillations qui paraissent irrégulières mais dont une analyse soigneuse prouve la périodicité. Dans d'autres arythmies, les battements sont imprévisiblement erratiques. Aucune de ces pathologies à irrégularité n'a cependant la signature du chaos non linéaire, bien que les pulsations puissent paraître "chaotiques" au sens courant du terme.

Il faut cependant noter que certains physiologistes, conservateurs de la sagesse médicale classique, affirment que de tels résultats ne sont qu'illusion. Ils se basent sur le fait qu'une analyse déterministe n'est réalisable que lorsque toutes les variables du problème sont connues. Or, ce n'est pas le cas pour l'étude du rythme cardiaque !!!

Les analyses futures du rythme cardiaque vont utiliser de plus en plus le spectre de Fourier ou la représentation dans l'espace des phases. Le logiciel résultant de mon travail permet d'obtenir les résultats de ces deux analyses. Les tests effectués sur celui-ci n'ont pas la prétention de justifier les révélations de Ary Golberger et de son équipe. En effet, comme je l'ai expliqué au chapitre 2, je n'ai pas obtenu de données numériques représentatives du rythme cardiaque d'un patient en bonne santé. Malgré les efforts des infirmières du service cardiologie de la clinique Sainte-Camille, le format **top secret** des fichiers contenant les données fut un obstacle insurmontable. Afin de pallier ce problème, j'ai simulé divers rythmes cardiaques à l'aide de fonctions telles que"sinus" ou "random" qui fournit aléatoirement des valeurs. Ces tests ont révélé un bon fonctionnement du logiciel.

Certaines difficultées graphiques dues à l'utilisation du langage Borland C++ limitent les représentations des divers résultats. Une amélioration à ce niveau peut être envisagée.

Pour terminer, je tiens à signaler que grâce à un faible nombre d'améliorations, ce logiciel pourrait être inséré dans un logiciel tel que Stratascan. Cet ajout permettrait d'obtenir une analyse encore plus exhaustive du rythme cardiaque des patients. Il offrirait ainsi la possibilité de comparer les analyses classiques avec les nouvelles techniques envisagées et de pouvoir débattre sur des données numériques réelles au lieu de données théoriques.

Bibliographie :

1°) Chaos et fractales en médecine
Mémoire de T. Reniers
FNDP Namur, Licence-Maîtrise en Informatique,Septembre 1992

2°) Chaos et fractales en physiologie humaine
Ary Golberger, David Rigney, Bruce West
Pour la Science 150, 50 - 57, 1990

3°) Nonlinear Dynamics of the heartbeat
Ary Golberger, Valmik Bhargava, Bruce J. West, Arnold J. Mandell
Physica D 17, 207 - 214, 1985

4°) There is a theory of heart
Glass L. , Hunter P.
Physica D 43, 1 - 16, 1990

5°) Les objets fractals : 3° édition suivie de survol du langage fractal.
Mandelbrot B.
Flammarion, 1989

6°) Le chaos
James Crutchfield, Dayne Farmer, Norman Packard, Robert Shaw
Pour la science Février 1987

7°) Fractals in the physical Sciences
Takayasu H.
Manchester University Press 1990

8°) Model 563 Stratascan TM Holter Analysis System
Operator's manual AV PUB 90 - 9
Software Version A4 Preliminary June 28, 1991

9°) From clocks to chaos : The rythms of life
Glass Leon and Mackey C. Michael

10°) Numerical Recipes, The Art of Scientific Computing (Fortran Version)
Press W.H., Flannery B.P., Teukolsky Saul A., Vetterling W.T.
Cambridge University Press.

11°) Programmation Windows en Turbo C++ et borland C++
Leblanc Gérard
Editions EYROLLES, Paris, 1992

Annexe

Installation et utilisation du logiciel :

L'installation du logiciel est très simple. Il suffit de copier le fichier exécutable *rythme.exe* sur le disque dur. Ensuite, se placer dans Windows et ,à l'aide du gestionnaire de fichiers, lancer l'exécution du fichier *rythme.exe*.

L'utilisation du logiciel requiert les fichiers de données numériques des simulations de rythmes cardiaques. Il suffit de les copier sur le disque dur au moment de l'installation du logiciel.

Voici une rapide description des fichiers de données numériques :

- **périodique.dat** : il contient les données numériques relatives à la simulation d'un rythme cardiaque périodique. La fonction utilisée pour cette simulation est la suivante :
  sin (x)*10 + 80.

- **perturbe.dat** : il contient les données numériques d'un rythme périodique perturbé. Il est doublement perturbé, d'abord par une harmonique **sin (x/10)*5**, ensuite par **random (10)** qui fournit aléatoirement un nombre entre 0 et 10. La fonction utilisée pour cette simulation est la suivante :
  sin (x)*10 + 80 + sin (x/10)*5 + random (10).

- **aléatoire.dat** : il contient les données numériques relatives à un signal totalement aléatoire. Il ne possède aucun signe de périodicité. Il est le résultat de la fonction suivante :
  90 + random (10).

Listing du programme

```
# include <stdio.h>
# include <owl.h>
# include <dialog.h>
# include <LISTBOX.h>
# include <BUTTON.h>
# include <filedial.h>
# include "rythme.h"
# include <string.h>
# include <math.h>


TPremiere :: TPremiere(LPSTR titre):TWindow(NULL,titre)
{
 AssignMenu ("MENU_1");
 init = FALSE;
  Attr.X = 1;
  Attr.Y = 1;
  Attr.W = 605;
  Attr.H = 446;

  Dprinc = new TDialprinc (this,"DIALPRINC");
}

void TPremiere :: SetupWindow ()
 {
 /* Ouvrir la boite de dialogue principale */
  TWindow::SetupWindow ();
  GetApplication()-> MakeWindow(Dprinc);
 }

void TPremiere :: GetWindowClass (WNDCLASS & WinClass)
{
  TWindow::GetWindowClass (WinClass);
  WinClass.hIcon = LoadIcon (GetApplication () -> hInstance, "ICONE");
}

void TPremiere :: WMGetMinMaxInfo(RTMessage Msg)
 {
  LPPOINT lpPoint;

  lpPoint = (LPPOINT)Msg.LParam;
  lpPoint+=3;
  lpPoint-> x = 605;
  lpPoint-> y = 446;
  lpPoint+=1;
  lpPoint-> x = 605;
  lpPoint-> y = 446;
  }


void TPremiere :: Ouvrir ( RTMessage )
```

```
 {
  /* Griser les deux autres sous-menus */
   init = TRUE ;
   hmp = GetMenu(HWindow);
   hmo = GetSubMenu(hmp,0);

  /* Ouverture du fichier qui contient les donnees */
   strcpy(NomFich,"*.*");
   if (GetApplication()->ExecDialog(new TFileDialog(this,SD_FILEOPEN,NomFich)) ==
IDOK)
   {
    fd = fopen(NomFich,"r");
    ((TDialprinc*)Dprinc)->RemplirTableau(fd);
    EnableMenuItem(hmo,CM_OUVRIR,MF_GRAYED);
    EnableMenuItem(hmo,CM_QUITTER,MF_GRAYED);
    EnableWindow (((TDialprinc*)Dprinc) -> Bspectre -> HWindow, TRUE);
    EnableWindow (((TDialprinc*)Dprinc) -> Bdim -> HWindow, TRUE);
   }
 }

 void TPremiere :: Fermer ( RTMessage )
 {
  TFenFix * Fen;
  if (init)
   {
    Fen = ((TDialprinc*)Dprinc)-> Fprinc;
    ((TFenPrinc*)Fen) -> ok = FALSE;
    /* Fermer le fichier ouvert precedemment */
    fclose (fd);
    delete ((TDialprinc*)Dprinc)->tableau;
    /* Rendre actif les deux autres sous-menus */
    EnableMenuItem(hmo,CM_OUVRIR,MF_ENABLED);
    EnableMenuItem(hmo,CM_QUITTER,MF_ENABLED);
    EnableWindow (((TDialprinc*)Dprinc) -> Bspectre -> HWindow, FALSE);
    EnableWindow (((TDialprinc*)Dprinc) -> Bdim -> HWindow, FALSE);

    init = FALSE;
    InvalidateRect (((TFenPrinc*)Fen) -> HWindow, NULL, TRUE);
   }
 }


 void TPremiere :: Quitter ( RTMessage )
 {
  ((TDialprinc*)Dprinc)->CloseWindow();
  CloseWindow();
 }

 TPrapplic::TPrapplic(LPSTR nom,HANDLE hi,HANDLE hj,LPSTR com,int
nbr):TApplication(nom, hi, hj, com, nbr)
 {
```

```
}

void TPrapplic::InitMainWindow()
{
 MainWindow=new TPremiere("Rythme cardiaque");
}

/* Fonctions a definir pour la boite de dialogue principale */

TDialprinc :: TDialprinc(TWindowsObject * parent, LPSTR name) : TDialog (parent,name)
{
/* Creation des boutons et de la fenetre */
 Bspectre = new TButton (this,ID_BSPECTRE);
 Bdim = new TButton (this,ID_BDIM);
 Fprinc = new TFenPrinc (this,"",540,300);
}

void TDialprinc :: SetupWindow ()
{
 RECT rc;
 /* Mise au point de la boite de dialogue principale */
  TDialog :: SetupWindow();
  GetClientRect(HWindow,&rc);
  MoveWindow(HWindow,0,0,rc.right-rc.left,rc.bottom-rc.top,TRUE);
  SetWindowPos (Fprinc -> HWindow, 0,  22,  16, 0, 0, SWP_NOSIZE |
SWP_NOZORDER | SWP_NOACTIVATE| SWP_NOREDRAW);
  EnableWindow (Bspectre -> HWindow, FALSE);
  EnableWindow (Bdim -> HWindow, FALSE);

}

void TDialprinc :: RemplirTableau(FILE* fd)
{
  HDC hdc;
  int larg,haut;
  RECT rc;

  /* Remplissage du tableau avec les donnees  */
  long i ;
  double bat ;
  /* Determiner le nombre de donnees dans le fichier */
  nbr = 0 ;
  while (feof(fd) == 0)
   {
      nbr++;
      fscanf(fd,"%lf\n",&bat);
   };
  rewind(fd);
  /* Remplissage du tableau proprement dit */
```

```
  tableau = new double [nbr];
  i=0;
  while (feof(fd) == 0)
   {
       fscanf(fd,"%lf\n",&tableau[i]);
       i++;
   };
   /* Initialisation des elements pour TFenPrinc */
  ((TFenPrinc*)Fprinc)-> Init(tableau,nbr);
  ((TFenPrinc*)Fprinc)-> ok=TRUE;

  InvalidateRect (Fprinc -> HWindow, NULL, TRUE);
  }

 void TDialprinc :: TraitSpectre(RTMessage)
  {
    GetApplication()->ExecDialog(new TDialtert(this,"dialtert"));
  }

 void TDialprinc :: TraitDimension(RTMessage)
  {
   GetApplication()->ExecDialog(new TDialsec(this,"dialsec"));
  }

/* Fonction a definir pour la boite de dialogue qui va
  s'occuper du calcul de la dimension fractal */

 TDialsec :: TDialsec(TWindowsObject* Parent , LPSTR nom)
  : TDialog ( Parent, nom)
  {
   CalDim = new TButton(this,ID_CALCDIM);
   Fsec = new TFenSec(this,"",484, 234);
   Dime=new TStatic(this,ID_EDITDIM,10);
  }

 void TDialsec :: SetupWindow ()
  {
   char str[10];

   TDialog :: SetupWindow ();
   dimMapping=0;
   DesMap(((TDialprinc*)Parent)->tableau,((TDialprinc*)Parent)->nbr);
   sprintf(str,"%2.6f",dimMapping);
   Dime->SetText (str);

   SetWindowPos(Fsec->HWindow,0,28,12, 0, 0, SWP_NOSIZE | SWP_NOZORDER |
SWP_NOACTIVATE| SWP_NOREDRAW);
   }

 void TDialsec::Ok(RTMessage)
  {
```

```
  DeleteObject(hMapping);
  CloseWindow(IDOK);
 }

void TDialsec::DesMap(double* tableau, long num)
 {
  char str[10];
  double max=-1000,
         min=1000,
         borneinf,bornesup,aux;
  HDC hdc,hMemdc;
  HPEN stylo_orig,stylo_dessin,stylo_axes;
  int j;
 /* Calcul des limites presentes dans les donnees */
  for (j=0;j<num;j++)
   {
      if (tableau[j]<min)
        min = tableau[j];
      else if (tableau[j]>max)
        max = tableau[j];
   }
  aux=0;
  while (aux<min-5)
   {
     aux=aux+5;
   };
  borneinf=aux;
  bornesup=0;
  while (bornesup<max)
   {
      bornesup=bornesup+5;
   }
 /* Creation du Bitmap ou l'on dessine le mapping */
  hdc=GetDC(HWindow);
  hMemdc=CreateCompatibleDC(hdc);
  hMapping=CreateCompatibleBitmap(hdc,482,232);
  SelectObject (hMemdc, hMapping);
  PatBlt (hMemdc, 0, 0, 482, 232, WHITENESS);

 /* Affichage des legendes */
 /*  SetTextAlign (hdc, TA_CENTER);*/
  TextOut(hMemdc,60, 2, "Rythme cardiaque au battement suivant", 37);
  TextOut(hMemdc,80, 215, "Rythme cardiaque (battements/min)",33);
 /* Changements de systemes de coordonnees */
  SetViewportOrg(hMemdc,30,192);
  SetMapMode(hMemdc,MM_ANISOTROPIC);
  SetViewportExt(hMemdc,422,162);
  SetWindowExt(hMemdc,(bornesup-borneinf)*100,(-bornesup+borneinf)*100);
  SetWindowOrg(hMemdc,borneinf*100,borneinf*100);
 /* Creation du stylo pour les axes */
  stylo_axes=CreatePen(PS_SOLID,0,RGB(0,0,255));
```

```
    stylo_orig=SelectObject(hMemdc,stylo_axes);
   /* Trace des axes de coordonnees */
    MoveTo(hMemdc,borneinf*100,borneinf*100);
    LineTo(hMemdc,borneinf*100,bornesup*100);
    MoveTo(hMemdc,borneinf*100,borneinf*100);
    LineTo(hMemdc,bornesup*100,borneinf*100);
   /* Trace des reperes sur les axes */
    SetTextAlign (hMemdc,TA_CENTER);
    aux=borneinf;
    while(aux<bornesup)
     {
        MoveTo(hMemdc,aux*100,borneinf*100);
        LineTo(hMemdc,aux*100,borneinf*100+ 3.0*100*(bornesup - borneinf)/162.0);
      sprintf(str,"%3.0f",aux);
        TextOut(hMemdc,aux*100,borneinf*100- 10.0*(bornesup -
borneinf)/162.0*100,str,strlen (str));
        aux+=5;
     }
    aux=borneinf;
    while(aux<bornesup)
     {
      MoveTo(hMemdc,borneinf*100,aux*100);
        LineTo(hMemdc,borneinf*100+ 5.0*(bornesup - borneinf)/422.0*100,aux*100);
      sprintf(str,"%3.0f",aux);
        TextOut(hMemdc,borneinf*100 - 20.0*(bornesup - borneinf)/422.0*100,aux*100 +
5.0*(bornesup - borneinf)/162.0*100,str,strlen (str));
        aux+=5;
     }
   /* Creation du stylo pour le dessin */
    stylo_dessin=CreatePen(PS_SOLID,0,RGB(255,0,0));
    SelectObject(hMemdc,stylo_dessin);
   /* Trace du graphique proprement dit */
    MoveTo(hMemdc,tableau[0]*100,tableau[1]*100);
    for (j=1;j<num-1;j++)
     {
        LineTo(hMemdc,tableau[j]*100,tableau[j+1]*100);
     }
   /* Selectionner le stylo d'origine et detruire les autres */
    SelectObject(hMemdc,stylo_orig);
    DeleteObject(stylo_dessin);
    DeleteObject(stylo_axes);
   /* Detruire le contexte */
    DeleteDC(hMemdc);
   /* liberation du hdc */
    ReleaseDC(HWindow,hdc);
   }

  void TDialsec :: CalcDim(RTMessage)
   {
    HCURSOR hOldCur;
    char str[10];
```

```
  COLORREF rgbcolor,
           rouge=RGB(255,0,0);
  RECT rc;
  HDC hdc;
  hOldCur = SetCursor(LoadCursor(NULL,IDC_WAIT));
  long count=0,i,j,
       imin=1000,
       imax=0,
       jmin=1000,
       jmax=0,
       cote;
  GetClientRect(((TFenSec*)Fsec)->HWindow,&rc);
  hdc = GetDC(((TFenSec*)Fsec)->HWindow);
  /* Calcul de la dimension fractal */
  for (i=1;i<=(rc.right-10);i++)
   {
      for (j=1;j<=(rc.bottom-10);j++)
       {
        rgbcolor=GetPixel(hdc,i,j);
        if (rgbcolor == rouge)
         {
          count=count+1;
          if (i<imin) imin=i;
          if (i>imax) imax=i;
          if (j<jmin) jmin=j;
          if (j>jmax) jmax=j;
         };
       };
   };
  if((imin!=1000)&&(jmin!=1000))
   {
      if((imax-imin)>(jmax-jmin))
       {
        cote=imax-imin+1;
       }
      else
       {
        cote=jmax-jmin+1;
       }
      dimMapping=log(count)/log(cote);
   }
  SetCursor(hOldCur);
  sprintf(str,"%2.6f",dimMapping);
  Dime->SetText (str);
  ReleaseDC(((TFenSec*)Fsec)->HWindow,hdc);
 }

/* Fonctions a definir por la boite de dialogue qui
   va s'occuper de la realisation du spectre */
```

```
TDialtert :: TDialtert(TWindowsObject* Parent , LPSTR nom)
 : TDialog ( Parent, nom)
 {
   Ftert = new TFenTert(this,"",484,236);
 }

void  TDialtert :: RemplirData()
 {
  long i ;
  double moy;

  moy = 0;
  for (i=0; i<((TDialprinc*)Parent)->nbr; i++)
  {
    moy = moy + ((TDialprinc*)Parent)->tableau[i];
  }
  moy = moy / ((TDialprinc*)Parent)->nbr;

  num = 1 ;
  while (num<((TDialprinc*)Parent)->nbr)
      {
       num = num*2;
      }
  data = new double[num+1];
  i=1;
  while(i<=((TDialprinc*)Parent)->nbr)
   {
      data[i]=((TDialprinc*)Parent)->tableau[i-1]-moy;
      i++;
   }
  while(i<=num)
   {
      data[i]=0;
      i++;
   }
 }

void TDialtert :: TransFourCompl(double* data,long num, int signe)
 {
  double wr,wi,wpr,wpi,wtemp,theta,tempr,tempi;
  long n,m,j,mmax,istep,i;

  n=2*num;
  j=1;
  for (i=1;i<=n;i=i+2)
   {
      if (j>i)
       {
        tempr=data[j];
        tempi=data[j+1];
        data[j]=data[i];
```

```
            data[j+1]=data[i+1];
            data[i]=tempr;
            data[i+1]=tempi;
           };
          m=n/2;
          while ((m>=2)&&(j>m))
          {
           j=j-m;
           m=m/2;
          }
         j=j+m;
    };
    mmax=2;
    while(n>mmax)
    {
     istep=2*mmax;
        theta=6.28318530717959/(signe*mmax);
        wpr=-2*sin(0.5*theta)*sin(0.5*theta);
        wpi=sin(theta);
        wr=1;
        wi=0;
        for(m=1;m<=mmax;m=m+2)
        {
          for(i=m;i<=n;i=i+istep)
          {
           j=i+mmax;
           tempr=wr*data[j]-wi*data[j+1];
           tempi=wr*data[j+1]+wi*data[j];
           data[j]=data[i]-tempr;
           data[j+1]=data[i+1]-tempi;
           data[i]=data[i]+tempr;
           data[i+1]=data[i+1]+tempi;
          }
          wtemp=wr;
          wr=wr*wpr-wi*wpi+wr;
          wi=wi*wpr+wtemp*wpi+wi;
        }
        mmax=istep ;
    }
  }

void TDialtert :: TransFourReel(double* data,long num,int signe)
  {
   long i,n2p3,i1,i2,i3,i4;
   double wr,wi,wpr,wpi,wtemp,theta,c1,c2,wrs,wis,
          h1r,h1i,h2r,h2i;
   theta = 3.141592653589793/num;
   c1 = 0.5;
   if(signe==1)
   {
       c2 = -0.5;
```

```
        TransFourCompl(data,num,1);
    }
    else
    {
        c2 = 0.5;
        theta = -theta;
    }
    wpr=-2*sin(0.5*theta)*sin(0.5*theta);
    wpi=sin(theta);
    wr=1+wpr;
    wi=wpi;
    n2p3=2*num+3;
    for (i=2;i<=num/2;i++)
    {
        i1=2*i-1;
        i2=i1+1;
        i3=n2p3-i2;
        i4=i3+1;
        wrs=wr;
        wis=wi;
        h1r=c1*(data[i1]+data[i3]);
        h1i=c1*(data[i2]-data[i4]);
        h2r=-c2*(data[i2]+data[i4]);
        h2i=c2*(data[i1]-data[i3]);
        data[i1]=h1r+wrs*h2r-wis*h2i;
        data[i2]=h1i+wrs*h2i+wis*h2r;
        data[i3]=h1r-wrs*h2r+wis*h2i;
        data[i4]=-h1i+wrs*h2i+wis*h2r;
        wtemp=wr;
        wr=wr*wpr-wi*wpi+wr;
        wi=wi*wpr+wtemp*wpi+wi;
    }
    if(signe==1)
    {
        h1r=data[1];
        data[1]=h1r+data[2];
        data[2]=h1r-data[2];
    }
    else
    {
        h1r=data[1];
        data[1]=c1*(h1r+data[2]);
        data[2]=c1*(h1r-data[2]);
        TransFourCompl(data,num,-1);
    }
}


void TDialtert :: SetupWindow ()
{
  TDialog :: SetupWindow ();
```

```
    RemplirData();
    TransFourReel(data,num/2,1);
    ((TFenTert *)Ftert) -> ok = TRUE;
    SetWindowPos(Ftert->HWindow,0,23,21, 0, 0, SWP_NOSIZE | SWP_NOZORDER |
SWP_NOACTIVATE| SWP_NOREDRAW);
   }


 void TDialtert :: Ok(RTMessage Msg)
   {
    delete data ;
    TDialog :: Ok(Msg);
   }



 /* Fonctions a definir pour la fenetre de la boite de
    dialogue principale ou va s'afficher le rythme cardiaque */



 TFenPrinc ::  TFenPrinc (PTWindowsObject Parent, LPSTR Title, int alarg, int ahaut)
   :TFenFix (Parent, Title, alarg, ahaut)
   {
     ok = FALSE;
   }


 void TFenPrinc :: Init (double* tableau, long nbr)
  {
   aux = tableau ;
   num = nbr ;
  }

 void TFenPrinc :: Paint(HDC hdc,PAINTSTRUCT&)
  {
   int larg,haut;
   RECT rc;
   /* Definition de la dimension de fenetre */
   GetClientRect(HWindow,&rc);
   larg = rc.right-rc.left;
   haut = rc.bottom-rc.top;
   Rectangle (hdc,0,0,larg,haut);
   if (ok)
     Dessin(hdc,haut,larg);
   }

 void TFenPrinc :: Dessin(HDC hdc,int haut,int larg)
   {
    int j ;
```

```
 char str[4];
 RECT lpry,lprx;
 HPEN stylo_orig,stylo_axes, stylo_dessin;
/* Affichage des legendes*/
 SetTextAlign (hdc, TA_CENTER);
 TextOut(hdc,larg/2, 5, "Rythme cardiaque (Battements/min)",33);
 TextOut(hdc,larg/2, haut - 20, "Temps (sec)",11);

/* Creation du stylo pour les axes */
 stylo_axes = CreatePen(PS_SOLID,0,RGB(0,0,255));
 stylo_orig = SelectObject(hdc,stylo_axes);

 stylo_dessin=CreatePen(PS_SOLID,0,RGB(255,0,0));

/* Changement des coordonnees*/
 SetViewportOrg(hdc,30,haut-40);
 SetMapMode(hdc,MM_ANISOTROPIC);
 SetViewportExt(hdc,larg-60,haut-70);
 SetWindowExt(hdc,num-1,-90);
 SetWindowOrg(hdc,0,60);

/* Trace des axes de coordonnees  */
 MoveTo(hdc,0,60);
 LineTo(hdc,num-1,60);
 MoveTo(hdc,0,60);
 LineTo(hdc,0,150);
/* Trace des reperes sur les axes*/
j = num/10.0 ;
 while (j<num-1)
  {
      MoveTo(hdc,j,60);
      LineTo(hdc,j,61);
    sprintf(str,"%3d",j);
    TextOut(hdc,j,59,str,3);
      j+=num/10.0;
  };
 j = 70 ;
 while (j<=150)
  {
      MoveTo(hdc,0,j);
      LineTo(hdc,1,j);
    sprintf(str,"%3d",j);
      TextOut(hdc,- num/30,j,str,3);
      j+=10;
  };
/* Creation du stylo pour le dessin */

 SelectObject(hdc,stylo_dessin);
/* Trace du graphique  */
 MoveTo(hdc,0,aux[0]);
 for(j=1;j<num;j++)
```

```
    {
        LineTo(hdc,j,aux[j]);
    };
  /* Selectionner le stylo d'origine et detruire les autres */
   SelectObject(hdc,stylo_orig);
   DeleteObject(stylo_dessin);
   DeleteObject(stylo_axes);

  }

/* Fonction a definir por la fenetre de la boite de dialogue
   Dialsec ou l'on doit afficher le bitmap du mapping */

 TFenSec :: TFenSec(PTWindowsObject Parent, LPSTR Title, int alarg, int ahaut)
   :TFenFix (Parent, Title, alarg, ahaut)
  {
  }

 void TFenSec :: Paint(HDC hdc ,PAINTSTRUCT&)
  {
   HDC hMemdc;
   BITMAP bm;

   hMemdc=CreateCompatibleDC(hdc);
   SelectObject(hMemdc,((TDialsec*)Parent)->hMapping);
   GetObject(((TDialsec*)Parent)->hMapping,sizeof bm,(LPSTR)&bm);
   BitBlt(hdc,0,0,bm.bmWidth,bm.bmHeight,hMemdc,0,0,SRCCOPY);
   DeleteDC(hMemdc);
  }

/*  Fonctions a definir pour la fenetre ou l'on doit afficher
    le spectre du rythme cardiaque */


 TFenTert :: TFenTert(PTWindowsObject Parent, LPSTR Title, int alarg, int ahaut)
   :TFenFix (Parent, Title, alarg, ahaut)
  {
    ok = FALSE;
  }

 void TFenTert :: Paint(HDC hdc, PAINTSTRUCT&)
  {
   int larg,haut,i,j;
   char str[5];
   double aux,
     min=1000,
     max=-1000;
   RECT rc,lpry,lprx;
   HPEN stylo_orig,stylo_axes, stylo_dessin;

   if (ok)
```

```
{
 /* Determiner les bornes des donnees a afficher */
 if (abs (((TDialtert*)Parent)->data[2]) < min)
      min = abs(((TDialtert*)Parent)->data[2]);
 if (abs(((TDialtert*)Parent)->data[2]) > max)
      max = abs(((TDialtert*)Parent)->data[2]);

 for (i=3;i < ((TDialtert*)Parent)->num;i=i+2)
     { aux = sqrt((((TDialtert*)Parent)->data[i])*(((TDialtert*)Parent)->data[i])+
                 (((TDialtert*)Parent)->data[i+1])*(((TDialtert*)Parent)->data[i+1]));
       if (aux<min)
        min = aux;
       if (aux>max)
        max = aux;
  }
 /* Definition de la dimension de fenetre */
  GetClientRect(HWindow,&rc);
  larg = rc.right-rc.left;
  haut = rc.bottom-rc.top;
  SetTextAlign (hdc, TA_CENTER);
 /* Affichage des legendes*/
  TextOut (hdc,larg/2, 5, "Amplitude",9);
  TextOut (hdc,larg/2, haut-20,"Fréquence",9);
 /* Changement des coordonnees*/
    SetViewportOrg(hdc,30 ,haut-40);
  SetMapMode(hdc,MM_ANISOTROPIC);
  SetViewportExt(hdc,larg-60,haut-70);
  SetWindowExt(hdc,500,-(max-min));
  SetWindowOrg(hdc,0,min);
 /* Creation du stylo pour les axes */
  stylo_axes = CreatePen(PS_SOLID,0,RGB(0,0,255));
  stylo_orig = SelectObject(hdc,stylo_axes);
 /* Trace des axes de coordonnees  */
  MoveTo(hdc,0,min);
    LineTo(hdc,500,min);
  MoveTo(hdc,0,min);
  LineTo(hdc,0,max);
 /* Trace des reperes sur les axes */
 sprintf(str,"%1.2lf",0.0);
 TextOut(hdc,0,-(max-min)/20,str,3);
 j = 100 ;
  while (j<=500)
   {
       MoveTo(hdc,j,min);
       LineTo(hdc,j,min+((max-min)/50));
     sprintf(str,"%1.2lf",j/1000.0);
     TextOut(hdc,j,min-((max-min)/30),str,4);
       j+=100;
      };
 j = 1;
 aux = (max-min)/10.0;
```

```
    while (aux <= max-min)
     {
        MoveTo(hdc,0,min+aux);
        LineTo(hdc,1,min+aux);
      sprintf(str,"%1.2lf",j/10.0);
        TextOut(hdc,-15,min+aux,str,3);
        j+=1;
      aux = aux+(max-min)/10;
     };
    /* Creation du stylo pour le dessin */
     stylo_dessin=CreatePen(PS_SOLID,0,RGB(255,0,0));
     SelectObject(hdc,stylo_dessin);
    /* Trace du graphique  */

     aux = sqrt((((TDialtert*)Parent)->data[3])*(((TDialtert*)Parent)->data[3])+
                  (((TDialtert*)Parent)->data[4])*(((TDialtert*)Parent)->data[4]));
       MoveTo(hdc,1000.0*2/(((TDialtert*)Parent)->num),aux);
     i=3;
     for(j=5;j<(((TDialtert*)Parent)->num);j=j+2)
      {
        aux = sqrt((((TDialtert*)Parent)->data[j])*(((TDialtert*)Parent)->data[j])+
                  (((TDialtert*)Parent)->data[j+1])*(((TDialtert*)Parent)->data[j+1]));
        LineTo(hdc,1000.0*i/(((TDialtert*)Parent)->num),aux);
       i=i+1;
      };
     LineTo(hdc,1000.0*i/(((TDialtert*)Parent)->num),
           ((TDialtert*)Parent)->data[2]);

    /* Selectionner le stylo d'origine et detruire les autres */
     SelectObject(hdc,stylo_orig);
     DeleteObject(stylo_dessin);
     DeleteObject(stylo_axes);
   }
 }


/* Programme principal */

int PASCAL WinMain(HANDLE hinst,HANDLE hPrevInst,LPSTR lpCmd,int nCmd)
 {
  TPrapplic Premiere ("Prapplic",hinst,hPrevInst,lpCmd,nCmd);

  Premiere.Run();
  return Premiere.Status;

 }
```

```
# define CM_OUVRIR 1
# define CM_FERMER 2
# define CM_QUITTER 3
# define ID_BSPECTRE 101
# define ID_BDIM 102
# define ID_AFFDIM 103
# define ID_EDITDIM 104
# define ID_CALCDIM 105

#include "tfenfix.h"
#include <static.h>


class TPremiere:public TWindow
{
 public:

  HMENU hmp,hmo;
  BOOL init;
  TDialog  *Dprinc ;
  char NomFich [MAXPATH];
  FILE * fd;

  TPremiere(LPSTR titre);
  virtual void Quitter (RTMessage)=[CM_FIRST+CM_QUITTER];
  virtual void Ouvrir (RTMessage)=[CM_FIRST+CM_OUVRIR];
  virtual void Fermer (RTMessage)=[CM_FIRST+CM_FERMER];
  virtual void SetupWindow ();
  virtual void WMGetMinMaxInfo(RTMessage)=[WM_GETMINMAXINFO];
  virtual void GetWindowClass (WNDCLASS & WinClass);
};


class TPrapplic:public TApplication
{
 public:
  TPrapplic(LPSTR nom,HANDLE hi,HANDLE hj,LPSTR com,int nbr);
  virtual void InitMainWindow();
 };

class TDialprinc:public TDialog
 {
  public :
   TButton * Bspectre, * Bdim;
   TFenFix * Fprinc;
   double * tableau;
   long nbr;

   TDialprinc(TWindowsObject * Parent, LPSTR nom);
   ~TDialprinc() {};
   virtual void SetupWindow ();
```

```
  virtual void RemplirTableau(FILE* fd);
  virtual void TraitSpectre(RTMessage)=[ID_FIRST+ID_BSPECTRE];
  virtual void TraitDimension(RTMessage)=[ID_FIRST+ID_BDIM];
 };

class TDialsec : public TDialog
 {
  public :
   TStatic* Dime;
   TFenFix * Fsec;
   HBITMAP hMapping ;
   float dimMapping;
   TButton* CalDim;

   TDialsec(TWindowsObject * Parent, LPSTR nom);
   virtual void SetupWindow ();
   virtual void Ok(RTMessage);
   virtual void DesMap (double* tableau, long num);
   virtual void CalcDim (RTMessage)=[ID_FIRST+ID_CALCDIM];
 };

class TDialtert : public TDialog
 {
  public :
   TFenFix * Ftert;
   double* data;
   long num;

   TDialtert(TWindowsObject * Parent, LPSTR nom);
   virtual void SetupWindow ();
   virtual void Ok(RTMessage);
   virtual void RemplirData();
   virtual void TransFourReel(double* data,long num,int signe);
   virtual void TransFourCompl(double* data,long num,int signe);
 };

class TFenPrinc: public TFenFix
 {
   public:
     BOOL ok;
     double * aux;
     long num;
     TFenPrinc (PTWindowsObject Parent, LPSTR Title, int alarg, int ahaut);
     virtual void Paint (HDC PaintDC, PAINTSTRUCT&);
     virtual void Init(double* tableau,long nbr);
     virtual void Dessin(HDC hdc,int haut,int larg);
 };


class TFenSec : public TFenFix
 {
```

```
 public:
   TFenSec (PTWindowsObject Parent,LPSTR Tiltle, int alarg, int ahaut);
   virtual void Paint ( HDC hdc, PAINTSTRUCT&);
} ;

class TFenTert : public TFenFix
 {
 public:
   double* data;
   long num ;
   BOOL ok;

   TFenTert (PTWindowsObject Parent,LPSTR Tiltle, int alarg, int ahaut);
   virtual void Paint(HDC hdc,PAINTSTRUCT&);
 };
```

```
# include "tfenfix.h"

/*************************************************************************/
/*                Définition de la classe TFenFix                        */
/*************************************************************************/

/******************** Constructeur de la classe ************************/
TFenFix::TFenFix (PTWindowsObject Parent, LPSTR Title, int alarg, int ahaut): TWindow
(Parent, Title)
{
  Attr.Style |= WS_CHILD | WS_VISIBLE;

  init = FALSE;
  larg = alarg;
  haut = ahaut;

  if (GetSystemMetrics (SM_CXSCREEN) > 800)
    fact = FACT;
  else
    fact = 1;
}


/*************** Redéfinition de la méthode GetClassName ***************/
LPSTR TFenFix::GetClassName ()
{
  return "FenFix";
}


/*************** Redéfinition de la méthode GetWindowClass **************/
void TFenFix::GetWindowClass (WNDCLASS& ClasseWin)
{
  TWindow::GetWindowClass (ClasseWin);
  ClasseWin.style = CS_BYTEALIGNWINDOW;
}


/*************** Redéfinition de la méthode SetupWindow ****************/
void TFenFix::SetupWindow ()
{
  TWindow::SetupWindow ();
  SetWindowPos (HWindow, 0, 0, 0, larg, haut, SWP_NOMOVE | SWP_NOZORDER |
SWP_NOACTIVATE);
}
```

```
# define FACT       1.255
# include <owl.h>

/*************************************************************************/
/*                Déclaration de la classe TFenFix                       */
/*************************************************************************/

class TFenFix: public TWindow
{
  int  larg, haut;

  protected:
    BOOL  init;
    float fact;

  public:
    TFenFix (PTWindowsObject Parent, LPSTR Title, int alarg, int ahaut);
    virtual LPSTR GetClassName ();
    virtual void GetWindowClass (WNDCLASS&);
    virtual void SetupWindow ();
};
```