Alain Prignet

Quelques applications de l'analyse

23 octobre 2012

Université Paris-Est Marne-la-Vallée
France

Table des matières

Présentation

Un cours d'analyse partant d'applications réelles (mais traitées de façon académiques). Les cours et TD/TP sont totalement intégrés et utilisent en permance `octave` ou `scilab`.

Prévu pour 60h au total en L3 IMI (ingéniérie mathémathique et informatique), deux gros chapitres : Filtrage par analyse de Fourier, Simplexe. Et un petit chapitre rapide sur les EDO. Les autres chapitres sont des essais partiellement abandonnés.

L'évaluation se fait par deux examens en controle continu sur chacun des deux chapitres important et la rédaction d'un rapport qui doit insister sur les observations numériques faites par chacun.

Ce sont des notes de cours parfois elliptiques et à considérer avec un très fort esprit critique.

Chapitre 1

Filtrage et analyse de Fourier

1.1 Problèmes

1.1.1 Débruitage

On a un signal (son, courbe, résultat de mesure, ...) à une dimension ou une image à deux dimension présentant un bruit aléatoire. On souhaite réduire ce bruit tout en préservant le mieux possible l'objet d'origine. On suppose que le bruit varie independannent sur chaque pixel ou sur chaque échantillon : il s'agit donc d'un détail très fin contrairement au reste du signal ou de l'image, et donc de haute fréquence. On propose donc d'utiliser l'analyse de Fourier (série et transformation). Evidemment les détails très fin de l'ordre de la taille du bruit seront perdus.

1.1.2 DAC

On souhaite réaliser la conversion d'un signal numérique (prenant un nombre fini de valeurs) en un signal analogique (à valeur réelle). On va d'abord convertir le signal numérique en un signal toujours numérique mais ne prenant que deux valeurs et "haché" puis lui appliquer un filtre passe bas pour supprimer le "hachage" (détail fin fréquence élevée) et ne conserver que la moyen (fréquence basse).

1.1.3 FM

On souhaite démoduler de la FM. Il y a deux étapes, la démodulation stricte de la modulation de fréquence, puis la réconstruction des canaux gauche et droite. Pour cela on est amené à utiliser des filtres passe bas et passe bande que l'on construiera à partir de la FFT.

1.2 Définition

Définition 1 *Soit f une application de $\mathbf{R}$ dans $\mathbf{R}$. Si $f(x+T) = f(x)$ pour tout x on dira qu'elle admet $T > 0$ réel pour période. Si une fonction admet une période, elle est périodique.*

(Ici T est UNE période et pas nécessairement LA période)

Définition 2 *On appelle LA période T d'une application f de $\mathbf{R}$ dans $\mathbf{R}$ le plus petit réel strictement positif période de f.*

Exemple : $\cos(4\pi x)$ a pour période $1/2$, et $\cos(4\pi x)\sin(2\pi y)$ $(1/2, 1)$

Définition 3 *Sur $\mathbf{R}$ on appelle fréquence, l'inverse de la période : $1/T$.*

1.3 Série de Fourier

1.3.1 Coefficients et spectre

Cas continu

L'idée consiste à décomposer une fonction f périodique de période 1, à l'aide des fonctions $\cos(2\pi nx)$ et $\sin(2\pi mx)$ (elles même "périodiques" (au sens large) de période 1) pour n et m dans $\mathbf{Z}$.

Cette famille est agréable car elle est orthogonale pour le produit scalaire naturel ($(f,g) = \int_0^1 f(x)g(x)dx$ pour des fonctions intégrables ou continues par morceaux sur $[0,1]$) pour n et m dans $\mathbf{N}$. C'est-à-dire que

Rappel : $e^{ia} = \cos\alpha + i\sin\alpha$ donc en particulier $e^{i0} = 1$, $e^{i\pi} = -1$, $e^{i\pi/2} = i$ etc $\cos(a) = (e^{ia} + e^{-ia})/2$ et $\sin(a) = (e^{ia} - e^{-ia})/(2i)$.

Propriété 4 *Pour n et m dans $\mathbf{Z}$ avec $n \neq m$ et $n \neq -m$, on a*

$$\int_0^1 \cos(2\pi nx)\cos(2\pi mx)dx = \int_0^1 \sin(2\pi nx)\sin(2\pi mx)dx = \int_0^1 \cos(2\pi nx)\sin(2\pi mx)dx = 0.$$

et pour $n \neq 0$

$$\int_0^1 \cos^2(2\pi nx) = \int_0^1 \sin^2(2\pi nx) = 1/2, \quad et \quad \int_0^1 \cos^2(2\pi 0x) = 1, \quad \int_0^1 \sin^2(2\pi 0x) = 0.$$

Démonstration : On a $\cos(a)\cos(b) = (e^{ia} + e^{-ia})(e^{ib} + e^{-ib})/4 = (e^{i(a+b)} + e^{-i(a+b)} + e^{i(a-b)} + e^{-i(a-b)})/4$ soit

$$\cos(a)\cos(b) = (\cos(a+b) + \cos(a-b))/2.$$

Et de même, $\sin(a)\sin(b) = (e^{i(a+b)} + e^{-i(a+b)} - e^{i(a-b)} + e^{-i(a-b)})/(-4)$ soit

$$\sin(a)\sin(b) = (\cos(a-b) - \cos(a+b))/2.$$

Et enfin $\cos(a)\sin(b) = (e^{i(a+b)} - e^{-i(a+b)} - e^{i(a-b)} + e^{-i(a-b)})/(4i)$ soit

$$\cos(a)\sin(b) = (\sin(a+b) - \sin(a-b))/2.$$

Ainsi $\int_0^1 \cos(2\pi nx)\cos(2\pi mx)dx = (1/2)\int_0^1(\cos(2\pi(n+m)x) + \cos(2\pi(n-m)x)dx = (1/2)[\sin(2\pi(n+m)x)/(n+m)) + \sin(2\pi(n-m)x)/(n-m)]_0^1 = 0$ par périodicité. Et de même pour les autres.

Bien qu'il y ait beaucoup de définitions différentes, on choisit la suivante :

Définition 5 *On appelle coefficients de Fourier de l'application f intégrable et périodique de période 1, pour $k \in \mathbf{Z}$,*

$$c_k = a_k + ib_k = \int_0^1 f(x)e^{-i2\pi kx}dx.$$

Soit

$$a_k = \int_0^1 f(x)\cos(2\pi kx)dx \quad et \quad b_k = -\int_0^1 f(x)\sin(2\pi kx)dx.$$

Définition 6 *On appelle spectre d'une application f périodique et intégrable, les valeurs de $k \in \mathbf{Z}$ telles que $c_k \neq 0$.*

Exemple : on note $f(x) = \cos(4\pi x)$ et $g(x) = 1$ sur $]0,1/2[$, $g(x) = -1$ sur $]1/2,1[$, $g(0) = g(1/2) = g(1) = 0$ et périodique de période 1.

Pour f on voit que $c_k = 0$ si $k \neq 2$ et $k \neq -2$. Et $c_2 = c_{-2} = 1/2$. Son spectre est $\{-2,2\}$.

Pour g, on a

$$a_k = \int_0^{1/2} \cos(2\pi kx)dx - \int_{1/2}^1 \cos(2\pi kx)dx \quad \text{et} \quad b_k = -\int_0^{1/2} \sin(2\pi kx)dx + \int_{1/2}^1 \sin(2\pi kx)dx.$$

On voit sans calculs que $a_n = 0$ pour tout n par imparité de g. De plus $b_k = 2\int_{1/2}^{1} \sin(2\pi kx)dx$. Si $k = 0$, $b_0 = 0$, si $k \neq 0$, $b_k = 2/(2\pi k)[-\cos(2\pi kx)]_{1/2}^{1} = 2/(2\pi k)[-1 + (-1)^k]$, soit $b_k = 0$ pour k pair et $b_k = -2/(\pi k)$ pour k impair. Le spectre de g est donc constitués des entiers relatifs impairs.

Propriété 7 *Si f est réelle, on a $c_{-k} = \overline{c_k}$ ou encore $a_{-k} = a_k$ et $b_{-k} = -b_k$.*

Cas discret

On suppose ici qu'on ne connaît f qu'en certains points. Les calculs des intégrales peuvent être fait de façon approchés, par exemple avec la méthode des rectangles : si l'on connaît $f(a)$ on approche $\int_a^b f(x)dx$ par la surface du rectangle $(b-a)f(a)$ (dessin). En supposant équidistants les points où f est connu, on a donc (avec la méthode des rectangles à gauche)

$$\widetilde{a}_k = (1/N)\sum_{n=0}^{N-1} f(n/N)\cos(2\pi kn/N) \quad \widetilde{b}_k = -(1/N)\sum_{n=0}^{N-1} f(n/N)\sin(2\pi kn/N)$$

$$\widetilde{c}_k = (1/N)\sum_{n=0}^{N-1} f(n/N)e^{-2\pi kn/N}$$

Pour $N = 4$ et la fonction g : $g(0) = 1$, $g(1/4) = 1$, $g(1/2) = 0$, $g(3/4) = -1$, on obtient : $\widetilde{a}_0 = (g(0) + g(1/4) + g(1/2) + g(3/4))/4 = 1/4$, $b_0 = 0$, $\widetilde{a}_1 = (g(0) - g(1/2))/4 = 1/4$, $b_1 = -(g(1/4) - g(3/4))/4 = -1/2$, $\widetilde{a}_2 = (g(0) - g(1/4) + g(1/2) - g(3/4))/4 = 1/4$, $b_2 = 0$, $\widetilde{a}_3 = (g(0) - g(1/2))/4 = 1/4$, $b_3 = -(-g(1/4) + g(3/4))/4 = 1/2$ On a donc $\widetilde{a}_k = 1/4$ constant c'est l'erreur d'intégration, devrait avoir $\widetilde{a}_k = 0$ et on a les $\widetilde{b}_k$ nuls pour k pair comme dans le cas continue et proche de $\mp 2/\pi$ pour $k = \pm 1$. Ceci est cohérent.

Pour f, et toujours $N = 4$, $f(0) = 1$, $f(1/4) = -1$, $f(1/2) = 1$, $f(3/4) = -1$, d'où : $\widetilde{a}_0 = 0$, $\widetilde{b}_0 = 0$, $\widetilde{a}_1 = 0$, $\widetilde{b}_1 = 0$, $\widetilde{a}_2 = 1$, $\widetilde{b}_2 = 0$, $\widetilde{a}_3 = 0$, $\widetilde{b}_3 = 0$. Cohérent avec les valeurs continues.

Ce qui conduit au code octave suivant :

```
clear

N=20
M=40
x=(0:(N-1))/N;

y1=cos(4*pi*x);

y2=-sign(x-1/2);


y=y1;

k=-M:M;
for j=1:length(k)
  an(j)=sum((y.*cos(2*pi*k(j)*x)))/N;
  bn(j)=-sum((y.*sin(2*pi*k(j)*x)))/N;
end
plot(k,an,k,bn)
```

Onservations :

1) convergence pour $N \to \infty$ (en $1/N$ pour g) des coefficients.

2) Si l'on observe les resultats pour la fonction $f(x) = \cos(4\pi x)$, on constate que le spectre discret contient non seulement -2 et 2 mais aussi (pour $N = 20$ et $M = 40$), -38, -22, -18, 18, 22, 38. Il s'agit du phénomène de repliement de spectre : le spectre "normal" est reproduit par une translation de longueur $20 = N$: 2 se retrouve en -38, -18, 22 et -2 en -22, 18 et 38. Même phénomene pour $g(x) = -sign(x - 1/2)$ sur $[0, 1]$, la valeur $-0,63$ se retrouve en -39, -19, 1, 21.

On peut donc utiliser les valeurs sur une période de longueur $20 = N$: soit k entre 0 et 19 soit entre -10 et 9 ou -9 et 10 (c'est-à-dire approxivement $[0, N[$ ou $[-N/2, N/2[$).

On a observé que les coefficients sont périodiques de période N.

Ceci ne contredit pas la convergence des $\widetilde{c}_k$ approchés vers les c_k exacts : pour k fixé les artefacts disparaissent pour $N/2 > k$.

Artefacts et théorème de Shannon

Il y a donc l'apparition d'"artefacts" numériques qui n'ont aucune interprétation continue. comparons $\widetilde{c}_k$ et $\widetilde{c}_{k+N}$, et pour cela observons a_{k+N}. Il faut noter que $x_n = (n-1)/N$ (méthode des rectangles à gauche) donc

$$\widetilde{a}_{k+N} = (1/N)\sum_{n=1}^{N} f(x_n)\cos(2\pi(k+N)\frac{n-1}{N}) = (1/N)\sum_{n=1}^{N} f(x_n)\cos(2\pi k\frac{n-1}{N} + 2\pi(n-1))$$
$$= (1/N)\sum_{n=1}^{N} f(x_n)\cos(2\pi k(n-1)/N) = \widetilde{a}_k$$

et de même $\widetilde{b}_{k+N} = \widetilde{b}_k$. Donc $\widetilde{c}_{k+N} = \widetilde{c}_k$.

Si comme dans l'exemple de la fonction f le spectre continu est inclus dans l'intervalle $[-N/2, N/2]$, on peut limiter sans perte le spectre discret à ce même intervalle. Si, en revanche, le spectre continu dépasse de l'intervalle $[-N/2, N/2]$ (comme dans le cas de la fonction g), il y a une confusion entre le spectre original et ses artefacts. On observe ici une illustration du théorème de Shannon :

Définition 8 *On appelle fréquence d'échantillonage $F_e = N$ pour un maillage de pas $1/N$ sur $[0, 1]$.*

Théoreme 9 *(Shannon 1949) If a function $f(t)$ contains no frequencies higher than W cps (hertz), it is completely determined by giving its ordinates at a series of points spaced $1/(2W)$ seconds apart.*

Théoreme 10 *(traduction Shannon 1949) Si une fonction $x(t)$ ne contient pas de fréquences supérieur à W hertz, elle est completement déterminée par la donnée de ses valeurs en une serie de points espacés de $1/(2W)$ secondes.*

Théoreme 11 *(adapté de Shannon) On ne peut considérer de façon discrète que des fonctions dont le spectre est contenu dans $[-F_e/2, F_e/2]$ où F_e est la fréquence d'echantillonage.*

Exemple : le CD audio est echantilloné à 44100Hz. Il permet donc de reproduire des fréquences jusqu'à 22050Hz (ie au delà des capacité de l'oreille humaine).

Démo du théorème d'origine : on suppose pour simplifier que $F_e = 1$, on part d'une fonction f et on note $\hat{f}$ sa transformée de Fourier, puis on prolonge par périodicité de période 1, $\hat{f}(\xi)$ en une fonction $\overline{f}$ définie sur $\mathbf{R}$ et les deux fonctions coïncident sur $]-1/2, 1/2[$. Les coefficients de Fourier pour $\overline{f}$

$$c_{-n} = \int_a^{a+1} \overline{f}(\xi)e^{2\pi n\xi}d\xi$$

on peut choisir $a = -1/2$ donc

$$c_{-n} = \int_{-1/2}^{1/2} \overline{f}(\xi)e^{2\pi n\xi}d\xi = \int_{-1/2}^{1/2} \hat{f}(\xi)e^{2\pi n\xi}d\xi = \int_{-\infty}^{\infty} \hat{f}(\xi)e^{2\pi n\xi}d\xi$$

il s'agit de la transformée de Fourier inverse de $\hat{f}$ donc de f : $c_{-n} = f(n)$ mais $\overline{f}$ coïncide (sous hypothèse de régularité) avec sa série de Fourier donc

$$\overline{f}(\xi) = \sum_{n\in\mathbf{Z}} c_n e^{i2\pi n\xi} = \sum_{n\in\mathbf{Z}} f(-n)e^{i2\pi n\xi} = \sum_{n\in\mathbf{Z}} f(n)e^{-i2\pi n\xi}$$

qui coïncide avec $\hat{f}$ sur $]-1,1[$. Ainsi $\hat{f}$ (et donc f par transformé de Fourier inverse) sont entièrement déterminés par la connaissance des $f(n)$, $n \in Z$. On a de plus

$$f(x) = \int_{\mathbf{R}} \hat{f}(\xi) e^{i2\pi x\xi} d\xi = \int_{-1/2}^{1/2} \hat{f}(\xi) e^{i2\pi x\xi} d\xi = \int_{-1/2}^{1/2} \sum_{n\in\mathbf{Z}} f(n) e^{-i2\pi n\xi} e^{i2\pi x\xi} d\xi$$
$$= \int_{-1/2}^{1/2} \sum_{n\in\mathbf{Z}} f(n) e^{i2\pi(x-n)\xi} d\xi = \sum_{n\in\mathbf{Z}} f(n) \int_{-1/2}^{1/2} e^{i2\pi(x-n)\xi} d\xi = \sum_{n\in\mathbf{Z}} f(n) \sin(\pi(x-n))/(\pi(x-n))$$

1.3.2 Série

Cas continu

On a les deux théorème suivant qui permettent de reconstituer une fonction périodique à partir de ses coefficients de Fourier :

Théoreme 12 *Si f est C^1 par morceaux (au sens un nombre fini de morceaux de la forme $]a,b[$ et f coïncide sur l'ouvert avec une fonction C^1 sur le fermé), et si aux points de discontinuité $f(x) = (f(x^+) + f(x^-))/2$ alors on a la convergence simple de la série de Fourier vers f :*

$$f(x) = \sum_{k\in\mathbf{Z}} c_k e^{i2\pi kx} = \sum_{k\in Z} a_k \cos(2\pi kx) - b_k \sin(2\pi kx) + i \sum_{k\in Z} a_k \sin(2\pi kx) + b_k \cos(2\pi kx)$$

et

Théoreme 13 *Si f est continue en tous points et C^1 par morceaux alors la convergence de la serie de Fourier est uniforme.*

On peut retrouver la formule utilisant les a_n et les b_n à partir de la formule utilisant les c_n.

Exemple : En appliquant les théorème, on peut dire que les séries convergent au moins simplement pour g et uniformément pour f. On ne peut pas, a priori, affirmer à l'aide de ces théorème que le série ne converge pas uniformément pour g. Cependant les sommes partielles de la séries sont continues, donc si la convergence était uniforme alors g serait continue. La convergence n'est donc pas uniforme. Ceci utilise :

Théoreme 14 *La limite uniforme de fonctions continues est continue*

Démo : ....

En fait on peut faire un peu mieux dans le premier théorème : remplacer C^1 par morceaux par continue par morceaux et dérivable par morceaux (avec des dérivées à gauche et à droite en tout points des morceaux)

Contre exemple : $x^2 \sin(1/x^2)$ sur $[0,1]$. La fonction est bien continue sur $[0,1]$ en prolongeant par 0, et sa dérivé (sur $]0,1]$) $2x\sin(1/x^2) - 2\cos(1/x^2)/x$ ne peut clairement pas de prolonger en 0, et pourtant il y a une dérivée en 0^+ : $[(x^2\sin(1/x^2)) - 0]/(x-0) = x\sin(1/x^2)$ donc la dérivé en 0^+ est 0.

Exemple : pour la fonction f le spectre est fini, la série est donc juste une somme et on a $(1/2)\cos(2\pi(-2)x) + (1/2)\cos(2\pi(2)x) = \cos(4\pi x)$. Pour la fonction g, si l'on choisit $x = 1/4$, on obtient :

$$1 = -\sum_{m\in\mathbf{Z}} -2/(\pi(2m+1))\sin(2\pi(2m+1)/4)$$

or

$$\sin(2\pi(2m+1)/4) = \sin(\pi m + \pi/2) = (-1)^m$$

donc on retrouve la formule

$$\pi/2 = \sum_{m\in\mathbf{Z}} (-1)^m/(2m+1).$$

Cas discret

On ne peut calculer la somme infinie. On se limite donc entre $-M$ et M :

```
clear

N=100
M=50
x=(0:(N-1))/N;

y1=cos(4*pi*x);

y2=-sign(x-1/2);

%y=y1;
y=y2;

k=-M:M;
for j=1:2*M+1
  ak(j)=sum((y.*cos(2*pi*k(j)*x)))/N;
  bk(j)=-sum((y.*sin(2*pi*k(j)*x)))/N;
end
%plot(k,an,k,bn)

z=0*x;
for j=1:2*M+1
  z=z+ak(j)*cos(2*pi*k(j)*x)-bk(j)*sin(2*pi*k(j)*x);
end
plot(x,y,x,z)
```

On constate comme prévu par le théorème de Shannon que la fonction et sa série (partielle) n'est proche de la fonction d'origine que pour $M \leq N/2$ (si le spectre de la fonction est infini).

Si on essaie successivement $M = 1$, puis 2, 3 etc, on voit la convergence des sommes partielles vers la fonction.

Si $M > N/2$, il n'y a plus convergence de la série vers la fonction : par exemple si $M = N$

$$\sum_{k=-N}^{N} c_k e^{i2\pi kj/N} = \sum_{k=-N}^{-1} c_k e^{i2\pi kj/N} + \sum_{k=0}^{N} c_k e^{i2\pi kj/N}$$

et par périodicité des c_k, en posant $l = k + N$

$$\sum_{k=-N}^{-1} c_k e^{i2\pi kj/N} = \sum_{l=0}^{N-1} c_{l-N} e^{i2\pi (l-N)j/N} = \sum_{l=0}^{N-1} c_{l-N} e^{i2\pi lj/N} e^{-i2\pi j} = \sum_{l=0}^{N-1} c_l e^{i2\pi lj/N}$$

donc

$$\sum_{k=-N}^{N} c_k e^{i2\pi kj/N} = 2 \sum_{k=0}^{N-1} c_k e^{i2\pi kj/N} + c_0$$

qui est proche de $c_0 + 2f(x_j)$ et non $f(x_j)$.

Les seules composantes utilisables sont une période : $k = 0$ à $N - 1$ que l'on peut "symetriser", on suppose d'abord N pair

$$\sum_{k=0}^{N-1} c_k e^{i2\pi kj/N} = \sum_{k=0}^{N/2} c_k e^{i2\pi kj/N} + \sum_{k=N/2+1}^{N-1} c_k e^{i2\pi kj/N} = \sum_{k=0}^{N/2} c_k e^{i2\pi kj/N} + \sum_{k=-N/2+1}^{-1} c_k e^{i2\pi kj/N}$$

donc

$$\sum_{k=0}^{N-1} c_k e^{i2\pi kj/N} = \sum_{k=-N/2+1}^{N/2} c_k e^{i2\pi kj/N}$$

et si N est impair

$$\sum_{k=0}^{N-1} c_k e^{i2\pi kj/N} = \sum_{k=0}^{(N-1)/2} c_k e^{i2\pi kj/N} + \sum_{k=(N-1)/2+1}^{N-1} c_k e^{i2\pi kj/N} = \sum_{k=0}^{(N-1)/2} c_k e^{i2\pi kj/N} + \sum_{k=-(N-1)/2}^{-1} c_k e^{i2\pi kj/N}$$

donc

$$\sum_{k=0}^{N-1} c_k e^{i2\pi kj/N} = \sum_{k=-(N-1)/2}^{(N-1)/2} c_k e^{i2\pi kj/N}$$

On pourra reprendre le cas N impair lors du filtrage par la suite car la somme est symétrique.

Si on reprend les coefficients de Fourier approchés pour f et g et que l'on effectue les calculs en les x_n : pour f, on a vu que seul $c_2 = 1$ est non nul. La série vaut $s(0) = 1$, $s(1/4) = -1$, $s(1/2) = 1$, $s(3/4) = -1$. Et coïncide donc avec f.

Pour g, $Res(0) = a_0 + a_1 + a_2 + a_3 = 1/4 + 1/4 + 1/4 + 1/4 = 1$, $Res(1/4) = (a_0 - a_2) - (b_1 - b_3) = (1/4 - 1/4) - (-1/2 - 1/2) = 1$, $Res(1/2) = (a_0 - a_1 + a_2 - a_3) = 0$ et $Res(3/4) = (a_0 - a_2) - (-b_1 + b_3) = -1$.

On remarque que, dans ce cas particulier, l'on retrouve les valeurs d'origine. Explication ultérieure.

Phénomème de Gibbs

Retour au cas continu mais affiché discretement :

```
clear

N=1000
M=90
x=0:1/N:1;

for j=1:N+1
  if (x(j)<1/2)
    y2(j)=1;
  else
    if (x(j)>1/2)
      y2(j)=-1;
    else
      y2(j)=0;
    end
  end
end


y=y2;

k=-M:M;
for j=1:2*M+1
  if (mod(k(j),2)~=0)
    bn(j)=-2/(pi*(k(j)));
  else
    bn(j)=0;
  end
end

z=0*x;
for j=1:2*M+1
  z=z-bn(j)*sin(2*pi*k(j)*x);
```

```
end
plot(x,y,x,z)
```

On observe qu'il faut ici M petit devant N pour voir les détails. Et en particulier le maximum de la somme partielle est de l'ordre de $1,18$ soit environ 18% d'erreur : c'est le phenomèm de Gibbs.

On peut retrouver cela : Soit la somme partielle (attention renommage d'indice) :

$$S_N(x) = 2/\pi \sum_{n=-N}^{N} \sin(2\pi(2n+1)x)/(2n+1).$$

Comme la somme est finie, on peut aisement dériver : $S'_N(x) = 4\sum_{n=-N}^{N} \cos(2\pi(2n+1)x)$ donc $S'_N(x) = Re(4\sum_{n=-N}^{N} e^{i2\pi(2n+1)x})$. Or

$$\begin{aligned}
\sum_{n=-N}^{N} e^{i2\pi(2n+1)x} &= e^{i2\pi(-2N+1)x} \sum_{n=-N}^{N} e^{i2\pi 2(n+N)x} \\
&= e^{i2\pi(-2N+1)x} \frac{1-e^{i2\pi 2(2N+1)x}}{1-e^{i2\pi 2x}} \\
&= e^{i2\pi(-2N+1)x} \frac{e^{i2\pi(2N+1)x}}{e^{i2\pi x}} \frac{e^{-i2\pi(2N+1)x}-e^{i2\pi(2N+1)x}}{e^{-i2\pi x}-e^{i2\pi x}} \\
&= e^{i2\pi(-2N+1)x} \frac{e^{i2\pi(2N+1)x}}{e^{i2\pi x}} \frac{e^{-i2\pi(2N+1)x}-e^{i2\pi(2N+1)x}}{e^{-i2\pi x}-e^{i2\pi x}} \\
&= e^{i2\pi x} \sin(2\pi(2N+1)x)/\sin(2\pi x)
\end{aligned}$$

donc

$$S'_N(x) = 4\cos(2\pi x)\sin(2\pi(2N+1)x)/\sin(2\pi x).$$

Les extrema sont donc atteints (entre autre) pour $x = 0$, $x = 1/4$ mais surtout pour $2\pi(2N+1)x = \pi$ soit $x_N = 1/(4N+2)$. On souhaite maintenant évaluer $S_N(x_N)$. Comme les calculs sur S_N sont difficiles et ceux de S'_N plus faciles, on écrit :

$$S_N(x_N) = \int_0^{x_N} S'_N(x)dx = 4\int_0^{1/(4N+2)} \cos(2\pi x)\frac{\sin(2\pi(2N+1)x)}{\sin(2\pi x)}dx.$$

Comme les bornes tendent vers 0, on effectue le changement de variable $t = \pi(4N+2)x$, d'où :

$$S_N(x_N) = 4\int_0^{\pi} \cos(t/(2N+1))\frac{\sin(t)}{\sin(t/(2N+1))}\frac{dt}{2\pi(2N+1)}.$$

Or lorsque N tend vers $+\infty$, $t/(2N+1)$ tend vers 0 donc $\cos(t/(2N+1)) \approx 1$ et $\sin(t/(2N+1)) \approx t/(2N+1)$ si bien que

$$S_N(x_N) \approx 2/\pi \int_0^{\pi} \frac{\sin(t)}{t}dt \approx 1,178$$

C'est-à-dire une erreur de 18% entre la fonction et la somme partielle (quand N tend vers ∞).

On notera que ce phénomène disparait dans le cas discret. Il est maximum pour $M = N/4$ puis regresse pour disparaitre pour $M = N/2$. Un avantage des calculs discrets.

1.3.3 Filtrage

k entre $-M$ et M.
Pour $\cos(4\pi x) + \sin(8\pi x)$,

- $M \leq 1$ rien
- $2 \leq M \leq 3$ cos
- $4 \leq M \leq N/2$ cos + sin
- $N/2 < M$ problème

1.4 Transformée de Fourier

Permet de traiter les fonctions non periodiques grâce au fait que le spectre est maintenant continu.

En fait cela n'est intéressant que dans le cas continu : dans le cas discret et notamment la TFD on se limite en fait au spectre en des valeurs entières et il s'agit donc plutôt d'une approximation de la série de Fourier.

1.4.1 Transformée directe

Cas continu

Définition 15 *Si f une fonction mesurable sur $\mathbf{R}$ et si $\int_{\mathbf{R}} |f(x)|dx < +\infty$ alors on peut définir $\hat{f}(\xi) = \int_{\mathbf{R}} f(x)e^{-i2\pi x\xi}dx$ que l'on appelle transformée de Fourier de f, notée $\hat{f}$.*

Démo : $|f(x)e^{-i2\pi x\xi}| = |f(x)|$ et $\int_{\mathbf{R}} |f(x)|dx < +\infty$ donc $\int_{\mathbf{R}} f(x)e^{-i2\pi x\xi}dx$ est bien défini pour tout ξ.

Remarque : $Re(\hat{f})(\xi)$ est pair, et $Im(\hat{f})(\xi)$ impair.

Remarque $cos(2\pi x)$ sur $\mathbf{R}$ ne satisfait pas les hypothèses mais $cos(2\pi x)$ sur $[0,1]$ et 0 ailleurs oui.

Remarque : $\widehat{f/[0,1]}(k) = c_k$. Généralisation des coefficients de Fourier.

Exemple : $f(x) = 1$ sur $[-1/2, 1/2]$ et 0 ailleurs : pour $\xi \neq 0$

$$\hat{f}(\xi) = \int_{-1/2}^{1/2} e^{-i2\pi\xi x}dx = \frac{-1}{i2\pi\xi}[e^{-i2\pi\xi x}]_{-1/2}^{1/2} = \frac{1}{i2\pi\xi}(e^{i\pi\xi} - e^{-i\pi\xi}) = \sin(\pi\xi)/(\pi\xi) = sinc(\pi\xi)$$

Approximation discrète

On utilise à nouveau la méthode des rectangles à gauche.

$$\hat{f}(\xi) = \int_{\mathbf{R}} f(x)e^{-i2\pi x\xi}dx \approx \frac{1}{N}\sum_{n\in\mathbf{Z}} f(\frac{n}{N})e^{-i2\pi\xi n/N}.$$

Rq Comme approximation discrète des coefficients de Fourier mais avec $\xi \in \mathbf{R}$ au lieu de $k \in \mathbf{Z}$.

La somme est en fait finie car on suppose que f est nulle en dehors de $[0,1]$ (ici on ne suppose plus f périodique) :

$$\hat{f}(\xi) = \int_0^1 f(x)e^{-i2\pi x\xi}dx \approx \frac{1}{N}\sum_{n=0}^{N-1} f(\frac{n}{N})e^{-i2\pi\xi n/N}.$$

$$\hat{f}(\xi+N) \approx \frac{1}{N}\sum_{n=0}^{N-1} f(\frac{n}{N})e^{-i2\pi(\xi+N)n/N} = \frac{1}{N}\sum_{n=0}^{N-1} f(\frac{n}{N})[e^{-i2\pi\xi n/N}e^{-i2\pi n}]$$
$$= \frac{1}{N}\sum_{n=0}^{N-1} f(\frac{n}{N})e^{-i2\pi\xi n/N} \approx \hat{f}(\xi).$$

L'approximation discrète de la transformée de Fourier est donc périodique de période N (on retrouve ici le repliement de spectre).

```
clear

N=100
M=50
P=10
x=(0:(N-1))/N;

y1=cos(4*pi*x);
```

```
y2=-sign(x-1/2);

%y=y1;
y=y2;

xi=(-M*P:M*P)/P:
for j=1:length(xi)
  rfc(j)=sum((y.*cos(2*pi*xi(j)*x)))/N;
  ifc(j)=-sum((y.*sin(2*pi*xi(j)*x)))/N;
end
```

Observation : Pour N, P grand (et $M = N/2$), on approche bien *sinc* pour la fonction porte ci-dessus. $max(0, sign(1/2 - abs(x))$

Observations : période N, étend les coeff de Fourier, pair pour $Re(\hat{f})$, impair pour $Im(\hat{f})$

$P = 1$, on retrouve les coefficients de Fourier. P grand, on voit $\hat{f}$ comme une fonction continue qui étale les coefficients de Fourier.

TFD

Définition 16 *Si f est connue en les points n/N avec $n \in \{0, \ldots, N-1\}$ on calcule $\hat{f}(\xi) = (1/N) \sum_{n=0}^{N-1} f(n/N) e^{-i2\pi\xi n/N}$.*

En fait dans la définition habituelle de la TFD (et en particulier sa version "rapide" la FFT) il n'y a pas le $1/N$ qui est en revanche présent dans la TFD inverse. En mettant le $1/N$ ici, il est plus facile d'interpréter le TFD comme une version discrète de la transformée de Fourier.

Les calculs coïncident avec ceux effectués pour l'approximation discrète des coefficients de Fourier. Pour $N = 4$, et $f = (1, 1, 0, -1)$ en $0, 1/4, 1/2, 3/4$, on a vu que $tfd(0) = tfd(1) = tfd(3) = 0$, $tfd(2) = 1$. Et pour $g = (1, 1, 0, -1)$, $tfd(0) = 1/4$, $tfd(1) = 1/4 - i/2$, $tfd(2) = 1/4$, $tfd(3) = 1/4 + i/2$.

En fait g c'est plutôt $[0, 1, 0, -1]$ qui donne $[0, -i/2, 0, i/2]$.

```
......
plot(real(fft(y)))
plot(imag(fft(y)))
```

Si on compare avec l'approximation discrète pour $P = 1$, on voit que seul le $1/N$ diffère.

Observation : pour g qui est impaire, on a ici la partie rélle presque nulle ($P = 1$), on avait trouvé une fonction non nulle pour P plus grand.

1.4.2 Transformée inverse

Cas continu

Définition 17 *Soit g une fonction mesurable sur $\mathbf{R}$ et telle que $\int_{\mathbf{R}} |g(\xi)| d\xi < +\infty$, alors on appelle transformée de Fourier inverse $\check{g}(x) = \int_{\mathbf{R}} g(\xi) e^{i2\pi x\xi} d\xi$.*

Théoreme 18 *Si f est une fonction mesurable sur $\mathbf{R}$ et si $\int_{\mathbf{R}} |f(x)| dx < +\infty$ et si $\int_{\mathbf{R}} |\hat{f}(\xi)| d\xi < +\infty$ alors $\check{\hat{f}}(x) = f(x)$ presque partout.*

Rq : presque partout sens à préciser....

Rq : pas évident de montrer que $\check{(}sinc)(x) = \chi_{-1/2,1/2}$. Et ce n'est même pas une conséquence du théorème car $\int_{\mathbf{R}} sin(\pi\xi)/(\pi x)$ est simplement convergente : (on ne regarde que le coté $+\infty$)

$$\int_1^X \frac{sin(\pi\xi)}{\pi\xi} d\xi = \left[\frac{-cos(\pi\xi)}{\pi^2\xi}\right]_1^X - \int_1^X \frac{cos(\pi\xi)}{\pi^2\xi^2} d\xi$$

le premier terme tend vers $1/\pi^2$, le second est convergent par majoration par $1/(\pi\xi)^2$, pour l'eventuelle convergence absolue, on a $|sin(y)| \geq sin^2(y)$ donc

$$\int_1^X \frac{|sin(\pi\xi)|}{\pi\xi}d\xi \geq \int_1^X \frac{\sin^2(\pi\xi)}{\pi\xi}d\xi = \int_1^X \frac{1-\cos(2\pi\xi)}{2\pi\xi}d\xi = \int_1^X \frac{1}{2\pi\xi}d\xi - \int_1^X \frac{\cos(2\pi\xi)}{2\pi\xi}d\xi$$

le second terme est convergent par la même intégration par partie qu'avant mais le premier terme est divergent.

Si on veut on peut utiliser des série mais c'est plus compliqué.

$$\int_1^\infty \frac{sin(\pi\xi)}{\pi\xi}d\xi = \sum_{n=1}^\infty \int_n^{n+1} \frac{sin(\pi\xi)}{\pi\xi}d\xi = \sum_{n=1}^\infty \frac{(-1)^n}{\pi}\int_n^{n+1} \frac{|sin(\pi\xi)|}{\xi}d\xi$$

qui est donc une série alternée dont le terme général est décroissant puisque

$$\int_{n+1}^{n+2} \frac{|sin(\pi\xi)|}{\xi}d\xi = \int_n^{n+1} \frac{|sin(\pi\xi)|}{\xi+1}d\xi \leq \int_n^{n+1} \frac{|sin(\pi\xi)|}{\xi}d\xi$$

Approximation discrète

On notera que le phénomème de repliement de spectre est aussi présent ici. On ne peut donc considérer que $N/2 \leq \xi \leq N/2$.

On utilise à nouveau la méthode des rectangles à gauche. On discrétise $[-N/2, N/2]$ par intervalles de taille $1/P$, on a donc pour $n \in \{0, \ldots, NP-1\}$ (on suppose pour simplifier que N ou P est pair)

$$f(n/N) \approx \frac{1}{P}\sum_{k=-NP/2}^{NP/2-1} \hat{f}(k/P)e^{i2\pi(k/P)(n/N)}$$

Si on choisit $P = 1$, ce qui signifie que l'on n'utilise les valeurs de $\hat{f}(\xi)$ que pour ξ entier, on peut alors interpéter cette approximation plutôt comme celle de la série de Fourier.

```
clear

N=100
M=50
P=10
x=(0:(N-1))/N;

y1=cos(4*pi*x);

y2=-sign(x-1/2);

%y=y1;
y=y2;

xi=(-M*P:M*P)/P:
for j=1:length(xi)
  rfc(j)=sum((y.*cos(2*pi*xi(j)*x)))/N;
  ifc(j)=-sum((y.*sin(2*pi*xi(j)*x)))/N;
end

for j=1:N
  rfci(j)=sum(rfc.*cos(2*pi*xi*x(j))-ifc.*sin(2*pi*xi*x(j)))/P;
  ifci(j)=sum(rfc.*sin(2*pi*xi*x(j))+ifc.*cos(2*pi*xi*x(j)))/P;
end
plot(x,y,x,fci)
```

On voit la convergence de $\check{f}$ vers f quand M croit ($M < N/2$). On voit que pour M proche de $N/2$ l'approximation est meilleure pour P grand. Si $M > N/2$, repliement de spectre et l'approximation est très mauvaise.

TFD inverse

Définition 19 *Si $\hat{f}$ est connue en les points k avec $k \in \{0,\dots,N-1\}$ on calcule $\check{f}(n/N) = \sum_{k=0}^{N-1} \hat{f}(k)e^{i2\pi kn/N}$ pour $n \in \{0,\dots,N-1\}$.*

Ceci correspond à l'approximation discrète de la transformée inverse pour $P = 1$. Ici le repliement de spectre a été utilisé autrement : au lieu de considérer $-N/2 \le \xi \le N/2$, on choisit $0 \le \xi \le N$ (périodicité de $\check{f}$ discret).

Pour la TFD inverse, il y a en général un $1/N$ qui vient du fait que ce coefficient n'a pas été mis dans la TFD directe. On ne le met pas ici pour que l'on ait bien la méthode des rectangles à gauche.

On retrouve exactement les calculs déjà vus pour les séries de Fourier dans le cas discret. Pour $N = 4$ et $f = (1,-1,1,-1)$, on a vu que $tfd = (0,0,1,0)$ et pour $g = (1,1,0,-1)$, $tfd(1/4, 1/4 - i/2, 1/4, 1/4 + i/2)$. Et la TFD inverse est : pour f, $Re(tfdi)(0,1/4,1/2,3/4) = (1,-1,1,-1)$ et pour g, $Re(tfdi)(0,1/4,1/2,3/4) = (1,1,0,-1)$. Le fait que l'on retrouve f et g n'est pas un hasard :

Dans ce cas discret on a même :

Théoreme 20 *$\check{f}(n/N) = f(n/N)$ pour $n \in \{0,\dots,N-1\}$.*

Démo :

$$\check{f}(n/N) = \sum_{k=0}^{N-1} \hat{f}(k)e^{i2\pi kn/N} = \sum_{k=0}^{N-1} e^{i2\pi kn/N}\left(\frac{1}{N}\sum_{m=0}^{N-1} f(m/N)e^{-i2\pi km/N}\right)$$
$$= \frac{1}{N}\sum_{k=0}^{N-1}\sum_{m=0}^{N-1} f(m/N)e^{i2\pi kn/N}e^{-i2\pi km/N} = \frac{1}{N}\sum_{m=0}^{N-1} f(m/N)\sum_{k=0}^{N-1} e^{i2\pi k(n-m)/N}$$

si $m = n$ alors

$$\sum_{k=0}^{N-1} e^{i2\pi k(n-m)/N} = \sum_{k=0}^{N-1} 1 = N,$$

si $m \neq n$ alors

$$\sum_{k=0}^{N-1} e^{i2\pi k(n-m)/N} = \sum_{k=0}^{N-1} (e^{i2\pi(n-m)/N})^k = \frac{1-(e^{i2\pi(n-m)/N})^N}{1-e^{i2\pi(n-m)/N}} = \frac{1-e^{i2\pi(n-m)}}{1-e^{i2\pi(n-m)/N}} = 0$$

car n et m sont entiers. On remarque que l'on a retrouvé (en mettant $-m$ à la place de m), l'orthogonalité des $e^{i2\pi kn/N}$. Ainsi on a donc

$$\check{f}(n/N) = \frac{1}{N}[f(n/N)N + 0] = f(n/N).$$

On notera que l'on peut considérer ce théorème comme une inégalité algébrique abstraite et l'utiliser sur n'importe quel intervalle (pas seulement sur $[0,1]$).

```
.......
real(ifft(fft(y))
```

On voit que la seule différence est bien le $1/N$.

Filtrage et FFT

On veut mettre en place un filtre passe bas : il coupe les fréquences élevées et ne laisse que les basses fréquences. Dans la réalité les filtres sont progressifs. Ici $\hat{f}(\xi)\chi_{\{|\xi|\le M\}}$.

Pour simplifier un peu, on suppose N impair dans ce qui suit.

On reprend le calcul effectué précedemment pour les séries de Fourier :

$$\sum_{k=0}^{N-1} \hat{f}(k)e^{i2\pi kj/N} = \sum_{k=-(N-1)/2}^{(N-1)/2} \hat{f}(k)e^{i2\pi kj/N}$$

Si l'on tronque le spectre au delà de $M \leq (N-1)/2$:

$$\sum_{k=-M}^{M} \hat{f}(k)e^{i2\pi kj/N} = \sum_{k=-M}^{-1} \hat{f}(k)e^{i2\pi kj/N} + \sum_{k=0}^{M} \hat{f}(k)e^{i2\pi kj/N}$$
$$= \sum_{k=0}^{M} \hat{f}(k)e^{i2\pi kj/N} + \sum_{k=N-M}^{N-1} \hat{f}(k)e^{i2\pi kj/N}$$

Il s'agit donc d'enlever les composantes de $\hat{f}$ pour k compris entre $M+1$ et $N-M-1$. (dessin)

```
clear

N=100
M=50
x=(0:(N-1))/N;

y1=cos(4*pi*x);

y2=-sign(x-1/2);

%y=y1;
y=y2;

xi=0:N-1;
for j=1:N
  rfc(j)=sum((y.*cos(2*pi*xi(j)*x)))/N;
  ifc(j)=-sum((y.*sin(2*pi*xi(j)*x)))/N;
end

rfc(M+1:N-M-1)=zeros(1:N-2*M-1);

for j=1:N
  fci(j)=sum(rfc.*cos(2*pi*xi*x(j))-ifc.*sin(2*pi*xi*x(j)));
end
plot(x,y,x,fci)
```

On observe.....

1.5 Applications

1.5.1 Débruitage de signal en dimension 1

Pour $f(x) = \cos(4\pi x)\sin(8\pi x)$ bruitée et $N = 100$, on retrouve le fait que le débruitage est optimal pour $M = 5$. Au dessous la fonction est trop altérée et au dessus trop bruitée.

La situation est beaucoup plus difficile pour l'autre exemple.

```
clear
N=101
M=5

x=0:1/N:1-1/N;
y1=cos(4*pi*x)*sin(8*pi*x);
y2=-sign(x-1/2);

y=y1+(ran(1,N)-1/2);
```

```
tfdy=fft(y);

for k=M+1:N-M-1
  tfdy(k)=0;
end

plot(x,y1,x,y,x,ifft(tfdy))
```

1.5.2 Débruitage d'image et dimension 2

Pour une fonction sur $\mathbf{R}^N$, on a $T \in \mathbf{R}^N$ dont chaque coordonnée vérifie la propriété unidimensionnelle ci-dessus.

On remplace $x\xi$ par le produit scalaire $< x, \xi >$. A peu près tout reste pareil (il y a juste un $1/N^2$ dans la TFD et des sommes et intégrales doubles).

```
N=5
t=0:1/N:1-1/N;
k=0:N-1;
y=(cos(2*pi*t))'*sin(4*pi*t);

mesh(t,t,y)

tfdy=fft2(y)

mesh(k,k,tfdy)
```

1.5.3 DAC

1. On part du PCM : qui est comme notre y mais encodé sur 3 bit (par arrondi). (pour le CD c'est 16 bit)
2. On convertit en 1 bit haché (à haute fréquence : $2^3 = 8$ fois la fréquence de base). (pour le CD cela donnerait $44100 * 2^{16} = 2,89$ GHz)
3. On filtre par passe bas.

simple

```
clear

N=101
S=2^5
% shannon dit M<=N*(S-1)/2 et M<=N/2
Mmax = N*(S-1)/2
Mmax = N/2
M=50
x=(0:(N-1))/N;

y1=cos(4*pi*x)+sin(20*pi*x);

y2=-sign(x-1/2);

y=y1;
%y=y2;

% arrondi
ypcm=round((y+2)/4*(S-1));

ya=ypcm/(S-1)*4-2;
```

```
plot(x,y,x,ya)

bs=zeros(S,S-1);
for j=1:S
  for l=S-j+1:S-1
    bs(j,l)=1;
  end
end

% creation du bitstream
ybs=zeros(1,N*(S-1));
for j=1:N-1
  ybs(1,1+(j-1)*(S-1):S-1+(j-1)*(S-1))=bs(ypcm(j)+1,:);
end

% x raffine
xr=(0:(S-1)*N-1)/((S-1)*N);

% coeff de fourier ou tfd
ybsc=fft(ybs);

ybsc(1,M+1:N*(S-1)-M-1)=zeros(1,1:N*(S-1)-2*M-1);

yf=real(ifft(ybsc));
% plot(xr,ybs,xr,yf)

yr=yf*4-2;

plot(x,y,xr,yr)
```

compliqué

```
clear;

clf();
hold on;
axis([0,2*pi,-1,1])

Nc=1000;
N=50
M=2^8

tc=0:2*pi/Nc:2*pi;
fc=(sin(tc)+cos((N/3)*tc)/2+sin(3*tc)/2)/2;
plot(tc,fc,"b")

t=0:2*pi/N:2*pi;
f=(sin(t)+cos((N/3)*t)/2+sin(3*t)/2)/2;
%plot(t,f)
for i=1:N
  xf(i*2-1)=t(i);
  xf(i*2)=t(i+1);
  yf(i*2-1)=f(i);
  yf(i*2)=f(i);
end
xf=xf-pi/N;
```

```
plot(xf,yf,"r");

% pcm
fd=(floor((f*M)-0.5)+1)/M;
for i=1:N
  yfd(i*2-1)=fd(i);
  yfd(i*2)=fd(i);
end
plot(xf,yfd,"g")

% surechantillonage, il faut S>=M pour permettre toutes les valeurs
S=M*2
for i=1:N+1
    fds((i-1)*S+1:i*S)=fd(i)*ones(1,S);
    ts((i-1)*S+1:i*S)=(t(i)-(pi/N))+(0:S-1)*(2*pi/N)/S;
end

% dac 1 bit
% 0 -> 00000000 0/0
% 1 -> 00000001 1/8
% 2 -> 00010001 2/8
% 3 -> 00100101 3/8
% 4 -> 01010101 4/8
% 5 -> 01011011 5/8
% 6 -> 01110111 6/8
% 7 -> 01111111 7/8
% 8 -> 11111111 8/8
accumulateur=0;
for i=1:N*S
  accumulateur=accumulateur+(fds(i)*M+M);
  if (accumulateur>=2*M)
    fdsc(i)=1;
    yfdsc(i*2-1)=1;
    yfdsc(i*2)=1;
    accumulateur=accumulateur-2*M;
  else
    fdsc(i)=0;
  end
end
spectre=fft((fdsc-1/2)*2);
val=floor(N*S/2)-1
% la "bonne" valeur du filtre est val- la plus grand frequence du
% signal.
%filtre=1578;
%filtre=min(filtre,val);
%spectre(2+val-filtre:N*S-val+filtre)=0;
spectre(2+floor(N/2+2):N*S-floor(N/2+2))=0;
fdscl=real(ifft(spectre));
for i=1:N*S;
  yfdscl(i*2-1)=fdscl(i);
  yfdscl(i*2)=fdscl(i);
  xfs(i*2-1)=ts(i);
  xfs(i*2)=ts(i+1);
end
plot(xfs,yfdscl,"k");
```

1.5.4 FM

Fichier http ://palosaari.fi/linux/v4l-dvb/rtl2832u_fm/ 83Mo 21s échantillonné à la fréquence 2048000Hz (identifié grâce à la porteuse à 19kHz).

FFT : spectre centré en 0 : la porteuse qui donne la fréquence de la radio a déjà été supprimée par le tuner.

Fichier entrelacé donnant pour chaque échantillon, la partie rélle et la partie imaginaire (I et Q en traitement du signal) codées sur un octet (numérisation 8 bits) non signé : on enleve 128 pour transformer en valeur signée (on peut diviser par 128 pour normaliser le signal entre -1 et 1).

Modulation FM : si le signal d'origine est $v(t)$ (réel), on module en fréquence (ie c'est la fréquence qui varie, l'amplitude est constante)

$$u(t) = e^{i2\pi\left(f_p t + \alpha \int_{-\infty}^{t} v(s)ds\right)}$$

où f_p est la fréquence de la porteuse et α un coefficient. La fréquence du signal modulé est donc

$$f_p + \alpha \frac{1}{t} \int_{-\infty}^{t} v(s)ds$$

c'est donc l'amplitude de v qui donne la fréquence du signal module.

On veut démoduler et donc retrouver v à partir de u (il y a de nombreux moyens notamment électronique pour cela). Pour $w(t) = e^{\phi(t)}$ on a $Re(w) = \cos(\phi(t))$ et $Im(w) = \sin(\phi(t))$ donc $\phi(t) = \arctan(Im(w)/Re(w))$ donc ici

$$2\pi \left(f_p t + \alpha \int_{-\infty}^{t} v(s)ds\right) = \arctan(Re(u)/Im(u))$$

et si l'on dérive (de toute façon on ignore les constantes multiplicative et les constants additives seront perdues lors du filtrage)

$$2\pi \left(f_p + \alpha v(t)\right) = \frac{(Re(u)/Im(u))'}{1 + (Re(u)/Im(u))^2} = \frac{(Re(u)'Im(u)) + Re(u)Im(u)'}{Re(u)^2 + Im(u)^2}$$

(on peut aisément en faire une version discrète en faisant la différence de deux échantillons successifs.

Version mono rapide

```
clear
% 2^20 echantillons font 0,51s
% total 21 s
fe=2048000;
N=2^20;
N=5*N;
Ns2=ceil(N/2);
xi(1:Ns2)=(1:Ns2)-1;
xi(Ns2+1:N)=(Ns2+1:N)-(N+1);
xi=xi/N*fe;
duree = N/fe
fid = fopen('/tmp/franceinfo.bin');
[v, count] = fread(fid,[2,N],'unsigned char');
% scilab fid=mopen('/tmp/franceinfo.bin')
% scilab x=mget(2*N,'uc',fid);
% scilab mclose(fid);
% scilab v(1,1:N)=x(1:2:2*N-1);
% scilab v(2,1:N)=x(2:2:2*N);
% centrage du signal
v = (v-127.5);
```

```
w=v(1,:)+i*v(2,:);

filtre=max(0,sign(100000-abs(xi)));
wf=ifft(filtre.*fft(w));
v(1,:)=real(wf);
v(2,:)=imag(wf);

u = (v(1,2:N).*(v(2,2:N)-v(2,1:N-1))-v(2,2:N).*(v(1,2:N)-v(1,1:N-1)))
./(v(1,2:N).^2+v(2,2:N).^2);

% perte d'un echantillon lors du calcul de derive
u(N)=u(N-1);

% filtre passe bande entre 30Hz et 15kHz conserver que G+D.
filtre2 = max(0,sign(15000-abs(xi))) .* max(0,sign(abs(xi)-30));

fftuf = filtre2.* fft(u);
% on veut arriver a sous echantilloner a 48kHz donc freq = 24kHZ
se = max(0,sign(24000-abs(xi)));
indices=find(se);
ufse=ifft(fftuf(indices));
ufse=ufse/max(abs(ufse));

wavwrite(ufse,48000,16,'audio.wav')
```

Version stéréo

```
clear
total=time();
tempsinit=time();
% 2^20 echantillons font 0,51s
% total 21 s
fe=2048000;
N=2^20;
fid = fopen('rtl2832u_fm_sample_FIXED.bin');
%fseek(fid, N*10, SEEK_SET);
N=40*N;
[v, count] = fread(fid,[2,N],'unsigned char');
duree = N/fe
% scilab fid=mopen('/tmp/franceinfo.bin')
% scilab x=mgeti(N,'uc',fid);
% scilab mclose(fid);
% scilab v(1,1:N/2)=x(1:2::N-1);
% scilab v(2,(1:N/2)=x(2:2:N);
%v = (v-128)/128;
%en fait division par 128 inutile dans le calcul de delta
v = (v-128);
tempsinit=time()-tempsinit
% 8.5s
```

Compte tenu de la lenteur, on commence par diviser la fréquence par 16 ce qui fait que l'on passe de 2048000 Hz en 8 bits à 128000 Hz en 8+4 bits = 12 bits (car $2^4 = 16$) en faisant la moyenne de 16 échantillons.

```
tempsmoyenne=time();
% filtrage : moyenne de 8 valeurs
%v2 = zeros(2,N/8);
```

```
facteur=16;
%a = [1,0,0,0,0,0,0,0];
a = zeros(1,facteur);
a(1) = 1;
%b = [1,1,1,1,1,1,1,1]/8;
b = ones (1,facteur)/facteur;
v2(1,:) = filter(b,a,v(1,:));
v2(2,:) = filter(b,a,v(2,:));
clear v;
v(:,1:N/facteur) = v2(:,facteur:facteur:N);
clear v2;
N=N/facteur
fe=fe/facteur
tempsmoyenne=time()-tempsmoyenne
% 11.5s
```

Ensuite on peut faire la dérivé discrète : on a finit de démoduler la FM et à constante additive et multiplicatives près on a le signal démodulé.

```
tempsderive=time();
delta = (v(1,2:N).*(v(2,2:N)-v(2,1:N-1))-v(2,2:N).*(v(1,2:N)-v(1,1:N-1)))
./(v(1,2:N).^2+v(2,2:N).^2);
tempsderive=time()-tempsderive
% 0.35s

% perte d'un echantillon lors du calcul de derive
N=N-1;
```

On calcule la FFT de la dérivée (et on remet le spectre sur $[-Fe/2, Fe/2]$ (indépendant du nombre d'échantillon)

```
Ns2=ceil(N/2);
l(1:Ns2)=(1:Ns2)-1;
l(Ns2+1:N)=(Ns2+1:N)-(N+1);
l=l/N*fe;

tempsfftdelta=time();
fftdelta=fft(delta);
tempsfftdelta=time()-tempsfftdelta
% 2s
```

Le signal démodulé contient la somme des canaux gauche et droite dans la bande de fréquence -15 à 15 kHz (on enleve en -30 et 30 Hz donc la partie constante). On filtre la FFT par un filtre passe bande en mettant à 0 ce qui dépasse, puis on fait la FFT inverse.

```
% filtre passe bande entre 30Hz et 15kHz conserver que G+D.
tempsfiltregpd=time();
filtre3 = fftdelta .* max(0,sign(15000-abs(l))) .* max(0,sign(abs(l)-30));
tempsfiltregpd=time()-tempsfiltregpd

% calcul de G+D
tempsifftgpd=time();
gpd = ifft(filtre3);
tempsifftgpd=time()-tempsifftgpd
% 1.5s
```

On a donc de la FM monophonique. Pour la stéréophonie : le signal qui contient la différence entre les canaux gauche et droite est situé dans la bande de fréquence 23kHz à 53 kHz (largeur

30kHz comme pour la partie mono) et donc centré en 38kHz. Pour le ramener autour de 0Hz, on utilise la porteuse à 19kHz et on multiplie 2 fois. En effet multiplier par une porteur translate les fréquences

$$A(t)e^{i2\pi f(t)t}\cos(2\pi f_p t) = \frac{A(t)}{2}\left(e^{i2\pi(f(t)+f_p)t} + e^{i2\pi(f(t)-f_p)t}\right)$$

on a donc après une première multiplication, une bande entre 4 et 34 kHz centré en 19 kHz et une entre 42 et 72 kHz centrée en 57 kHz. Et après une seconde multiplication (on ignore la partie négative symétrique), une bande entre -15 et 15 kHz centrée en 0, une entre 23 et 53 kHz centrée en 38, et une entre 61 et 91 kHz centrée en 76 kHz. Il ne reste donc qu'à filtrer entre -15 et 15 kHz (en enlevant la partie en dessous de 30 Hz)

```
% extraction de la porteuse de 19kHz pour recuperer G-D
% passebande entre 18.5 et 19.5 kHz
tempsfiltre=time();
filtre3 = fftdelta .* max(0,sign(19050-abs(l))) .* max(0,sign(abs(l)-18950));
tempsfiltre=time()-tempsfiltre

% porteuse a 19kHz
% normalisation pour avoir max(real(p19))=1
tempsifft=time();
p19 = ifft(filtre3);
maxp19 = max(abs(real(p19)));
p19 = p19/maxp19;
tempsifft=time()-tempsifft

% on applique 2 fois la porteuse pour transport G-D de autour de 38kHz
% vers autour de 0kHz
gmd=delta.*p19.*p19;

tempsfft=time();
fftgmd=fft(gmd);
tempsfft=time()-tempsfft

% faire un filtre passe bas a 15kHz (ou passe bande avec 30Hz) pour ne
% conserver que G-D.
tempsfiltre=time();
filtre3 = fftgmd .* max(0,sign(15000-abs(l))) .* max(0,sign(abs(l)-30));
tempsfiltre=time()-tempsfiltre

% calcul de G+D
tempsifft=time();
gmd = ifft(filtre3);
tempsifft=time()-tempsifft
```

Il ne reste plus qu'a soustraire et additionner pour retrouver les canaux gauche et droite (amplitudes à ajuster).

```
% G = [(G+D)+(G-D)]/2
wave(1,:) = (real(gpd)+real(gmd))/2;
maxg = max(abs(wave(1,:)));
% D = [(G+D)-(G-D)]/2
wave(2,:) = (real(gpd)-real(gmd))/2;
maxd = max(abs(wave(2,:)));
maxgd = max(maxg,maxd)
wave = wave / (1.5 * maxgd);

tempsmoyenne=time();
```

```
% fe 2048000/48000 = 42.6667
% deja fait 8 reste 5
% filtrage : moyenne de 5 valeurs
facteur=3;
a = zeros(1,facteur);
a(1) = 1;
b = ones (1,facteur)/facteur;
wave2(1,:) = filter(b,a,wave(1,:));
wave2(2,:) = filter(b,a,wave(2,:));
petitwave(:,1:floor(N/facteur)) = wave2(:,facteur:facteur:N);
tempsmoyenne=time()-tempsmoyenne

% fe 2048000/48000 = 42.6667
%decim2=time();
%coef = fe/48000;
%nb = floor((N-2)/coef);
%m = 1+floor((0:nb)*coef);
%petitwave(:,1:nb+1) = wave(:,m);
%decim2=time()-decim2

wavwrite(petitwave',48000,16,'audio.wav')

total=time()-total
% 30s (plus long que temps reel)
```

Chapitre 2

Optimisation linéaire et simplexe

2.1 Problème

Il s'agit de trouver un optimum dans des problèmes linéaires. (De façon un peu inattendu cela peut même permettre de résoudre les sudoku.)

Les problèmes classiques sont "économiques" : comment maximiser ses gains sous contraintes.

L'exemple le plus simple serait de maximiser x sur $\mathbf{R}$: impossible on trouve $+\infty$ et ∞. Alors on peut essayer sur un borné $[1, 2]$ et on trouve en effet 1 et 2. Les conditions sont linéaires : $x \leq 2$ et $x \geq 1$.

On cherche à maximiser J (minimiser $-J$) sur U un convexe de $\mathbf{R}^N$.

Algorithme du simplexe : Dantzig (1963).

Les arguments théoriques viennet du livre de Ciarlet.

2.2 Un premier cas très simple

2.2.1 Idées

Maximiser

$$J(u_1, u_2) = 2u_1 + u_2$$

sur $u_1 \geq 0$, $u_2 \geq 0$ tels que $u_1 + u_2 \leq 1$. que l'on peut interpreter comme le choix de production entre deux produit de façon à maximiser le gain J. On voit bien que comme u_1 rapporte plus que u_2 la solution est $u_1 = 1$ et $u_2 = 0$, ce qui rapporte 2.

Notons U le convexe donnant les conditions sur u_1 et u_2. U est un triangle (dessin). Et on observe qu'il a deux types de cotés différents : deux cotés, $u_1 = 0$ et $u_2 = 0$ parallèles aux axes et un oblique $u_1 + u_2 = 1$. Et les sommets sont $(0, 0)$, $(1, 0)$ et $(0, 1)$. Pour l'algorithme qui va suivre, il est plus pratique que tous les cotés (et les sommets) jouent le même rôle :

On commence par transformer le problème en un problème équivalent avec condition "égalité" :

Maximiser

$$J(u_1, u_2) = 2u_1 + u_2$$

sur $u_1 \geq 0$, $u_2 \geq 0$, $u_3 \geq 0$ tels que $u_1 + u_2 + u_3 = 1$.

On renote U l'ensemble $u_1 \geq 0$, $u_2 \geq 0$, $u_3 \geq 0$ tels que $u_1 + u_2 + u_3 = 1$. Il s'agit de l'intersection du plan (dimension 2 dans un espace de dimension 3) avec les trois plans $u_1 = 0$, $u_2 = 0$, $u_3 = 0$. Le résultat est un triangle qui a 3 sommets : $(1, 0, 0)$, $(0, 1, 0)$ et $(0, 0, 1)$ (dessin).

On va voir que les extrema sont atteints sur les sommets. Et on va donc se déplacer de sommet en sommet.

2.2.2 Mise en oeuvre

Partons du sommet "trivial" : $(0, 0, 1)$ (cas où on ne produit rien $u_1 = u_2 = 0$ et ou on ne gagne rien $J = 0$, on est donc loin de la solution qui rapporte 2). Il se traduit par $(0, 0)$ dans les variables d'origines.

L'ensemble U étant un triangle dans un plan (dim 2) dans l'espace (dim 3), on peut exprimer une des variables à partir des deux autres : $u_3 = 1 - u_1 - u_2$ (une équation à une inconnue et deux paramètres). On a deux variables principales u_1 et u_2 et une variable déduite u_3 et J ne dépend que des variables principales (on peut s'y ramener). Les variables principales sont nulles.

Pour améliorer le gain, il faut modifier une des variables (principales) intervenant dans J et (puisqu'on part de 0) ayant un coefficient positif : u_1 ou u_2. On choisit ici u_2 (pour voir une convergence en 2 étapes).

On se déplace dans le plan en faisant bouger u_2 et en déduisant u_3 grâce à l'équation : $u_3 = 1 - u_1 - x - 2$. Si on a $u_2 = \theta$, alors $u_3 = 1 - \theta$ mais comme $u_3 \geq 0$, il faut $u_2 = \theta \leq 1$. On choisit $u_2 = \theta = 1$ puisqu'on veut être sur le bord de U. On arrive donc au point $(0, 1, 0)$, qui est bien un des trois sommets de U,et pour lequel $J = 1$ le gain a bien augmenté.

On ajuste le rôle des variables : u_2 devient une variable déduite et u_1 et u_3 les variables principales. On récrit J en fonction de u_1 et u_3 : on a $u_2 = 1 - u_1 - u_3$ donc $J = 2u_1 + u_2 = 2u_1 + 1 - u_1 - u_3 = 1 + u_1 - u_3$.

On peut à nouveau se déplacer vers 2 sommets en faisant varier u_1 ou u_3. Mais la situation s'améliore au sens où il n'y a plus qu'une direction pouvant faire augmenter J : u_1 (faire augmenter u_3 ferait diminuer J). On fait augmenter u_1 en suivant $u_2 = 1 - u_1 - u_3$ donc en faisant diminuer u_2, la limite $u_2 = 1 - \theta \geq 0$ donne $u_1 = \theta \leq 1$. On choisit donc $u_1 = 1$ et on est donc au sommet $(1, 0, 0)$ pour lequel $J = 2$.

Peut-on continuer ? Il faut aussi ajuster les variables : u_1 devient une variable déduite et u_2 et u_3 les variables principales : $u_1 = 1 - u_2 - u_3$ donc $J = 1 + u_1 - u_3 = 1 + 1 - u_2 - u_3 - u_3 = 2 - u_2 - 2u_3$. Cette fois les coefficients de u_2 et u_3 sont négatifs ou nuls : on est sur que J ne peut plus augmenter et on a donc fini. $(1, 0, 0)$ est bien une solution assurant le maximum. Soit $(1, 0)$ en variables d'origine et $J = 2$.

Donner un exemple 3x3 de pivot de Gauss et Gauss-Jordan pour expliquer la mise à jour de a et e.

```
clear

% inconnues positives
% u_1 + u_2 <= 1

% j(v)= 2 u_1 + u_2

% 1 inconnue supplementaire pour transformer les contraintes en =

% inconnues positives
% u_1 + u_2 + u_3 = 1
% c u = d
c(1,1)=1;
c(1,2)=1;
c(1,3)=1;

d(1,1)= 1;

% j(v)= 0 + 2 u_1 + u_2 + 0 u_3
% j= au +e
a(1,1)=2;
a(1,2)=1;
a(1,3)=0;

e=0;

% et ainsi avoir comme sommet evident 0 0 1
u(1,1)=0;
u(2,1)=0;
```

```
u(3,1)=1;

J=e+a*u

% il faut distinguer les inconnues principales des inconnues deduites
deduite=3;

% ici J ne depend pas de u_3

atteint=0;

while (atteint==0)
  % si les a reels sont negatifs ou nuls, le max est atteint
  atteint=1;
  for i=1:3
    if ((i~=deduite) & (a(1,i)>0))
      atteint=0;
      arete=i;
    end
  end

  % si le max n'est pas atteint on se deplace sur une des aretes
  % u(arete) va augmenter de theta et les u(deduite) diminuer \'a 0
  if (atteint==0)
    theta=d(1,1)/c(1,deduite);
    % la longueur maximale du deplacement est theta
    % on se deplace dans la direction arete en ne modifiant que la
    % variable deduite
    u(arete,1)=theta;
    u(deduite,1)=0;

    % on echange remplace u_arete par u_deduite dans J
    % J=\sum a_i u_i et u_arete = (d-\sum i\not=arete c_i u_i)/c_arete
    % J=a_arete d/c_arete + \sum i (a_i-a_arete c_i/c_arete) u_i
    coef= a(1,arete)/c(1,arete);
    e = e + coef * d(1,1) ;
    a(1,:)= a(1,:) - coef * c(1,:);
    deduite=arete;
    J=e-a*u
    u
  end
end
```

2.3 Quelques résultats

Il s'agit maintenant de se placer dans un cadre plus général.

Définition 21 *On dit qu'un ensemble U est convexe si pour tout u et $v \in U$ et $0 \leq \theta \leq 1$, on a $\theta u + (1 - \theta v) \in U$*

Exemple : $u \geq 0$ et $u_1 + u_2 + u_3 = 1$.....

Définition 22 *On dit que J est affine sur $\mathbf{R}^N$ si $J(u) = e + au$ pour $e \in R$ et $a \in \mathbf{R}^N$*

Exemple : $J(u) = 2u_1 + u_2$

2.3.1 Sommets

La propriété suivante est difficile à illustrer même pour l'exemple précédent, car ce n'est pas évident d'avoir une norme dans l'hyperplan (en fait si...) et c'est très éloigné de ce cours. A oublier ?

Propriété 23 *Pour J affine et non constante, les extrema ne peuvent être à l'intérieur du convexe U.*

Démo : Si $J(u) = e + a * u$ avec $a \neq 0$ (si $a = 0$, $J = e$ et J est extremum en tout point de U) est maximum en u et que u est à l'interieur de U alors il y a un $\varepsilon > 0$ tel que $B(u, \varepsilon) \subset U$. Mais cette boule contient $\varepsilon a/(2\|a\|)$ et $J(u + \varepsilon a/(2\|a\|)) = J(u) + \varepsilon\|a\|/2 > J(u)$, donc J n'atteignait pas son max en u.

On va voir (et on a observé) que le max est même atteint sur les sommets.

Définition 24 *Un point u est un sommet d'un ensemble U convexe dont les bords sont "affines", si pour tout v et w dans U et $0 < \lambda < 1$, $u = \lambda v + (1 - \lambda)w$ alors $v = w = u$.*

(dessin).

Propriété 25 *Si $U = \{u \geq 0, Cu = d\}$, alors les sommets sont les points de U tels que les colonnes C_i correspondant aux $u_i > 0$ sont libres.*

On voit que plus il y a de $u_i > 0$ plus la condition est difficile à réaliser. Dans l'example précédant, $C = (1, 1, 1)$, les colonnes sont toutes identiques et ne peuvent être libres s'il y en a plus qu'une donc les sommets sont de la forme $(0, 0, ?)$ et $Cu = d$ donne $(0, 0, 1)$ (et les deux autres sommets).

Si on considère maintenant $U = \{u \in \mathbf{R}^4, u \geq 0, u_1 + u_2 + u_3 = 1, 4u_1 + u_2 + u_4 = 3\}$, on a vérifie que U est convexe et

$$C = \begin{pmatrix} 1 & 1 & 1 & 0 \\ 4 & 1 & 0 & 1 \end{pmatrix}$$

La matrice C ayant 2 lignes, elle ne peux pas comporter plus de 2 colonnes libres. Donc un sommet a au plus 2 coordonnées non nulles. Vérifier que C_2, C_3, C_4 ne sont pas libres mais que C_3, C_4 le sont.

Comme le rang (à préciser) est ≤ 2 puisque C a deux lignes et ≥ 2 puisque C contient la matrice identité de taille 2. On peut calculer 2 coordonnées de u dès que l'on connait deux autres coordonnées de u. On peut donc avoir 2 coordonnées nulles (les 2 autres sont alors connues). Les sommets sont (à calculer) parmi $(0, 0, 1, 3)$, $(0, 1, 0, 2)$, $(0, 3, -2, 0)$, $(1, 0, 0, -1)$, $(3/4, 0, 1/4, 0)$, $(2/3, 1/3, 0, 0)$. Il reste $(0, 0, 1, 3)$, $(0, 1, 0, 2)$, $(3/4, 0, 1/4, 0)$, $(2/3, 1/3, 0, 0)$. Pour ces 4 sommets, les colonnes correspondantes sont respectivement

$$\begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} \begin{pmatrix} 1 & 1 \\ 0 & 1 \end{pmatrix} \begin{pmatrix} 1 & 4 \\ 1 & 0 \end{pmatrix} \begin{pmatrix} 1 & 1 \\ 4 & 1 \end{pmatrix}$$

inversibles et donc bien des sommets au sens de la propriété.

Démo de la propriété 25 :

1) Supposons que les colonnes de C correspondant à $u_i > 0$ sont libres. On a $u = \lambda v + (1 - \lambda)w$ avec $\lambda > 0$, $1 - \lambda > 0$, $u, v, w \geq 0$, alors dès qu'un v_i ou w_i est non nul, alors u_i aussi. Donc là où les u_i sont nuls, les v_i aussi. Considérons maintenant les indices des u_i non nuls les colonnes de C correspondantes sont donc libres : $Cv = d$ et $Cw = d$ donc $C(v - w) = 0$ soit $\sum_i C_i(v - w)_i = 0$ pour les cités précedemment (les autres $(v - w)_i$ sont nuls), donc ces $(v - w)_i$ sont aussi nuls. Donc $v = w$ et donc $u = v = w$. Et il s'agit bien d'un sommet.

2) Supposons que les colonnes de C correspondant à $u_i > 0$ sont liées, alors il existe des v_i non tous nuls tels que $\sum v_i C_i = 0$. Si on note, complete par 0 et que l'on note $v = (v_1, \ldots, v_N)$ alors $Cv = 0$ et $(u \pm \theta v)_i$ est nul si $u_i = 0$. En prenant θ petit, on voit que $(u \pm \theta v)_i > 0$ si $u_i > 0$. Comme $C(u \pm \theta v) = Cu \pm 0 = d$ et $u \pm \theta v \geq 0$, les $u \pm \theta$ sont dans U et $u = (u + \theta v)/2 + (u - \theta v)/2$. u n'est donc pas un sommet.

Propriété 26 *Si J admet son maximum sur U alors J atteint (aussi) ce maximum en un sommet de U.*

Démo : Supposons que J admette son maximum en $u \in U$. Notons I l'ensemble des indices i tels que $u_i > 0$. Deux possibilités : soit les C_i, pour $i \in I$, sont libres et u est un sommet et il n'y a rien à faire, soit les C_i ne sont pas libres et il s'agit de réduire I pour finir par obtenir une famille libre.

Puisque les C_i sont liés, soit la famille v_i, $i \in I$, telle que $\sum_I v_i C_i = 0$. Si l'on complete les v_i par 0 en un vecteur v de $\mathbf{R}^N$, on a $Cv = 0$. Soit $\theta \in \mathbf{R}$, on a donc $C(u + \theta v) = d$, pour que $u + \theta v \in U$, il donc faut $u + \theta v \geq 0$.

Si $v_i = 0$, alors $(u + \theta v)_i = u_i \geq 0$. Si $v_i \neq 0$ (et il y en a au moins un puisque $v \neq 0$) alors nécessairement $u_i > 0$, et il suffit que $|\theta v_i| \leq u_i$ pour avoir $(u + \theta v)_i \geq 0$. Soit donc $\theta_0 = \min\{u_i/|v_i|, v_i \neq 0\} > 0$ puisque pour les $v_i \neq 0$ on a $u_i > 0$. Ainsi pour tout $|\theta| \leq \theta_0$, on a $u + \theta v \geq 0$ et donc $u + \theta v \in U$.

Or $J(u + \theta v) = e + a(u + \theta v) = e + au + \theta av = J(u) + \theta av$ et J est maximum en u donc $J(u) \geq J(u) + \theta av$, pour tout $|\theta| \leq \theta_0$, donc $\theta av \leq 0$, on choisit d'abord $\theta_0 > 0$ puis $-\theta_0 < 0$ donc $av = 0$ donc $J(u + \theta v) = J(u)$ pour tout $|\theta| \leq \theta_0$, donc J est maximum en $u + \theta v$.

On va choisir θ pour qu'il annule au moins un des $u_i + \theta v_i$: soit i_0 tel que $\theta_0 = u_i/|v_i|$, alors $u_{i_0} + (-\theta_0 signe(v_{i_0}))v_{i_0} = u_{i_0} - u_{i_0}|v_{i_0}|/|v_{i_0}| = 0$. Donc pour $\theta = -\theta_0 signe(v_{i_0})$ on a un nouveau point où J est maximum et tel que le nouveau I a un élément de moins (i_0) que l'ancien. Et on reprend au début. (nombre fini d'étapes et il ne peut pas y avoir de colonne nulle sinon cette variable serait sans contrainte et donc probablement $supJ = \infty$).

2.4 Passage d'un sommet à un autre

2.4.1 Exemple

Voyons dans le cas de l'exemple ci-dessus ($U = \{u \in \mathbf{R}^4, u \geq 0, u_1+u_2+u_3 = 1, 4u_1+u_2+u_4 = 3\}$) comment circuler entre les 4 sommets.

- On part de $(0, 0, 1, 3)$. Puisque c'est un sommet, les colonnes 3 et 4 de C sont libres : on peut donc calculer u_3 et u_4 (variables déduites) fonction de u_1 et u_2 : $u_3 = 1 - u_1 - u - 2$ et $u_4 = 3 - 4u_1 - u_2$. Pour passer d'un sommet à un autre, on fait varier (augmenter) une des variables nulles : si $u_1 = \theta$ alors que u_2 reste nul : $u_3 = 1 - \theta$ et $u_4 = 3 - 4\theta$ mais u_3 et $u_4 \geq 0$ soient $\theta \leq 1$ et $\theta \leq 3/4$. On choisit donc $\theta = 3/4$ (on se déplace le plus loin possible pour arriver à un autre sommet) d'où $u_3 = 1/4$ et $u_4 = 0$: on est arrivé en $(3/4, 0, 1/4, 0)$. Si on avait fait varier u_2, on aurait $u_3 = 1 - u_2$ et $u_4 = 3 - u_2$ donc $\theta \leq 1$ et $\theta \leq 3$ donc $\theta = 1$ d'où $(0, 1, 0, 2)$. ($(3/4, 0, 1/4, 0) = (0, 0, 1, 3) + 3/4(1, 0, -1, -4)$ et $C_1 = C_3 + 4C_4$ et $(0, 1, 0, 2) = (0, 0, 1, 3) + 1(0, 1, -1, -1)$ et $C_2 = C_3 + C_4$)
- On part de $(3/4, 0, 1/4, 0)$. Les colonnes 1 et 3 sont libres, on peut calculer u_1 et u_3 fonctions de u_2 et u_4 : $u_1 = (3 - u_2 - u_4)/4$ et $u_3 = (1 - 3u_2 + u_4)/4$. Si $u_2 = \theta$ et $u_4 = 0$, on a $\theta \leq 3$ et $\theta \leq 1/3$, on choisit $\theta = 1/3$ et on arrive en $(2/3, 1/3, 0, 0)$. Si $u_4 = \theta$ et $u_2 = 0$, on a $\theta \leq 3$ et θ quelconque, on choisit $\theta = 3$ et on arrive en $(0, 0, 1, 3)$. ($(2/3, 1/3, 0, 0) = (3/4, 0, 1/4, 0) + 1/3(-1/4, 1, -3/4, 0)$ et $C_2 = C_1/4 + C_3/4$ et $(0, 0, 1, 3) = (3/4, 0, 1/4, 0) + 3(-1/4, 0, 1/4, 1)$ et $C_4 = (C_1 - C_3)/4 = C_1/4 - C_3/4$)

Et de même pour les autres :

$$\begin{array}{ccc} & \overset{u_1}{\rightleftarrows} & \\ (0,0,1,3) & \underset{u_4}{} & (3/4,0,1/4,0) \\ u_2 \downarrow\uparrow u_3 & & u_2 \downarrow\uparrow u_3 \\ & \overset{u_1}{\rightleftarrows} & \\ (0,1,0,2) & \underset{u_4}{} & (2/3,1/3,0,0) \end{array}$$

Version que l'on programmera : Si on laisse les contraintes sous forme d'équation : Considérons le cas du passage de $(3/4, 0, 1/4, 0)$ vers $(0, 0, 1, 3)$. On a $4u_1 + u_2 + u_4 = 3$ et $3u_2 + 4u_3 - u_4 = 1$. On choisit de faire croître $u_4 = \theta$: il y a une contrainte uniquement si cela réduit une des variables déduites, u_1 et u_3 (valant à ce moment 3/4 et 1/4), dans la première équation les coefficients de u_1 et u_4 sont de même signe, il y a une contrainte ($\theta \leq 3$) dans la seconde équation les coefficients de u_3 et u_4 sont de signe contraire, il n'y a pas de contrainte (on peut prendre $u_4 = \theta$ et $u_3 = 1/4 + \theta/4$ pour tout $\theta \geq 0$). Puisqu'on arrive à $\theta = 3$ et $u = (0, 0, 1, 3)$ qui est un sommet, C_3 et C_4 sont libres : il s'agit de transformer les équations pour aisément calculer u_3 et u_4 fonction de u_1 et

u_2. La premiere équation donne déjà u_4 fonction de u_1 et u_2. Dans la seconde, il s'agit de faire disparaître u_4 : on ajoute la première équation à la deuxième : $4u_1 + 4u_2 + 4u_3 = 4$. En résumé : on utilise l'équation donnant la "nouvelle" variable non nulle pour la faire disparaître dans les autres équations.

```
clear

N=4;
nbcontraintes=2;

c=[1,1,1,0;4,1,0,1];
d=[1;3];

deduite(1)=3;
deduite(2)=4;

u=zeros(N,1);

for i=1:nbcontraintes
  u(deduite(i))=d(i)/c(i,deduite(i));
end
u

arete =2;

thetaz=-1;

for i=1:nbcontraintes
 if (c(i,arete)*c(i,deduite(i))>0)
    theta = u(deduite(i))*c(i,deduite(i))/c(i,arete);
    if (thetaz==-1 || theta<thetaz)
      thetaz=theta;
      ligne =i;
    end
  end
end

u(arete)=thetaz;
for i=1:nbcontraintes
  u(deduite(i))=(d(i)-c(i,arete)*u(arete))/c(i,deduite(i));
end
u

deduite(ligne)=arete;

for i=1:nbcontraintes
  if (i~=ligne)
    coef= c(i,arete)/c(ligne,arete);
    c(i,:)=c(i,:)-coef * c(ligne,:);
    d(i)=d(i)-coef*d(ligne);
  end
end
```

2.4.2 Cas général

Il y a énormément de cas particuliers. Voir le Ciarlet pour tous les détails manquant.

Hypothèses

L'ensemble U contient des vecteurs satisfaisant $Cu = d$. Comme $u \in \mathbf{R}^N$, C a N colonnes. Soit M, le nombre de lignes. On a $rg(C) \leq N$ et $rg(C) \leq M$ (rg est le rang, la taille de la plus grande matrice carrée incluse dans C et inversible). Si $rg(C) = N$, il y a au plus une solution ceci n'est pas intéressant ici (voir cours d'analyse numerique matricielle). Si $rg(C) < N$, il peut y avoir plusieurs solutions et c'est le cas qui nous intéresse. Si de plus $rg(C) < M$ cela signifie qu'il y a des lignes liées en trop : soit les seconds membres sont compatibles et on peut les enlever soit ils sont incompatibles et il n'y a pas de solutions.

Ex : $u_1 + u_2 + u_3 = 1$, $4u_1 + u_2 + u_4 = 3$ et $3u_1 - u3 + u_4 = 5$ ou $= 2$. Une contrainte "en trop". Le rang est 2.

On peut donc se restreindre à : $M = rg(C) < N$. Si on résoud $Cv = 0$ on voit que l'on a donc M vraies inconnues et $N - M$ inconnues paramètres. On se place tout le temps dans ce cas (ou on s'y ramene).

Dans l'exemple précédent : $N = 4$, $M = 2$, $rg(C) = 2$. On peut calculer 2 inconnues fonction de 2 autres.

Méthode

On part d'un sommet u que l'on peut donc voir comme un $u \in U$ dont les coordonnées sont nulles en dehors de $\{i\} \cup I$ (i sera fixé par la suite) et tel que les C_k pour $k \in \{i\} \cup I$ sont libres. On peut supposer $card(\{i\} \cup I) = rg(C) = M$ en completant la famille des colonnes (s'il l'on ne pouvait pas completer, cela voudrait dire que l'on a une famille libre maximale donc une base : impossible puisqu'on se place dans le cas $card(\{i\} \cup I) < rg(C)$). Il est donc possible que des u_k soient nuls pour $k \in \{i\} \cup I$.

Ex : $u_1 + u_2 + u_3 = 1$, $4u_1 + u_2 + u_4 = 0$ et le sommet $(0, 0, 1, 0)$. On a C_3 que l'on peut completer par exemple par C_4 (ou C_1 ou C_2).

On va arriver (sauf dans un cas) à un point $u + \theta v$ satisfaisant la même chose avec $I \cup \{j\}$ et $j \neq i$ et qui sera un sommet.

Puisque les C_k pour $k \in \{i\} \cup I$ sont M vecteurs libres dans $\mathbf{R}^M$, ils forment une base aussi pour j, il existe α_k tel que $C_j = \sum_{\{i\} \cup I} \alpha_k C_k$. En choisissant $v_j = 1$, $v_k = -\alpha_k$ et en completant par 0, on alors $v \in \mathbf{R}^N$, $Cv = 0$. On considère alors $u + \theta v$ avec $\theta \geq 0$. 2 cas :

- Si $v_k \geq 0$ pour tout $k \in \{i\} \cup I$, il est possible de choisir n'importe quel $\theta \geq 0$, il est possible que $\sup J = \infty$.
- Si il y a au moins un $v_k < 0$ pour $k \in \{i\} \cup I$, il existe $\theta \geq 0$ tel que $u + \theta v \geq 0$ et au moins un des $k \in \{i\} \cup I$ est tel que $(u + \theta v)_k = 0$, on l'appelle i.

On oublie le premier cas.

Propriété 27 *Le nouveau point $u + \theta v$ est aussi un sommet de U.*

Démo : On considère donc maintenant les C_k pour $k \in I \cup \{j\}$. Les coordonnées de $u + \theta v$ sont nulles en dehors de $I \cup \{j\}$: $(u + \theta v)_i = 0$ et sont restées nulles pour les autres indices. Montrons que les C_k pour $k \in I \cup \{j\}$ sont libres, en supposant, par l'absurde, qu'ils ne le soient pas : il existe des β_k tels que $\sum_{I \cup \{j\}} \beta_k C_k = 0$ avec $\beta_j \neq 0$ (sinon les C_k pour $k \in I$ ne sont pas libres en contradiction avec le début) donc $C_j = \sum_I \gamma_k C_k$. Mais on a vu que $C_j = \sum_{\{i\} \cup I} \alpha_k C_k$ avec $\alpha_i = -v_i > 0$. Il y a donc 2 écritures dans la mème famille libre (en faisant la soustraction, on voit qu'on a une famille liée). Contradiction.

Lorsqu'on se déplace de $(0, 0, 1, 3)$ vers $(3/4, 0, 1/4, 0)$, on passe de u à $u + \theta v$ avec $u = (0, 0, 1, 3)$, $\theta = 3/4$, et $v = (1, 0, -1, -4)$. Comme les points sont dans U, on a $Cu = d$ et $C(u + \theta v) = d$ donc $Cv = 0$, c'est à dire, si C_i est la i^{eme} colonne de C, $C_1 = C_3 + C_4$. Supposons que C_1 et C_3 ne soient pas libres : $C_1 = \alpha C_3$ alors C_1 sécrit de 2 façon différentes dans une famille libre : contradition.

2.4.3 Un exemple complet en exercice

On fabrique 3 produits valant 3, 2 et 1 euros sous les contraintes (ouvriers ou matière première) $u_1 + u_2 + u_3 \leq 1$ et $4u_1 + u_2 + u_3 \leq 3$. Maximiser le gain.

$J(u) = 3u_1 + 2u_2 + u_3$ et on ajoute les variables u_4 et u_5 pour transformer les inégalités en égalités : $U = \{u \in \mathbf{R}^5, u \geq 0, u_1+u_2+u_3+u_4 = 1, 4u_1+u_2+u_3+u_5 = 3\}$. On part de $(0,0,0,1,3)$ (ie pas de production) et l'on vérifie que c'est bien un sommet. Les variables déduites sont u_4 et u_5.

Si l'on choisit les arêtes en prenant les plus grands coefficients dans J,

1) on choisit donc l'arête u_1 et on arrive en $(3/4, 0, 0, 1/4, 0)$ alors $J = 9/4$. Les variables déduites sont maintenant u_1 et u_4. Il faut donc faire disparaitre u_1 de J : $J(u) = 3u_1 + 2u_2 + u_3 = 3((3 - u_2 - u_3 - u_5)/4) + 2u_2 + u_3 = 9/4 + 5/4u_2 + 1/4u_3 - 3/4u_5$.

2) on choisit alors l'arête u_2, on arrive en $(2/3, 1/3, 0, , 0)$ et on "voit" que c'est fini.... à finir.

Il pourrait y avoir $C_5^3 = C_5^2 = 10$ sommets mais il n'y en a que 6 : $(0,0,0,1,3)$ $(J = 0)$, $(0,1,0,0,2)$ $(J = 2)$, $(0,0,1,0,2)$ $(J = 1)$, $(2/3, 1/3, 0, 0, 0)$ $(J = 8/3)$, $(2/3, 0, 1/3, 0, 0)$ $(J = 7/3)$ et $(3/4, 0, 0, 1/4, 0)$ $(J = 9/4)$. On peut tourner entre les 3 premiers et entre les 3 derniers et passer de $(0,0,0,1,3)$ à $(3/4, 0, 0, 1/4, 0)$, de $(0,1,0,0,2)$ à $(2/3, 1/3, 0, 0, 0)$ et de $(0,0,1,0,2)$ à $(2/3, 0, 1/3, 0, 0)$. U est donc un objet de dimension 3 dans un espace de dim 5 qui est un "demi carré" ; 2 faces triangulaires et 3 carrées. (dessin).

2.4.4 L'algorithme

On a $J(u) = au + e$. On suppose connu un u sommet de U.

On peut donc supposer que les coordonnées de u sont nulles en dehors de K, que les C_k pour $k \in K$ sont libres et que $card(K) = rg(C) = M$. Soit $w \in U$, Comme $Cw = d$, on peut donc calculer w_k pour $k \in K$ fonction des w_k pour $k \notin K$, alors $J(w) = \sum_{k \notin K} \alpha_k w_k + f$. Et si v est positif et tel que $Cv = 0$ alors pour $\theta \geq 0$, $u + \theta v \in U$ et $J(u + \theta v) = J(u) + \theta \sum_{k \notin K} \alpha_k v_k$. Donc on peut avoir $J(u + \theta v) > J(u)$ uniquement si au moins un $\alpha_k > 0$ (parmi les coordonnées non déduites de u, il n'y a qu'une coordonnée non nulle de v qui vaut 1).

Ainsi si $\alpha_k \leq 0$ pour tout $k \notin K$, u est le point où J est maximum. L'algorithme est terminé.

S'il existe au moins un $\alpha_k > 0$ pour $k \notin K$, on applique la méthode de changement de sommet en prenant $j = k$: soit on est dans le cas où on peut prendre $\theta \geq 0$ quelconque, et on on peut montrer que $\sup J = \infty$, on s'arrête, soit on est dans l'autre cas et on se déplace vers un autre sommet avec $J(u + \theta v) \geq J(u)$ (J n'a pas diminué) puis on recommence.

2.5 Le programme complet

On reprend notre second exemple :

Maximiser

$$J(u_1, u_2) = 2u_1 + u_2$$

sur $u_1 \geq 0$, $u_2 \geq 0$ tels que $u_1 + u_2 \leq 1$ et $4u_1 + u_2 \leq 3$.

que l'on peut interpreter comme le choix de production entre deux produit de façon à maximiser le gain J sous deux contraintes.

On commence par transformer le problème en un problème équivalent avec condition "égalité" :

Maximiser

$$J(u_1, u_2) = 2u_1 + u_2$$

sur $u_1 \geq 0$, $u_2 \geq 0$, $u_3 \geq 0$, $u_4 \geq 0$ tels que $u_1 + u_2 + u_3 = 1$ et $4u_1 + u_2 + u_4 = 3$.

On "sait" que les extrema sont atteints sur les sommets. Partons du sommet "trivial" : $(0,0,1,3)$ (cas où on ne produit rien et ou on ne gagne rien on est donc loin de la "bonne" solution). Il se traduit par $(0,0)$ dans les variables d'origine.

On a vu que C_3 et C_4 sont libres. Il faut donc écrire J fonction des autres variables : u_1 et u_2, c'est déjà le cas : $J(u) = 2u_1 + u_2$. Les deux coefficients sont > 0 le max n'est pas atteint et on peut changer de sommet avec $j = 1$ ou $j = 2$.

Si on fait augmenter u_2, on se déplace vers le sommet $(0,1,0,2)$. On a alors $J = 2 \times 0 + 1 = 1$, le gain a bien augmenté.

$(0,1,0,2)$ est bien un sommet car C_2 et C_4 sont libres. On exprime donc J fonction des autres variables u_1 et u_3 : on calcule u_2 et u_4 fonction de u_1 et u_3 puis on remplace. On a $u_2 = 1 - u_1 - u_3$ et $u_4 = 3 - 4u_1 - u_2 = 2 - 3u_1 + u_3$ donc $J = 2u_1 + u_2 = 2u_1 + 1 - u_1 - u_3 = 1 + u_1 - u_3$.

La situation s'améliore au sens où il n'y a plus qu'une direction pouvant faire augmenter J : faire augmenter u_1 (faire augmenter u_3 ferait diminuer J).

On se déplace donc de $(0,1,0,2)$ vers $(2/3,1/3,0,0)$, on a alors $J = 2 \times 2/3 + 1/3 = 5/3 > 1$. Le gain a bien augmenté.

$(2/3,1/3,0,0)$ est bien un sommet car C_1 et C_2 sont libres. On exprime donc J fonction des autres variables u_3 et u_4 : on calcule u_1 et u_2 fonction de u_3 et u_4 puis on remplace. On a $u_1 = (2 + u_3 - u_4)/3$ et $u_2 = (1 - 4u_3 + u_4)/3$ donc $J = 2u_1 + u_2 = (5 - 2u_3 - u_4)/3$.

Cette fois, les deux coefficients sont < 0, J ne peut plus que diminuer. Le maximum est donc atteint et vaut : $5/3$ atteint en $u_1 = 2/3$ et $u_2 = 1/3$.

Lors de la programmation on fera les resolutions de système par pivot de Gauss.

On reprend le choix de l'arête du programme à une contrainte et le changement de sommet vu précedemment.

```
clear

% inconnues positives

% contraintes cu=d

% u_1 + u_2 + u_3 = 1
c(1,1)=1;
c(1,2)=1;
c(1,3)=1;
c(1,4)=0;

d(1,1)= 1;
% 4 u_1 + u_2 + u_4 = 3
c(2,1)=4;
c(2,2)=1;
c(2,3)=0;
c(2,4)=1;

d(2,1)= 3;


% J=au+e
% max j(v)= 0 + 2 u_1 + u_2 + 0 u_3 + 0 u_4
a(1,1)=2;
a(1,2)=1;
a(1,3)=0;
a(1,4)=0;
a

e=0;

% il faut distinguer les inconnues relles des inconnues deduites
deduite(1)=3;
deduite(2)=4;

% et ainsi avoir comme sommet vident 0 0 1 3
u(1,1)=0;
u(2,1)=0;
u(3,1)=1;
u(4,1)=3;

J=e+a*u
```

```
% nombre d'inconnues
n=4;
% nombre de contraintes et rang de c
m=2;

atteint=0;

while (atteint==0)
  % si au moins un a est >0, le max n'est pas atteint
  atteint=1;
  for i=1:n
    if (a(1,i)>0)
      atteint=0;
      arete=i;
    end
  end

  % si le min n'est pas atteint on se deplace sur une des aretes
  % u(arete) va augmenter de theta et les u(deduite(i)) diminuer et l'un
  % d'eux atteindra 0
  if (atteint==0)
    theta=-1;
    for i=1:m
      if (theta<0)
        if ((c(i,arete)~=0)&(c(i,deduite(i))*c(i,arete)>0)
          theta=d(i,1)/c(i,arete)
          ligne=i
        end
      else
        if ((c(i,arete)~=0)&(c(i,deduite(i))*c(i,arete)>0)&(d(i,1)/c(i,arete)<theta)
          theta=d(i,1)/c(i,arete)
          ligne=i
        end
      end
    end
    % la longueur maximale du deplacement est theta

    % on se deplace dans la direction arete en ne modifiant que les
    % variables deduites
    u(arete,1)=theta;
    for i=1:m
      u(deduite(i),1)=u(deduite(i),1)-theta*c(i,arete)/c(i,deduite(i));
    end

    % on echange les roles de deduite(ligne) et arete
    % J=e+\sum a_j u_j et
    % u_arete = (d(ligne)-\sum j\not=arete c_ligne,j u_j)/c_ligne,arete
    % J=e+ a_arete d/c_arete + \sum j (a_j-a_arete c_j/c_arete) u_j
    coef= a(1,arete)/c(ligne,arete);
    e = e + coef * d(ligne,1) ;
    a(1,:) = a(1,:) - coef * c(ligne,:);

    % \sum c_i,j u_j = d_i
    % u_arete = (d(ligne)-\sum j\not=arete c_ligne,j u_j)/c_ligne,arete
    % \sum (c_i,j - c_i,arete c_ligne,j/c_ligne,arete) u_j
    % = d_i -c_i,arete d(ligne)/c_ligne,arete
```

```
    for i=1:m
      if (i~=ligne)
        coef = c(i,arete) / c(ligne,arete);
        c(i,:)=c(i,:) - coef * c(ligne,:);
        d(i,1)=d(i,1) - coef * d(ligne,1);
      end
    end

    deduite(ligne)=arete;

    J=e+a*u
  end % if non atteint
end % while non atteint
```

2.6 Un petit sudoku

On souhaite résoudre un sudoku "réduit" grâce à la méthode du simplexe :

$$\begin{array}{cccc} 1 & ? & ? & ? \\ 2 & ? & ? & 3 \\ 4 & ? & ? & 2 \\ ? & ? & ? & 1 \end{array}$$

Commençons par noter cela avec une matrice à trois dimensions : premièr indice la ligne, deuxième la colonne, troisième la valeur. u est nul sauf $u(1,1,1) = 1$, $u(2,1,2) = 1$, $u(2,4,3) = 1$, ... soit 6 contraintes. A cela s'ajoute celles du sudoku

- Dans une case, il n'y a qu'un chiffre : $\sum_k u(i,j,k) = 1$ pour tous i et j. Soit 16 contraintes.
- Dans chaque ligne, chaque chiffre apparait 1 fois : $\sum_j u(i,j,k) = 1$ pour tous i et k. Soit 16 contraintes.
- Dans chaque colonne, chaque chiffre apparait 1 fois : $\sum_i u(i,j,k) = 1$ pour tous j et k. Soit 16 contraintes.
- Dans chaque carré de 4 cases, chaque chiffre apparait 1 fois : $\sum_{i=I,I+1} \sum_{j=J,J+1} u(i,j,k)$ pour tout $I, J = 1$ ou 3 et pour tout k. Soit 16 contraintes.

Il y a donc 64 contraintes habituelles et 6 contraintes dues aux chiffres déjà connus. Soit plus que déquations. Cela ne correspond pas aux hypothèses effectuées précedemment. Il n'y a pas de maximum à trouver : $J(u) = 0$. Enfin, trouver un sommet dans U c'est connaître la solution.... Il faut donc changer de problème :

Au lieu de chercher $u \leq 0$ satisfaisant $Cu = d$, on cherche (u, v) satisfaisant $u \leq 0$, $v \leq 0$, $Cu + v = d$ et maximisant $J(u, v) = -\sum v$. U non vide est équivalent à $\max J = 0$ (si U contient u alors $J(u, 0) = 0$ donc $\max J = 0$, si $\max J = 0$, il existe u et v tels que $J(u, v) = 0$ alors $v = 0$ donc U contient u). Si l'on trouve $\max J = 0$, U est non vide : le sudoku est résolu. Si $\max J < 0$, U est vide, le sudoku n'est pas résoluble.

Et ici il est facile de partir d'un sommet : $(0, d)$ (on a $d \geq 0$ en changeant les signes dans les équations de $Cu = d$) qui donne $J(d) = -\sum d$. Il s'agit bien d'un sommet car les nouvelles contraintes sont $(C|Id)(\frac{u}{v}) = d$ et la nouvelle matrice des contrainte est donc $(C|Id)$, les colonnes correspondant à v sont libres puisque elles forment la matrice Id.

Comme il y a $64+6$ contraintes et au départ 64 inconnues, on arrive à un système de $64+6 = 70$ contraintes mais $64 + 64 + 6 = 134$ inconnues. Il y a bien plus d'inconnues que de contraintes et le rang de la matrice des contraintes est $64 + 6$ (puisqu'elle contient Id).

On choisit pour commencer de considérer comme variables déduites les 70 dernières et il faut donc commencer par ecrire J fonction des 64 premières. Le reste est identique au code précedent.

On voit les matrices suivantes au cours de la résolution

$$\begin{pmatrix} ? & ? & ? & ? \\ ? & ? & ? & ? \\ ? & ? & ? & ? \\ ? & ? & ? & ? \end{pmatrix} \begin{pmatrix} ? & ? & ? & ? \\ ? & ? & ? & ? \\ ? & ? & ? & ? \\ ? & ? & ? & 4 \end{pmatrix} \begin{pmatrix} ? & ? & ? & ? \\ ? & ? & ? & ? \\ ? & ? & ? & 4 \\ ? & 4 & ? & ? \end{pmatrix} \begin{pmatrix} ? & ? & ? & ? \\ ? & ? & ? & ? \\ 4 & ? & ? & ? \\ ? & ? & ? & 4 \end{pmatrix} \begin{pmatrix} ? & ? & ? & ? \\ ? & ? & ? & 4 \\ 4 & ? & ? & ? \\ ? & ? & 4 & ? \end{pmatrix} \cdots$$

$$\ldots \begin{pmatrix} ? & 4 & 2 & 3 \\ 2 & 3 & 4 & 1 \\ 4 & 1 & 3 & 2 \\ 3 & 2 & 1 & 4 \end{pmatrix} \begin{pmatrix} ? & 4 & 2 & 3 \\ 2 & 3 & 1 & 4 \\ 4 & 1 & 3 & 2 \\ 3 & 2 & 4 & 1 \end{pmatrix} \begin{pmatrix} ? & 3 & 2 & 4 \\ 2 & 4 & 1 & 3 \\ 4 & 1 & 3 & 2 \\ 3 & 2 & 4 & 1 \end{pmatrix} \begin{pmatrix} 3 & 4 & 2 & 1 \\ 2 & 1 & 4 & 3 \\ 4 & 3 & 1 & 2 \\ 1 & 2 & 3 & 4 \end{pmatrix} \begin{pmatrix} 1 & 3 & 2 & 4 \\ 2 & 4 & 1 & 3 \\ 4 & 1 & 3 & 2 \\ 3 & 2 & 4 & 1 \end{pmatrix}$$

On voit un saut étonnant à l'avant dernière matrice (qui est cohérent mais à 2 contraintes de chiffres non vérifiées). Le code est le premier ci-dessous.

2.6.1 Premier code

```
clear

N=2;

% J=0
a=zeros(1,(N^2)^3);
e=0;

l=0;
%x(1,1,1)=1;
%x(2,1,2)=1;
%x(2,4,3)=1;
%x(3,1,4)=1;
%x(3,4,2)=1;
%x(4,4,1)=1;
uc=zeros(N^2,N^2);
uc(1,:)=[1,0,0,0];
uc(2,:)=[2,0,0,3];
uc(3,:)=[4,0,0,2];
uc(4,:)=[0,0,0,1];
uc

for i=1:(N^2)
  for j=1:(N^2)
    for k=1:(N^2)
      if (uc(i,j)==k)
        l=l+1;
        m=(k-1)*((N^2)^2)+(i-1)*(N^2)+j;
        c(l,m)=1;
        d(l,1)=1;
      end
    end
  end
end


% un chiffre par case sum(x(1,1,:))
for i=1:N^2;
  for j=1:N^2;
    l=l+1;
    for k=1:N^2;
      m=(k-1)*((N^2)^2)+(i-1)*(N^2)+j;
      c(l,m)=1;
    end
    d(l,1)=1;
  end
end
```

```
%lignes sum(x(1,:,1))
for i=1:N^2;
  for k=1:N^2;
    l=l+1;
    for j=1:N^2;
      m=(k-1)*((N^2)^2)+(i-1)*(N^2)+j;
      c(l,m)=1;
    end
    d(l,1)=1;
  end
end

%colonnes sum(x(:,1,1))
for j=1:N^2
  for k=1:N^2
    l=l+1;
    for i=1:N^2
      m=(k-1)*((N^2)^2)+(i-1)*(N^2)+j;
      c(l,m)=1;
    end
    d(l,1)=1;
  end
end

%blocs
for n=1:N
  for p=1:N
    for k=1:N^2
      l=l+1;
      for i=(n-1)*N+1:(n-1)*N+N
        for j=(p-1)*N+1:(p-1)*N+N
          m=(k-1)*((N^2)^2)+(i-1)*(N^2)+j;
          c(l,m)=1;
        end
      end
      d(l,1)=1;
    end
  end
end

nbcontraintes=l
nbinconnues=(N^2)^3

%il n'y a pas de sommet evident de U pour lancer les calculs.
%on change legerement de probleme :
%Cu=d devient Cu+v=d et la nouvelle inconnue est (u,v) et J=-\sum v le
%but est d'arriver a 0. le sommet initial est u=0 v=d
%on avait C 70x64 et d 70x1 et u 64x1
%on aura u 134x1 C 70x134 d 70x1 a 1x134 e 1x1
u(1:nbinconnues,1)=zeros(nbinconnues,1);
u(nbinconnues+1:nbinconnues+nbcontraintes)=d;
c(1:nbcontraintes,nbinconnues+1:nbinconnues+nbcontraintes)=diag(ones(1,nbcontraintes));
e=0;
a(1,nbinconnues+1:nbinconnues+nbcontraintes)=-ones(1,nbcontraintes);
```

```
J=a*u+e

deduite(1:nbcontraintes)=nbinconnues+1:nbinconnues+nbcontraintes;

%J depend des variables deduites, il faut faire les changements de
%variables
% J=e+\sum a_j u_j, on veut supprimer la dependance en les variables
% deduite : a_j=0 pour j>=65
% u_deduite(ligne)
% = (d_ligne-\sum j\not=deduite c_ligne,j u_j)/c_ligne,deduite(ligne)
% J=e+ a_deduite(ligne) d_ligne/c_ligne,deduite(ligne)
% + \sum j (a_j-a_deduite(ligne) c_ligne,j/c_ligne,deduite(ligne)) u_j

for ligne=1:nbcontraintes
  coef= a(1,deduite(ligne))/c(ligne,deduite(ligne));
  e = e + coef * d(ligne) ;
  a(1,:) = a(1,:) - coef * c(ligne,:);
end

atteint=0;

while (atteint==0)
  % si au moins un a est >0, le max n'est pas atteint
  atteint=1;
  for i=1:nbinconnues+nbcontraintes
    if (a(1,i)>0)
      atteint=0;
      arete=i;
    end
  end
atteint
arete

  % si le min n'est pas atteint on se dplace sur une des artes
  % u(arete) va augmenter de theta et les u(deduite(i)) diminuer et l'un
  % d'eux atteindra 0
  if (atteint==0)
    theta=-1;
    for i=1:nbcontraintes
      if (theta<0)
        if (c(i,arete)~=0)
          if (c(i,deduite(i))*c(i,arete)>0)
            theta=d(i,1)/c(i,arete);
            ligne=i;
          end
        end
      else
        if (c(i,arete)~=0)
          if ((c(i,deduite(i))*c(i,arete)>0)&(d(i,1)/c(i,arete)<theta))
            theta=d(i,1)/c(i,arete);
            ligne=i;
          end
        end
      end
    end
```

```
    % la longueur maximale du dplacement est theta
    % il faudrait traiter theta <0 qui signifie J infini
theta
ligne
deduite(ligne)

    % on se dplace dans la direction arete en ne modifiant que les
    % variables techniques
    u(arete,1)=theta;
    for i=1:nbcontraintes
      u(deduite(i),1)=u(deduite(i),1)-theta*c(i,arete)/c(i,deduite(i));
    end

    % on affiche le resultat sous forme d'un carre
    uc=zeros(N^2,N^2);
    for i=1:N^2
      for j=1:N^2
        for k=1:N^2
          uc(i,j)+=k*u((k-1)*((N^2)^2)+(i-1)*(N^2)+j);
        end
      end
    end
uc

    % on echange les roles de deduite(ligne) et arete
    % J=e+\sum a_j u_j et
    % u_arete = (d(ligne)-\sum j\not=arete c_ligne,j u_j)/c_ligne,arete
    % J=e+ a_arete d/c_arete + \sum j (a_j-a_arete c_j/c_arete) u_j
    coef= a(1,arete)/c(ligne,arete);
    e = e + coef * d(ligne,1) ;
    a(1,:) = a(1,:) - coef * c(ligne,:);
J=a*u+e

    % \sum c_i,j u_j = d_i
    % u_arete = (d(ligne)-\sum j\not=arete c_ligne,j u_j)/c_ligne,arete
    % \sum (c_i,j - c_i,arete c_ligne,j/c_ligne,arete) u_j
    % = d_i -c_i,arete d(ligne)/c_ligne,arete
    for i=1:nbcontraintes
      if (i~=ligne)
        coef = c(i,arete) / c(ligne,arete);
        c(i,:)=c(i,:) - coef * c(ligne,:);
        d(i,1)=d(i,1) - coef * d(ligne,1);
      end
    end

    deduite(ligne)=arete;

  end % if non atteint

end % while non atteint
```

2.6.2 Deuxième code plus robuste

Il y a des problèmes avec des valeurs nulles qui ne le deviennent plus par arrondi.
Les contraintes particulieres ont été placées après les conditions générales.

Environ 371 itérations (donc 32 réelles faisant augmenter J).
Les contraintes sont omises dans le code ci-dessous.

```
clear

N=2;

l=0;
............
nbcontraintes=l
nbinconnues=(N^2)^3

temps=time();
%il n'y a pas de sommet evident de U pour lancer les calculs.
%on change legerement de probleme :
%Cu=d devient Cu+v=d et la nouvelle inconnue est (u,v) et J=-\sum v le
%but est d'arriver a 0. le sommet initial est u=0 v=d
%on avait C 70x64 et d 70x1 et u 64x1
%on aura u 134x1 C 70x134 d 70x1 a 1x134 e 1x1
u(1:nbinconnues,1)=zeros(nbinconnues,1);
u(nbinconnues+1:nbinconnues+nbcontraintes)=d(1:nbcontraintes);
c(1:nbcontraintes,nbinconnues+1:nbinconnues+nbcontraintes)=diag(ones(1,nbcontraintes));
e=0;
a(1,1:nbinconnues)=zeros(1,nbinconnues);
a(1,nbinconnues+1:nbinconnues+nbcontraintes)=-ones(1,nbcontraintes);

J=a*u+e

deduite(1:nbcontraintes)=nbinconnues+1:nbinconnues+nbcontraintes;

%J depend des variables deduites, il faut faire les changements de
%variables
% J=e+\sum a_j u_j, on veut supprimer la dependance en les variables
% deduite : a_j=0 pour j>=65
% u_deduite(ligne)
% = (d_ligne-\sum j\not=deduite c_ligne,j u_j)/c_ligne,deduite(ligne)
% J=e+ a_deduite(ligne) d_ligne/c_ligne,deduite(ligne)
% + \sum j (a_j-a_deduite(ligne) c_ligne,j/c_ligne,deduite(ligne)) u_j

for ligne=1:nbcontraintes
  coef= a(1,deduite(ligne))/c(ligne,deduite(ligne));
  e = e + coef * d(ligne) ;
  a(1,:) = a(1,:) - coef * c(ligne,:);
end

atteint=0;
compteur=0;
while (atteint==0)
  compteur+=1
  % si au moins un a est >0, le max n'est pas atteint
  atteint=1;
  for i=1:nbinconnues+nbcontraintes
    %if (a(1,i)>0)
    if (a(1,i)>1e-10)
      atteint=0;
      arete=i;
```

```
    end
  end
atteint
arete
coefa=a(1,arete)

  % si le min n'est pas atteint on se dplace sur une des artes
  % u(arete) va augmenter de theta et les u(deduite(i)) diminuer et l'un
  % d'eux atteindra 0
  if (atteint==0)
    theta=-1;
    for i=1:nbcontraintes
      if (theta<0)
        %if (c(i,arete)~=0)
        if (abs(c(i,arete))>10e-10)
          if (c(i,deduite(i))*c(i,arete)>0)
            theta=d(i,1)/c(i,arete);
            ligne=i;
          end
        end
      else
        %if (c(i,arete)~=0)
        if (abs(c(i,arete))>10e-10)
          if ((c(i,deduite(i))*c(i,arete)>0)&(d(i,1)/c(i,arete)<theta))
            theta=d(i,1)/c(i,arete);
            ligne=i;
          end
        end
      end
    end
    % la longueur maximale du dplacement est theta
    % il faudrait traiter theta <0 qui signifie J infini
theta
ligne
deduite(ligne)

    % on se dplace dans la direction arete en ne modifiant que les
    % variables deduites
    u(arete,1)=theta;
    for i=1:nbcontraintes
      u(deduite(i),1)=u(deduite(i),1)-theta*c(i,arete)/c(i,deduite(i));
    end
    % pour eviter les pb d'arrondi, on force
    u(deduite(ligne))=0;

    % on affiche le resultat sous forme d'un carre
    uc=zeros(N^2,N^2);
    for i=1:N^2
      for j=1:N^2
        for k=1:N^2
          uc(i,j)+=k*u((k-1)*((N^2)^2)+(i-1)*(N^2)+j);
        end
      end
    end
uc
```

```
    % on echange les roles de deduite(ligne) et arete
    % J=e+\sum a_j u_j et
    % u_arete = (d(ligne)-\sum j\not=arete c_ligne,j u_j)/c_ligne,arete
    % J=e+ a_arete d/c_arete + \sum j (a_j-a_arete c_j/c_arete) u_j
    coef= a(1,arete)/c(ligne,arete);
    e = e + coef * d(ligne,1) ;
    a(1,:) = a(1,:) - coef * c(ligne,:);
    % pour eviter les pb d'arrondi, on force
    a(1,arete)=0;
J=a*u+e

    % \sum c_i,j u_j = d_i
    % u_arete = (d(ligne)-\sum j\not=arete c_ligne,j u_j)/c_ligne,arete
    % \sum (c_i,j - c_i,arete c_ligne,j/c_ligne,arete) u_j
    % = d_i -c_i,arete d(ligne)/c_ligne,arete
    for i=1:nbcontraintes
      if (i~=ligne)
        coef = c(i,arete) / c(ligne,arete);
        c(i,:)=c(i,:) - coef * c(ligne,:);
        d(i,1)=d(i,1) - coef * d(ligne,1);
        % pour eviter les pb d'arrondi, on force
        c(i,arete)=0;
      end
    end

    deduite(ligne)=arete;

  end % if non atteint

end % while non atteint
duree=time()-temps
```

2.6.3 Troisième code avec pivot de Gauss

En fait la matrice est de rang 40 pour 64, on applique le pivot de Gauss pour supprimer les lignes "doublons". 54 itérations (dont 18 réelles faisant augmenter J) et $6,5$ fois plus rapide que le précedent.

On ne présente que le code ajouté au précedent (après l'énoncé des contraintes).

Comme les contraintes ne sont plus exprimées comme les contraintes initiales, la relaxation due à l'ajout des variables supplémentaires conduit à des situations "bizarres".

J valant successivement
-42, -29, -28, -17, -16,
-15, -14, -13, -12, -11,
-10, -9, -8, -7,
-6, -5, -2, -1, 0

$$\begin{pmatrix} ? & ? & ? & ? \\ ? & ? & ? & ? \\ ? & ? & ? & ? \\ ? & ? & ? & ? \end{pmatrix} \begin{pmatrix} ? & ? & ? & ? \\ ? & ? & ? & ? \\ ? & ? & ? & ? \\ ? & ? & 4 & ? \end{pmatrix} \begin{pmatrix} ? & ? & ? & ? \\ ? & ? & ? & ? \\ 4 & ? & ? & ? \\ ? & ? & 4 & ? \end{pmatrix} \begin{pmatrix} ? & ? & ? & ? \\ ? & 4 & ? & ? \\ 4 & ? & ? & ? \\ ? & ? & 4 & ? \end{pmatrix} \begin{pmatrix} ? & ? & ? & 4 \\ ? & 4 & ? & ? \\ 4 & ? & ? & ? \\ ? & ? & 4 & ? \end{pmatrix} \cdots$$

$$\cdots \begin{pmatrix} ? & ? & ? & 4 \\ ? & 4 & ? & ? \\ 4 & ? & ? & ? \\ 3 & ? & 4 & ? \end{pmatrix} \begin{pmatrix} ? & ? & ? & 4 \\ ? & 4 & ? & ? \\ 4 & ? & 3 & ? \\ 3 & ? & 4 & ? \end{pmatrix} \begin{pmatrix} ? & ? & ? & 4 \\ ? & 4 & ? & 3 \\ 4 & ? & 3 & ? \\ 3 & ? & 4 & ? \end{pmatrix} \begin{pmatrix} ? & 3 & ? & 4 \\ ? & 4 & ? & 3 \\ 4 & ? & 3 & ? \\ 3 & ? & 4 & ? \end{pmatrix} \begin{pmatrix} ? & 3 & ? & 4 \\ ? & 4 & ? & 3 \\ 4 & ? & 3 & ? \\ 2+3 & ? & 4 & ? \end{pmatrix} \cdots$$

$$\cdots \begin{pmatrix} ? & 3 & ? & 4 \\ ? & 4 & ? & 3 \\ 4 & ? & 3 & 2 \\ 2+3 & ? & 4 & ? \end{pmatrix} \begin{pmatrix} ? & 3 & ? & 4 \\ ? & 4 & ? & 3 \\ 4 & 2 & 3 & 2 \\ 2+3 & ? & 4 & ? \end{pmatrix} \begin{pmatrix} ? & 3 & ? & 4 \\ 2 & 4 & ? & 3 \\ 4 & 2 & 3 & 2 \\ 2+3 & ? & 4 & ? \end{pmatrix} \begin{pmatrix} ? & 3 & 2 & 4 \\ 2 & 4 & ? & 3 \\ 4 & 2 & 3 & 2 \\ 2+3 & ? & 4 & ? \end{pmatrix} \cdots$$

$$\cdots \begin{pmatrix} ? & 3 & 2 & 4 \\ 2 & 4 & ? & 3 \\ 4 & 2 & 3 & 2 \\ 2+3 & ? & 4 & 1 \end{pmatrix} \begin{pmatrix} ? & 3 & 2 & 4 \\ 2 & 4 & ? & 3 \\ 4 & 2 & 3 & 2 \\ 2+3 & 1 & 4 & 1 \end{pmatrix} \begin{pmatrix} ? & 3 & 2 & 4 \\ 2 & 4 & ? & 3 \\ 4 & 1 & 3 & 2 \\ 3 & 2 & 4 & 1 \end{pmatrix} \begin{pmatrix} ? & 3 & 2 & 4 \\ 2 & 4 & 1 & 3 \\ 4 & 1 & 3 & 2 \\ 3 & 2 & 4 & 1 \end{pmatrix} \begin{pmatrix} 1 & 3 & 2 & 4 \\ 2 & 4 & 1 & 3 \\ 4 & 1 & 3 & 2 \\ 3 & 2 & 4 & 1 \end{pmatrix}$$

```
ligne=1;
for colonne=1:nbinconnues
  probleme=0;
  if (c(ligne,colonne)==0)
    j=ligne+1;
    while((j<=nbcontraintes)&&(c(j,colonne)==0))
      j+=1;
    end
    if (j==nbcontraintes+1)
      probleme=1;
    else
      cligne=c(ligne,:);
      c(ligne,:)=c(j,:);
      c(j,:)=cligne;
      dligne=d(ligne,1);
      d(ligne,1)=d(j,1);
      d(j,1)=dligne;
    end
  end
  if (probleme==0)
    for j=1:nbcontraintes
      if (j~=ligne)
        coef=c(j,colonne)/c(ligne,colonne);
        c(j,:)=c(j,:)-coef*c(ligne,:);
        d(j,1)=d(j,1)-coef*d(ligne,1);
      end
    end
    ligne +=1;
  end
end
rank(c)
nbcontraintes=ligne-1

%on a besoin de d>=0 pour la suite
for ligne=1:nbcontraintes
  if (d(ligne,1)<0)
    c(ligne,:)=-c(ligne,:);
    d(ligne,1)=-d(ligne,1);
  end
end


%sans gauss
%on avait C 70x64 et d 70x1 et u 64x1
%on aura u 134x1 C 70x134 d 70x1 a 1x134 e 1x1
%avec gauss
%on avait C 46x64 et d 46x1 et u 64x1
%on aura u 110x1 C 46x110 d 46x1 a 1x110 e 1x1
```

2.7 Choix de l'arête

Arêtes au hasard de la numérotation (on peut aussi parcourir les i dans l'autre sens, les itérations ne sont pas les mêmes) :

```
atteint=1;
for i=1:nbinconnues+nbcontraintes
  %if (a(1,i)>0)
  if (a(1,i)>1e-10)
    atteint=0;
    arete=i;
  end
end
```

Arêtes choisies pour maximiser le coefficient correspondant dans J ie a_i :

```
atteint=1;
maxa=-1;
% choix de arete
for i=1:nbinconnues+nbcontraintes
  %if (a(1,i)>0)
  if (a(1,i)>1e-10)

    poids=a(1,i);

    atteint=0;
    if ((maxa==-1)||(poids>maxa))
      maxa=poids;
      arete=i;
    end
  end % if a(
end % for ou while i
```

Arêtes choisies pour maximiser le coefficient correspondant dans J ie a_i mais pondéré comme dans la librairie glpk (avec la norme euclidienne de la colonne normalisée) ${a_i}^2/(1 + \|C_i\|^2)$:

```
atteint=1;
maxpoids=-1;
% choix de arete
for i=1:nbinconnues+nbcontraintes
  %if (a(1,i)>0)
  if (a(1,i)>1e-10)

    % repris de glpk
    %on fait un calcul de poids, puisque a>0, on considere une
    %variable non deduite. ici c (et d) ont deja ete normalises
    %c(j,deduite(j))=1
    gamma=1+(c(1:nbcontraintes,i).*c(1:nbcontraintes,i));
    poids=a(1,i)^2/gamma;
    %poids=a(1,i);

    atteint=0;
    if ((maxpoids==-1)||(poids>maxpoids))
      maxpoids=poids;
      arete=i;
    end
  end % if a(
end % for ou while i
```

```
.......
    for i=1:nbcontraintes
      if (i~=ligne)
      .........
      end
      % on normalise c et d pour avoir c(i,deduite(i))=1
      coef=c(i,deduite(i));
      c(i,:)=c(i,:)/coef;
      d(i,1)=d(i,1)/coef;
    end
```

2.7.1 Octave glpk

Dans tous les cas, octave-glpk met une fraction de seconde pour les calculs.

2.7.2 Sudoku 4x4

64 inconnues principales plus autant que de contraintes.

- choix max indice sans gauss 64 contraintes : 9,7s 340 chgts sommet 33 vrais.
- choix max coef sans gauss 64 contraintes : 1,7s 60 chgts sommet 13 vrais.
- choix à la glpk sans gauss 64 contraintes : 2,8s 83 chgts sommet 10 vrais.
- choix max indice avec gauss 40 contraintes : 1,5s 57 chgts sommet 18 vrais.
- choix max coef avec gauss 40 contraintes : 1,6s 45 chgts sommet 15 vrais.
- choix à la glpk avec gauss 40 contraintes : 1,7s 48 chgts sommet 15 vrais.

2.7.3 Sudoku 9x9

Sudoku 9x9 avec Gauss pour avoir autant de contraintes que le rang 729 inconnues, 275 contraintes :

- arête choisies au hasard et correction des erreurs d'arrondi (en recalculant tout à partir de la liste des variables déduites quand c'est nécessaire) : 2h37 de calcul, 47544 changements de sommet dont 336 "vrais". Cela ne boucle jamais (comme le dit le Ciarlet).
- arêtes maximisant a_i, convergence en 230 s (4 minutes), 1647 changements de sommet donc 286 "vrais". jusqu'à la fin, plante pour $J = -0,2$.
- arêtes maximisant a_i mais pondéré comme dans glpk par ${a_i}^2/(1 + \|C_i\|^2)$ (C_i colonne j et norme 2) converge en 200 s (3 min), 1219 changements de sommet dont 126 "vrais"

Même sudoku 9x9 sans Gauss 729 inconnues, 350 contraintes :

- arêtes choisies au hasard et correction des erreurs d'arrondi (en recalculant tout à partir de la liste des variables déduites quand c'est nécessaire) : Convergence impossible. Les erreurs d'arrondis le font diverger.
- arêtes maximisant a_i, convergence presque jusqu'à la fin, plante pour $J = -0,2$.
- arêtes maximisant a_i mais pondéré comme dans glpk par ${a_i}^2/(1 + \|C_i\|^2)$ (C_i colonne j et norme 2) converge en 530 s (9 min), 2292 changements de sommet dont 765 "vrais"

2.8 Programme complet

```
clear

N=3;

l=0;

nbinconnues=(N^2)^3
u(1:nbinconnues,1)=zeros(nbinconnues,1);

uc=zeros(N^2,N^2);
```

```
uc(1,:)=[0,7,0,0,0,9,4,3,1];
uc(2,:)=[0,8,1,2,5,0,0,9,0];
uc(3,:)=[0,0,4,0,0,0,0,0,0];
uc(4,:)=[5,0,0,0,3,7,0,0,6];
uc(5,:)=[0,3,0,1,8,0,9,0,0];
uc(6,:)=[0,0,0,0,0,2,0,0,0];
uc(7,:)=[0,0,7,0,9,0,0,8,0];
uc(8,:)=[0,0,0,0,0,1,0,2,0];
uc(9,:)=[0,0,0,0,0,0,0,0,3];
uc

for i=1:(N^2)
  for j=1:(N^2)
    for k=1:(N^2)
      if (uc(i,j)==k)
        l=l+1;
        m=(k-1)*((N^2)^2)+(i-1)*(N^2)+j;
        c(l,m)=1;
        d(l,1)=1;
        connue(l)=m;
      end
    end
  end
end

% un chiffre par case sum(x(1,1,:)
for i=1:N^2;
  for j=1:N^2;
    l=l+1;
    for k=1:N^2;
      m=(k-1)*((N^2)^2)+(i-1)*(N^2)+j;
      c(l,m)=1;
    end
    d(l,1)=1;
  end
end


%lignes sum(x(1,:,1))
for i=1:N^2;
  for k=1:N^2;
    l=l+1;
    for j=1:N^2;
      m=(k-1)*((N^2)^2)+(i-1)*(N^2)+j;
      c(l,m)=1;
    end
    d(l,1)=1;
  end
end

%colonnes sum(x(:,1,1))
for j=1:N^2
  for k=1:N^2
    l=l+1;
    for i=1:N^2
      m=(k-1)*((N^2)^2)+(i-1)*(N^2)+j;
```

```
      c(l,m)=1;
    end
    d(l,1)=1;
  end
end


%blocs
for n=1:N
  for p=1:N
    for k=1:N^2
      l=l+1;
      for i=(n-1)*N+1:(n-1)*N+N
for j=(p-1)*N+1:(p-1)*N+N
          m=(k-1)*((N^2)^2)+(i-1)*(N^2)+j;
          c(l,m)=1;
end
      end
      d(l,1)=1;
    end
  end
end


nbcontraintes=l

temps=time();
rang=rank(c)

gj=0
if (gj==1)
  % reduction de gauss jordan pour avoir autant de ligne que le rang
  ligne=1;
  for colonne=1:nbinconnues
    probleme=0;
    if (c(ligne,colonne)==0)
      j=ligne+1;
      while((j<=nbcontraintes)&&(c(j,colonne)==0))
        j+=1;
      end
      if (j==nbcontraintes+1)
        probleme=1;
      else
        cligne=c(ligne,:);
        c(ligne,:)=c(j,:);
        c(j,:)=cligne;
        dligne=d(ligne,1);
        d(ligne,1)=d(j,1);
        d(j,1)=dligne;
      end
    end
    if (probleme==0)
      for j=1:nbcontraintes
        if (j~=ligne)
          coef=c(j,colonne)/c(ligne,colonne);
          c(j,:)=c(j,:)-coef*c(ligne,:);
```

```
          d(j,1)=d(j,1)-coef*d(ligne,1);
        end
      end
      ligne +=1;
    end
  end

  nbcontraintes=ligne-1
end

%on a besoin de d>=0 pour la suite
for ligne=1:nbcontraintes
  if (d(ligne,1)<0)
    c(ligne,:)=-c(ligne,:);
    d(ligne,1)=-d(ligne,1);
  end
end

%il n'y a pas de sommet evident de U pour lancer les calculs.
%on change legerement de probleme :
%Cu=d devient Cu+v=d et la nouvelle inconnue est (u,v) et J=-\sum v le
%but est d'arriver a 0. le sommet initial est u=0 v=d
%u(1:nbinconnues,1)=zeros(nbinconnues,1); deja fait en haut.
u(nbinconnues+1:nbinconnues+nbcontraintes)=d(1:nbcontraintes);
c(1:nbcontraintes,nbinconnues+1:nbinconnues+nbcontraintes)=diag(ones(1,nbcontraintes));
e=0;
a(1,1:nbinconnues)=zeros(1,nbinconnues);
a(1,nbinconnues+1:nbinconnues+nbcontraintes)=-ones(1,nbcontraintes);

J=a*u+e

deduite(1:nbcontraintes)=nbinconnues+1:nbinconnues+nbcontraintes;


%J depend des variables deduites, il faut faire les changements de
%variables pour supprimer la dependance en les variables deduites
% ie on veut a_j=0 pour j<=nbconnues et j>=nbcontraintes+1
% J=e+\sum a_j u_j,
% u_deduite(ligne)
% = (d_ligne-\sum j\not=deduite c_ligne,j u_j)/c_ligne,deduite(ligne)
% J=e+ a_deduite(ligne) d_ligne/c_ligne,deduite(ligne)
% + \sum j (a_j-a_deduite(ligne) c_ligne,j/c_ligne,deduite(ligne)) u_j

for ligne=1:nbcontraintes
  coef= a(1,deduite(ligne))/c(ligne,deduite(ligne));
  e = e + coef * d(ligne) ;
  a(1,:) = a(1,:) - coef * c(ligne,:);
end

% fin preliminaires
preliminaires=time()-temps

atteint=0;
compteur=0;
iteration=0;
permutation=0;
```

```
stop=0;
while ((atteint==0)&&(stop==0))
  compteur+=1
  permutation +=1
  % si au moins un a est >0, le max n'est pas atteint
  atteint=1;
  maxa=-1;
  % choix de arete
  for i=1:nbinconnues+nbcontraintes
    %if (a(1,i)>0)
    if (a(1,i)>1e-10)

      % repris de glpk
      %on fait un calcul de poids, puisque a>0, on considere une
      %variable non deduite. ici c (et d) ont deja ete normalises
      gamma=1+(c(1:nbcontraintes,i).*c(1:nbcontraintes,i));
      poids=a(1,i)^2/gamma;
      %poids=a(1,i);

      atteint=0;
      if ((maxa==-1)||(poids>maxa))
        maxa=poids;
        arete=i;
      end
    end % if a(
  end % for ou while i

  %calcul de theta
  t1=-1;
  %for j=1:nbcontraintes
  j=0;
  while((t1~=0)&&(j~=nbcontraintes))
    j +=1;
    %if (c(j,i)~=0)
    if ((abs(c(j,arete))>1e-10)&&(c(j,deduite(j))*c(j,arete)>1e-10))
      t2=d(j,1)/c(j,arete);
      if ((t1==-1)||(t2<t1))
        t1=t2;
        ligne=j;
      end
    end
  end % for ou while j

  theta=t1;

arete
new=deduite(ligne)
theta

  % si pas de contrainte sur theta J=infini
  if (theta==-1)
   theta
   atteint=1
  end

  % si le min n'est pas atteint on se dplace sur une des artes
```

```
  % u(arete) va augmenter de theta et les u(deduite(i)) diminuer et l'un
  % d'eux atteindra 0
  if (atteint==0)
    % la longueur maximale du dplacement est theta
    if (theta<1e-10)
      theta=0
    end

    % on se dplace dans la direction arete en ne modifiant que les
    % variables deduites
    u(arete,1)=theta;
    for i=1:nbcontraintes
      u(deduite(i),1)=u(deduite(i),1)-theta*c(i,arete)/c(i,deduite(i));
    end
    % pour eviter les pb d'arrondi, on force
    u(deduite(ligne))=0;
    minu=min(u)
    if (minu<-1e-10)
      stop=1
    end


    % on echange les roles de deduite(ligne) et arete
    % J=e+\sum a_j u_j et
    % u_arete = (d(ligne)-\sum j\not=arete c_ligne,j u_j)/c_ligne,arete
    % J=e+ a_arete d/c_arete + \sum j (a_j-a_arete c_j/c_arete) u_j
    coef= a(1,arete)/c(ligne,arete);
    e = e + coef * d(ligne,1) ;
    a(1,:) = a(1,:) - coef * c(ligne,:);
    % pour eviter les pb d'arrondi, on force
    a(1,arete)=0;

    arrondi=0;

    Jold=J;
    J=a*u+e

duree=time()-temps
    if (J-Jold>1e-10)
      iteration +=1
      permutation=0
      % on affiche le resultat sous forme d'un carre
      uc=zeros(N^2,N^2);
      for i=1:N^2
        for j=1:N^2
          for k=1:N^2
            uc(i,j)+=k*u((k-1)*((N^2)^2)+(i-1)*(N^2)+j);
          end
        end
      end
      uc
    end
    if(J-Jold<-1e-9)
      diffJ=J-Jold
      %arrondi=1
    end
```

```
    deduite(ligne)=arete;

    % \sum c_i,j u_j = d_i
    % u_arete = (d(ligne)-\sum j\not=arete c_ligne,j u_j)/c_ligne,arete
    % \sum (c_i,j - c_i,arete c_ligne,j/c_ligne,arete) u_j
    % = d_i -c_i,arete d(ligne)/c_ligne,arete
    for i=1:nbcontraintes
      if (i~=ligne)
        coef = c(i,arete) / c(ligne,arete);
        c(i,:)=c(i,:) - coef * c(ligne,:);
        d(i,1)=d(i,1) - coef * d(ligne,1);
        % pour eviter les pb d'arrondi, on force
        c(i,arete)=0;
      end
      % on normalise c et d pour avoir c(i,deduite(i))=1
      coef=c(i,deduite(i));
      c(i,:)=c(i,:)/coef;
      d(i,1)=d(i,1)/coef;
    end

  end % if non atteint

end % while non atteint
duree=time()-temps
iteration

    % on affiche le resultat sous forme d'un carre
    uc=zeros(N^2,N^2);
    for i=1:N^2
      for j=1:N^2
        for k=1:N^2
          uc(i,j)+=k*u((k-1)*((N^2)^2)+(i-1)*(N^2)+j);
        end
      end
    end
uc
```

Chapitre 3

Systèmes dynamiques : cycles et chaos

3.1 Présentation

On s'intéresse à des systèmes dynamiques : des équations différentielles sur $\mathbf{R}^N$ ici avec $N = 2$ et $N = 3$. On va approcher numériquement leur solution et oberver si l'on peut le faire correctement.

3.2 Théorèmes

Théoreme 28 *(Cauchy-Peano) Soient N un entier non nul, $x_0 \in \mathbf{R}^N$ $r > 0$, t_0 réel et $a > 0$. Enfin soit f une fonction continue sur $]t_0 - a, t_0 + a[\times B(x_0, r)$. Alors il existe $0 < c < a$ et une solution (pas nécessairement unique) du problème de Cauchy : $x'(t) = f(t, x(t))$ et $x(t_0) = x_0$, où x est définie sur $]t_0 - c, t_0 + c[$ et à valeur dans $B(x_0, r)$.*

Non unicité : $x' = 3x^{2/3}$. Calculs des solutions. Admet notamment $x = 0$ et $x = t^3$ comme solution valant 0 en 0. Solutions $max(0, (t - b)^3)$

On notera que $3x^{2/3}$ est bien continue et bornée sur $\mathbf{R} \times]-R, R[$ (pour tout $R > 0$) le théoreme s'applique donc avec n'importe quels a, r, t_0, x_0.

Théoreme 29 *(Cauchy-Lipschitz) Si plus, (qui à restreindre a et r) il existe $K > 0$ tel que pour tout $s \in]t_0 - a, t_0 + a[$ et y et $z \in B(x_0, r)$, on ait $\|f(s, y) - f(s, z)\| \leq K\|y - z\|$ alors la solution est unique (au sens où deux solutions coïncident sur leur domaine commun).*

La fonction $3x^{3/2}$ ne vérifie pas les hypothèses du second théorème pour $x_0 = 0$ mais les vérifie pour $x_0 \neq 0$.

3.3 Schéma d'Euler explicite

Pour $x'(t) = f(t, x(t))$, t_0, x_0 et h donnés on construit $x_{n+1} = x_n + hf(t_0 + nh, x_n)$ et qui sera une approximation d la solution au sens où pour h petit, $x_n \approx x(t_0 + nh)$.

Voir cours de méthodes pour l'analyse numérique.

Exemple : $x'(t) = x(t)$ et $x(0) = 1$, solution exacte $x(t) = e^t$. Solution approchée $x_{n+1} = x_n + hx_n = (1 + h)x_n$ donc $x_n = (1 + h)^n$ (on a bien $x_0 = 1$) où x_n approximation en $t = nh$. Si $h = T/N$ alors x_N approximation de $x(T) = e^T$ or $x_N = (1 + T/N)^N$ qui tend vers e^T pour N grand car $(1 + T/N)^N = e^{N \ln(1+T/N)}$ et $N \ln(1 + T/N) \approx NT/N = T$ et exponentielle continue. On retrouve la convergence du schéma d'Euler.

3.4 Cycles et système proies-prédateurs

3.4.1 Etude

Pour α, β, γ, $\delta > 0$

$$x'(t) = x(t)(\alpha - \beta y(t)) = \alpha x(t) - \beta x(t)y(t)$$
$$y'(t) = y(t)(-\gamma + \delta * x(t)) = -\gamma y(t) + \delta y(t)x(t)$$

interpretation de chacun des termes. toutes seules les proies de multiplient mais tous seuls les prédateurs disparaissent d'où le signe moins devant γ.

Ce système vérifie les hypothèses des deux théoremes car ...

3.4.2 Approximation numérique

```
clear

alpha=1
beta=2
gamma=1
delta=3

N=400;
dt=1/N
T=20
t=0:dt:T-dt;
i=1;

x(i)=1/2
y(i)=1/2

for i=2:T*N
  xp=x(i-1);
  yp=y(i-1);

  x(i)=xp + dt * xp * (alpha - beta * yp);
  y(i)=yp + dt * (-yp) * (gamma - delta * xp );

end
plot(x,y)
%plot(t,x)
```

Pour h relativement grand, on voit une spirale. Il faut réduire h pour voir un cycle.

On peut observer le phenomène de convergence, en considerant $x(T)$ pour différentes valeurs de h. Si on choisit $T = 5$, on observe que la fonction $h \mapsto x(T)$ (en fait `x((size(x)(1)))`) est presque linéaire. Ceci correspond au fait que la méthode d'Euler est d'ordre 1 : l'erreur est en Ch donc $x(T) \approx a + bh$. Grâce à la regression linéaire on peut trouver la "meilleure" approximation de $x(T)$.

Si T est grand (50), on trouve plutôt un ordre de $1,2$. Il est possible que cela vienne des erreurs d'arrondis qui perturbent les calculs. Ou ? ? ? ?

3.5 Chaos et système de Lorentz

3.5.1 Le système

On s'intéresse à un système issu de la mécanique des fluides et étudié par Lorentz dans un logiciel de calcul de météorologie. Il a observé que s'il arrêtait le programme et le relançait à partir des

dernières valeurs calculées, il n'obtenait pas les mêmes résultats : ces valeurs avaient été arrondies par l'affichage.

C'est un phénomène de très forte sensibilité aux conditions initiales. Le système est chaotique.

Pour ρ, β, $\sigma > 0$

$$\begin{aligned} x'(t) &= \sigma(y(t) - x(t)) \\ y'(t) &= x(t)(\rho - z(t)) - y(t) \\ z'(t) &= x(t)y(t) - \beta z(t) \end{aligned}$$

On choisira $\rho = 28$, $\sigma = 10$ et $\beta = 8/3$, pour lesquelles apparait l'attracteur de Lorentz.

On partira de $x(0) = y(0) = z(0) = 1$.

3.5.2 Approximation numérique

```
clear

rho=28
sigma=10
beta=8/3

N=700;
dt=1/N
T=40
t=0:dt:T-dt;
i=1;

x(i)=1
y(i)=1
z(i)=1

for i=2:T*N
  xp=x(i-1);
  yp=y(i-1);
  zp=z(i-1);

  x(i)=xp + dt * sigma * (yp-xp);
  y(i)=yp + dt * (xp*(rho -zp) - yp);
  z(i)=zp + dt * (xp*yp - beta * zp);
end
%plot3(x,y,z) param3d en scilab
%axis([-15,0,-20,0,10,50])
%view(60,30)
plot(t,x)
```

Pour T suffisamment grand et de nombreuses valeurs du pas de temps, on observe l'attracteur en ailes de papillon. Mais si on trace $t \mapsto x(t)$, on observe des courbes très différentes pour des valeurs mêmes proches du pas de temps. On retrouve l'aspect chaotique.

En fait le schéma est très loin de la solution exacte et n'a pas convergé mais comme il y a un attracteur il n'y a pas d'explosion. La solution (fausse) reste dans la bonne zone.

L'erreur est bien en Ch mais C est énorme et les valeurs de h permettant une bonne approximation sont inattegnables. En fait C est en e^{cT}.

En revanche pour T petit (1), on retrouve la convergence attendue.

Chapitre 4

Dimension de fractales

4.1 Problème

On s'intéresse à des objets autosimilaires (fractales invention de B. Mandelbrot) inclus dans $\mathbf{R}^2$ mais de longueur infinie : leur dimension sera comprise entre 1 et 2.

4.2 Définition

Définition 30 *Dimension de Hausdorff. Soit $X \in \mathbf{R}^2$, et soient s et r réels strictement positifs, on définit*

$$H^{s,r}(X) = \inf\left\{\sum_{i=1}^{\infty} diam(A_i)^s, X \subset \bigcup_{i=1}^{\infty} A_i, A_i, diam(A_i) < r\right\}.$$

On appelle dimension de Hausdorff :

$$\dim_H(X) = \inf\{s, \lim_{r\to 0} H^{s,r}(X) = 0\} = \sup\{s, \lim_{r\to 0} H^{s,r}(X) = \infty\}.$$

La définition est compliquée et on admet que l'inf et le sup sont égaux. On rappelle que

Définition 31 *$diam(A) = \sup\{d(x,y), x \in A, y \in A\}$.*

Définition 32 *Une distance d est une application de $E \times E$ dans $\mathbf{R}^+$ telle que $d(x,y) = d(y,x)$ pour tous x et y dans E, $d(x,x) = 0$ est équivalent à $x = 0$ et que $d(x,y) \leq d(x,z) + d(z,y)$ pour tous x, y et z dans E.*

Ex des distances provenant de normes (voir aprs). $d_0(x,y) = 0$ si $x = y$ et $d(x,y) = 1$ si $x \neq y$ (on vérifie aisement chacune des propriétés).

Définition 33 *Une norme sur un K espace vectoriel E est une application $\|.\|$ de $E \times E$ dans $\mathbf{R}^+$ telle que $\|x\| = 0$ implique $x = 0$, $\|\lambda x\| = |\lambda|\|x\|$ pour tout $x \in E$ et $\lambda \in K$, et $\|x - y\| \leq \|x - z\| + \|z - y\|$*

On remarque qu'une norme définit une distance $d(x,y) = \|x - y\|$ et on a donc les exemples de normes sur $\mathbf{R}^2$: $\|x\|_1 = |x_1| + |x_2|$, $\|x\|_2 = \sqrt{{x_1}^2 + {x_2}^2}$ et $\|x\|_\infty = max(|x_1|, |x_2|)$.

On peut définir les boules pour chacunes de ces distances : $B(x,r) = \{y, d(x,y) \leq r\}$. Pour d_0, $B(x,1) = \{x\}$. Pour la distance provenant de $\|.\|_1$: $B(x,1)$ est un losange, pour $\|.\|_2$ un cercle et pour $\|.\|_\infty$ un carré.

Les boules sont souvent utilisées pour les A_i.

Exemple de dimension de Hausdorff :

Propriété 34 *Le segment $[0,1] \times \{0\}$ est de dimension 1*

Démo : On peut le voir dans $\mathbf{R}$ ou dans $\mathbf{R}^2$. Dans $\mathbf{R}$ c'est plus simple à écrire.

- Pour $s > 1$, on recouvre $[0,1]$ par des segments (fermés) de tailles $r/2$, il en faut $2/r$ (au moins), alors $H^{s,r} \leq 2/r(r/2)^s = (r/2)^{s-1}$. Donc $\lim_{r\to 0} H^{s,r} = 0$ donc $\dim_H([0,1]) \leq 1$.

- Pour $s = 1$, on considère $\sum_{i=1}^{\infty} diam(A_i)$ avec $[0,1] \subset \bigcup_{i=1}^{\infty} A_i$ et $diam(A_i) < r$. On va montrer que $\sum_{i=1}^{\infty} diam(A_i) \geq 1$. D'abord par compacité de $[0,1]$, on peut supposer (ou extraire) que les A_i sont en nombre fini. On raisonne alors par récurrence. Pour $n = 1$, $[0,1] \subset A_1$ donc $1 \leq diam(A_1)$. Si la proprétés est vraies pour n intervalles ouverts, montrons la pour $n+1$ intervalles ouverts. Si on note A_{n+1} l'intervalle le plus à droite (au sens de sa borne droite donc $b > 1$, sinon il faut ajouter un autre intervalle pour recouvrir $[0,1]$), soit $A_{n+1} =]a, b[$.

Soit $J = [0,1] \cap]-\infty, a]$. Alors J est un intervalle inclus dans $\cup_{i=1}^{n} A_i$ donc par récurrence $diam(J) \leq \sum_{i=1}^{n} diam(A_i)$.

Soit $K = [0,1] \cap]a, \infty[$, qui est aussi un intervalle mais qui est inclus dans A_{n+1} donc $diam(K) \leq diam(A_{n+1})$.

Enfin on a $diam([0,1]) \leq diam(J) + diam(K)$ (en fait $J = [0,a]$ et $K =]a,1]$ dans le seul cas intéressant où $0 < a < 1$).

- Soit maintenant $s < 1$, on a

$$\sum_{i=1}^{\infty} diam(A_i) = \sum_{i=1}^{\infty} diam(A_i)^s diam(A_i)^{1-s} \leq r^{1-s} \sum_{i=1}^{\infty} diam(A_i)^s$$

donc $\sum_{i=1}^{\infty} diam(A_i)^s \leq r^{s-1}$ et $\lim_{r\to 0} H^{s,r} = +\infty$ pour tout $s < 1$ donc $\dim_H([0,1]) \geq 1$.

Propriété 35 *L'ensemble des rationnels* $(\mathbf{Q} \cap [0,1]) \times \{0\}$ *est de dimension 0*

Démo : Soit f une bijection entre $\mathbf{N}$ et $\mathbf{Q}$. Donc $\mathbf{Q} = \{f(0), f(1), \ldots\}$. On considère en fait $(\mathbf{Q} \cap [0,1]) \times \{0\}$. On recouvre chaque $f(n)$ par une boule centrée sur ce point et de diametre $min(r/2, 1/(Cn^\sigma))$ où C et σ sont des constantes que l'on va ajuster par la suite. Alors $\sum_{k=1}^{\infty} diam(A_i)^s \leq \sum_{i=1}^{\infty} 1/(Cn^\sigma)^s = 1/C^s \sum_{i=1}^{\infty} 1/n^{\sigma s}$. Si l'on choisit $\sigma = 2/s$ pour $s > 0$, on a $\sum_{k=1}^{\infty} diam(A_i)^s \leq 1/C^s \pi/6$. Et comme on peut choisir C aussi grand que l'on souhaite : $H^{s,r} = 0$ pour tout $s > 0$ donc $\dim_H((\mathbf{Q} \cap [0,1]) \times \{0\}) = 0$.

Définition 36 *On recouvre un ensemble par une partition (dont les éléments sont de tailles (diamtres, rayons, ...) d'une partie de* $\mathbf{R}^2$ *on appelle dimension "boite" ou de Minkowski de E, la limite (si elle existe) de* $\lim_{r\to 0} -\ln N_r / \ln r$.

Par exemple avec N_r boules (partiellement ouvertes et fermées pour que cela fasse une partition, ie sans recouvrement) de rayon r pour la norme $\|.\|_\infty$ et

On notera que l'on peut prendre des boules de rayon r ou de diametre r cela ne change rien a la limite : pour une boule carrée, si $\|x\| \leq r$, le diametre est $2\sqrt{2}r$ et on a donc $\ln 2\sqrt{2}r = \ln 2\sqrt{2} + \ln r = \ln r(1 + \ln 2\sqrt{2}/\ln r)$. Et $\lim_{r\to 0} \ln 2\sqrt{2}/\ln r = 0$.

Propriété 37 *Le segment* $[0,1] \times \{0\}$ *est de dimension 1*

Démo : Toujours en dimension 1 pour simplifier, on voit que si l'on recouvre $[0,1]$ par des intervalles de tailles r, il en faut la partie entière superieure de $1/r$ (ex $r = 2/3$). Alors $1 = -\ln(1/r)/\ln r \leq -\ln N_r/\ln r \leq -\ln(1/r+1)/\ln r = 1 - \ln(1+r)/\ln r$ donc $\lim_{r\to 0} -\ln N_r/\ln r = 1$.

Propriété 38 *Le carré* $[0,1]^2$ *est de dimension 2.*

Démo : Si l'on recouvre ce carré par des petits carrés de taille r, il en faut la partie entière superieure de $1/r^2$ (ex $r = 2/3$). Alors $2 = -\ln(1/r^2)/\ln r \leq -\ln N_r/\ln r \leq -\ln(1/r^2+1)/\ln r = 2 - \ln(1+r^2)/\ln r$ donc $\lim_{r\to 0} -\ln N_r/\ln r = 2$.

Propriété 39 *L'ensemble des rationnels* $(\mathbf{Q} \cap [0,1]) \times \{0\}$ *est de dimension 1.*

Dmo : $\mathbf{Q}$ est dense dans $\mathbf{R}$ donc tout les ensembles (d'interieur non vide) qui recouvrent $[0,1]$ doivent être comptés. Le calculs est donc le même que pour $[0,1]$. La dimension est donc 1.

4.3 La courbe de Von Koch

Cela date d'un article de 1906. On effectue de façon récurrente la transformation suivante : on part du segment $[0,1] \times \{0\}$. On conserve les deux segments $[0,1/3] \times \{0\}$ et $[2/3,1] \times \{0\}$ et on replace le segment $[1/3,2/3] \times \{0\}$, par les deux autres segments du triangle équilatéral (orienté vers le haut) ayant $[1/3,2/3] \times \{0\}$ pour coté. En recommence ensuite en appliquant la même transformation à chacun des 4 segments. etc.

Les coordonnées du sommet du triangle sont $(1/2, \sqrt{3}/6)$. Mais il faut une formule plus générale pour effectuer les itérations suivantes : si on part du segment $[(x_1,y_1),(x_2,y_2]]$, on a ensuite les segments $[(x_1,y_1),(2x_1/3+x_2/3, 2y_1/3+y_2/3)]$ et $[(x_1/3+2x_2/3, y_1/3+2y_2/3),(x_2,y_2]]$, reste les deux "nouveaux" segments : le nouveau sommet est tel que le segment qui le relie au milieu de $[(x_1,y_1),(x_2,y_2]]$ est orthogonal à ce segment et de longueur $sqrt3/6$ fois la longueur du segment.

Ceci conduit donc à l'algorithme suivant :

```
clear

N=600
nb=5
3^nb
dessin=zeros(N,N);

courbe(1,1)=0;
courbe(1,2)=0;
courbe(2,1)=1;
courbe(2,2)=0;

tsegment=time();
for iter=1:nb

for k=1:4^(iter-1)
  courbe2(4*(k-1)+1,1)=courbe(k,1);
  courbe2(4*(k-1)+1,2)=courbe(k,2);
  courbe2(4*(k-1)+2,1)=(2*courbe(k,1)+courbe(k+1,1))/3;
  courbe2(4*(k-1)+2,2)=(2*courbe(k,2)+courbe(k+1,2))/3;
  courbe2(4*(k-1)+3,1)=(courbe(k,1)+courbe(k+1,1))/2-(courbe(k+1,2)-courbe(k,2))*sqrt(3)/6;
  courbe2(4*(k-1)+3,2)=(courbe(k,2)+courbe(k+1,2))/2+(courbe(k+1,1)-courbe(k,1))*sqrt(3)/6;
  courbe2(4*(k-1)+4,1)=(courbe(k,1)+2*courbe(k+1,1))/3;
  courbe2(4*(k-1)+4,2)=(courbe(k,2)+2*courbe(k+1,2))/3;
  courbe2(4*(k-1)+5,1)=courbe(k+1,1);
  courbe2(4*(k-1)+5,2)=courbe(k+1,2);
end
  courbe=courbe2;

end
tsegment=time()-tsegment
x=courbe(:,1);
y=courbe(:,2);
plot(x,y)
tdessin=time();

tdessin=time()-tdessin
```

Pour calculer ou approcher la dimension boite, on suppose que la taille des segments est petites devant celle des boite (ie avec les inverses N comparé à 3^{nb}) et qu'il suffit donc de regarder dans quelles boites tombent les extremites de l'intervalle.

```
tboite=time();
```

```
for k=1:4^nb+1
  i=floor(courbe(k,1)*N);
  j=floor(courbe(k,2)*N);
  dessin(N-j,i+1)=1;
end
tboite=time()-tboite
imagesc(boite)
log(sum(sum(boite)))/log(N)
sum(sum(boite))
```

ce qui donne différentes valeurs de N_r fonction de r. On peut donc observer les oscillations de $-\ln N_r/\ln r$ autour de $1,28$ alors que la dimension est $\ln 4/\ln 3 \approx 1,26$. On peut observer que $\ln r \mapsto -\ln N_r$ est très proche d'une droite (ce qui est équivalent à observer que $N_r \approx C(1/r)^d$ où d est la dimension boite. Pour estimer au mieux d on peut utiliser la regression linéaire (avec méthode des moindres carrés, voir après) :

```
r=[1/10,1/12,1/17,1/20,1/25,1/30,1/35,1/40,1/45,1/50,1/55,1/60,1/65,1/70,1/75,
1/80,1/85,1/90,1/95,1/100,1/120,1/140];
nr=[17,27,36,47,61,81,103,115,142,157,172,204,218,233,270,280,295,321,351,357,476,621];

plot(log(r),log(nr),log(r),-1.283.*log(r)+0.02)
polyfit(log(r),log(nr),1)

#plot(r,log(nr)./log(r),r,-1.283+0.02./log(r))
```

On constate que $C \approx e^{0,02} \approx 1$ et ici $d \approx 1,28$. Soit $1,6\%$ d'erreur par rapport à $1,26$. Ici $-\ln N_r/\ln r$ est proche de sa limite : $-\ln N_r/\ln r \approx -0,02/\ln r + d$ pour $r = 1/10$, $-0,02/\ln r = 0,009$ et pour $r = 1/140$, $-0,02/\ln r = 0,004$. Soit moins de 1% d'erreur.

4.4 Fractale de Newton

4.4.1 Définitions

bassins d'attractions d'une suite

Définition 40 *Pour une suite récurrente admettant au moins une limite, on appelle bassin d'attraction d'une des limites l, l'ensemble des valeurs initiales de la suite pour lesquelles la suite converge vers l.*

Exemple la suite $u_{n+1} = u_n^2$ admet deux limites possibles 0 et 1. Le bassin d'attraction dans $\mathbf{R}$ de 1 est $\{-1, 1\}$, celui de 0 est $]-1, 1[$.

fonctions holomorphes

On considère ici des fonctions de $\mathbf{C}$ dans $\mathbf{C}$. Elles peuvent aussi se voir comme des fonctions de $\mathbf{R}^2$ dans $\mathbf{R}^2$.

Comme fonction de $\mathbf{R}^2$ dans $\mathbf{R}^2$, une fonction peut être différentiable. On peut aussi généraliser sur $\mathbf{C}$ la notion réelle de dérivée.

Ex : $f(z) = z^3 - 1$ et $g(z) = \overline{z}$, ou encore $f(x+iy) = (x^3 - 3xy^2 - 1) + i(3x^2y - y^3)$ et $g(x+iy) = x - iy$. On peut voir ces deux fonctions comme de $\mathbf{R}^2$ dans $\mathbf{R}^2$: $f(x, y) = (x^3 - 3xy^2 - 1, 3x^2y - y^3)$ et $g(x, y) = (x, -y)$, on voit aisément qu'elles sont différentiables (comme polynomes). Pour voir si elles sont différentiables au sens de $\mathbf{C}$, il faut regarder $\lim_{z\to z_0}(f(z) - f(z_0))/(z - z_0)$ et $\lim_{z\to z_0}(g(z) - g(z_0))/(z - z_0)$.

D'une part $(f(z) - f(z_0))/(z - z_0) = (z^3 - {z_0}^3)/(z - z_0) = z^2 + zz_0 + {z_0}^2$ donc $\lim_{z\to z_0}(f(z) - f(z_0))/(z - z_0) = 3{z_0}^2$. Donc f est dérivable dans $\mathbf{C}$.

D'autre part $(g(z) - g(z_0))/(z - z_0) = \overline{z - z_0}/(z - z_0)$. Si $z = z_0 + h$ avec $h \to 0$, la limite est 1, si $z = z_0 + ih$ la limite est -1. Donc g n'est pas dérivable dans $\mathbf{C}$.

Définition 41 *Un fonction f de $\mathbf{C}$ dans $\mathbf{C}$ est holomorphe, si pour tout z_0, $\lim_{z\to z_0}(f(z)-f(z_0))/(z-z_0)$ existe.*

On voit par l'exemple ci-dessus que cette notion est plus forte que celle de la différentiabilité sur $\mathbf{R}^2$.

méthode de Newton

Pour résoudre de façon approchée $f(z)=0$ pour f différentiable de $\mathbf{R}^2$ dans $\mathbf{R}^2$, on observe que

$$0=f(z_{n+1})=f(x+h,y+k)\approx f(x,y)+Df(x,y)(h,k)=f(z_n)+Df(z_n)(z_{n+1}-z_n)$$

et on construit donc z_n par $z_{n+1}=z_n-Df(z_n)^{-1}f(z_n)$.

Pour résoudre de façon approchée $f(z)=0$ pour f holomorphe (sur $\mathbf{C}$), on observe que

$$0=f(z_{n+1})\approx f(z_n)+f'(z_n)(z_{n+1}-z_n)$$

et on construit donc z_n par $z_{n+1}=z_n-f(z_n)/f'(z_n)$.

4.4.2 dimension de la frontière des bassins d'attraction pour la méthode de newton pour $z^3-1=0$

Comme $f(z)=z^3-1$ est holomorphe et que l'on peut utiliser les calculs dans $\mathbf{C}$, on s'intéresse donc à la suite $z_{n+1}=z_n-({z_n}^3-1)/(3{z_n}^2)$.

bassins

On approche les bassins d'attractions en considérant z_0 dans le carré $[-2,2]^2$, avec un maillage uniforme : $z_0=-2+k*h+i(-2+l*h)$. Il est de taille $N+1$ par $N+1$ (pour $h=4/N$).

L'équation admet 3 solutions et la suite 3 limites, il y a donc 3 bassins d'attraction.

On notera une limitation du nombre d'itération au cas où la suite ne converge pas et le traitement de z_n petit.

```
clear;

# bug octave
imread("toto.png")

N=200;

tol=10^(-3);
maxi=50;

attraction=zeros(N+1,N+1);
boucle=time();
for k=1:N+1
  for l=1:N+1
    a=-2+4*(k-1)/N;
    b=-2+4*(l-1)/N;
    z=a+i*b;
    iter=0;
    while(abs(z^3-1)>tol & iter<maxi)
      if (abs(z)<tol)
        z=tol;
      end
      z2=z-(z^3-1)/(3*z^2);
      z=z2;
      iter=iter+1;
```

```
    end
    if (iter==maxi)
      attraction(N+2-l,k)=0;
    else
      if (abs(z-1)<10*tol)
        attraction(N+2-l,k)=1;
      end
      if (abs(z+1/2-i*sqrt(3)/2)<10*tol)
        attraction(N+2-l,k)=2;
      end
      if (abs(z+1/2+i*sqrt(3)/2)<10*tol)
        attraction(N+2-l,k)=3;
      end
    end
  end
end
boucle=time()-boucle

imagesc(attraction)
#imwrite(uint8(60*attraction),"attraction.png")

max(max(attraction))
min(min(attraction))
```

frontière

On construit un maillage "dual" décalé d'une demi maille (horizontalement et verticalement). Ce nouveau maillage couvre une zone plus petite que le premier il sera de taille N. Chaque point du nouveau maillage est donc entouré de 4 points du premier maillage :

$$\begin{array}{ccccc} 1 & & 2 & & 3 \\ & 4 & & 5 & \\ 6 & & 7 & & 8 \end{array}$$

Où les points 4 et 5 appartiennent au second maillage, les autres appartement au premier.

On dira que la frontière passe en 4 si (1 et 2) ou (1 et 6) ne sont pas dans le même bassin et on dira que la frontière passe en 5 si (2 et 3) ou (2 et 7) ne sont pas dans le même bassin. On notera une sorte de disymetrie : si (2 et 7) ne sont pas dans le même bassin, on pourrait dire que la frontieère pas en 4, mais cela fait alors une frontière double dans le cas d'une frontire en diagonale.

Ceci n'est qu'une définition, on peut en choisir d'autres.

```
clear;

attraction=imread("attraction.png");

m=size(attraction)(1)-1

boucle=time();
frontiere=zeros(m,m);
for k=2:m
  for l=2:m
    if (attraction(k,l)~=attraction(k-1,l))
      frontiere(k-1,l)=1;
    end
    if (attraction(k,l)~=attraction(k,l-1))
      frontiere(k,l-1)=1;
    end
```

```
  end
end
boucle=time()-boucle
imagesc(frontiere)
#imwrite(uint8(200*frontiere),"frontiere.png")
```

dimension

On recouvre le maillage "dual" par des carrés et on compte le nombre de ces carrés contenant au moins un point de la frontière.

```
clear;

frontiere=imread("frontiere.png");

m=(size(frontiere))(1,1);
h=4/m
n=14

reduit=zeros(floor(m/n),floor(m/n));
boucle=time();
for k=1:ceil(m/n)
  for l=1:ceil(m/n)
    reduit(k,l)=min(1,sum(sum(frontiere(1+(k-1)*n:min(k*n,m),1+(l-1)*n:min(l*n,m)))));
  end
end
boucle=time()-boucle
imagesc(reduit)
-log(sum(sum(reduit)))/log(n*h)
sum(sum(reduit))
```

La convergence n'est pas très bonne : on trouve des valeurs de $2,33$ ou plus alors que la dimension est environ $1,42$.

On va utiliser la méthode des moindres carrés : on constate que $\ln N_r$ fonction de $\ln r$ est presque une droite. C'est à dire que $N_r \approx C(1/r)^d$:

```
nr=[26,40,47,61,78,112,143,216,315,414,501]
r=[1/2,8/20,7/20,6/20,1/4,4/20,16/100,1/8,2/20,8/100,7/100]
plot(log(r),log(nr))
polyfit(log(r),log(nr),1)
#-1.4837   2.2934
#plot(r,log(nr)./log(r),r,-1.4837+2.2934./log(r))
```

On observe donc que $\ln N_r \approx -1.4837 \ln r + 2.2934$, soit $N_r \approx 9,9(1/r)^{1.4837}$ et $-\ln N_r/\ln r \approx 1.4837 - 2.2934/\ln r$. Comme $1/\ln r$ tend bien vers 0 pour r petit on peut donc approcher la dimension de la frontière par $1,48$. Soit une erreur de 4% par rapport à 1.42 qui est une approximation "officielle".

Ici C est plus grand ce qui ralentit le passage à la limite : pour $r = 1/2$, on a $-2.2934/\ln r \approx 3,3$ et pour $r = 7/100$, $-2.2934/\ln r \approx 0,86$. On observe dans ce dernier cas que $1.4837 - 2.2934/\ln r \approx 2,34$ ce qui est proche de la meilleure valeur de $-\ln N_r/\ln r$ onservée.

Si l'on trace (ou dérive) $-1/\ln r$ fonction de r, on voit la pente infinie en 0, aussi $1/\ln r$ converge lentement vers 0. Si l'on veut $-2.2934/\ln r \approx 0,01$ il faut $r \approx 2.510^{-100}$ et si l'on veut "seulement" $0,1$, il faut $r \approx 10^{-10}$. Cette méthode de calcul de la pente est donc nécessaire.

4.4.3 Méthode des moindre carrés

On se place dans le cas simplifié suivant : on cherche la "meilleure" droite approchant les points $(0,0)$, $(2,3)$ et $(4,4)$, c'est-à-dire $y = ax + b$. On voit que pour cet exemple, on aura $a \approx 1$ et $b \approx 0$.

S'il n'y avait que deux points on pourrait calculer de façon exacte a et b a l'aide d'un système à deux équations et deux inconnues.

Ici il y a trois points. Observons où passe la droite que l'on cherche : $(0, b)$, $(2, 2a+b)$, $(4, 4a+b)$. On peut donc calculer l'erreur par rapport aux données. On utilisera la norme quadratique : $E^2(a, b) = (0-b)^2 + (3-(2a+b))^2 + (4-(4a+b))^2$. Soit

$$\begin{aligned} E^2(a,b) &= b^2 + 9 - 6(2a+b) + (2a+b)^2 + 16 - 8(4a+b) + (4a+b)^2 \\ &= 25 - 12a - 6b - 32a - 8b + b^2 + 4a^2 + 4ab + b^2 + 16a^2 + 8ab + b^2 \\ &= 25 - 44a - 14b + 20a^2 + 12ab + 3b^2 \end{aligned}$$

La fonction E^2 de $\mathbf{R}^2$ dans $\mathbf{R}$ étant régulière, on cherche l'endroit où ses dérivées partielles $\partial E^2/\partial a = -44 + 40a + 12b$ et $\partial E^2/\partial b = -14 + 12a + 6b$. s'annullent. Soit $40a + 12b = 44$ et $12a + 6b = 14$, en éliminant b : $16a = 16$ donc $a = 1$ et $b = 1/3$. Ce qui est proche comme prévu de 1 et 0. On le retrouve avec

```
x=[0,2,4]
y=[0,3,4]
polyfit(x,y,1)
```

Chapitre 5

GPS et équations non linéaires

5.1 Problème

On souhaite déterminer sa position à partir de données fournies par des satellites. Par exemple, en connaissant sa distance par rapport aux satellites.

Si on connait le temps de parcours (à la vitesse de la lumière) de l'heure donnée par le satellite (comparée à celle du recepteur), on a $d = vt$. Cependant v est très élevée (300000 km/s), aussi s'il y a une incertitude δt sur la mesure en temps on obtient une incertitude sur la distance δd : $d + \delta d = v(t + \delta t)$, soit $\delta = v\delta t$. Si δt est de l'ordre de la milliseconde alors δd est de l'ordre de 300 km. L'information est donc inutilisable.

L'idée consiste donc à comparer le temps de parcours de l'information venant de 2 satellites (munis d'une horloge très précise). Ainsi l'incertitude est fortement réduite : si le recepteur fonctionne à une fréquence de 100 MHz soit une période de 10^{-8} seconde, l'incertitude en distance se réduit à 3 m.

La distance étant obtenue par différence, pour obtenir un point en dimension 2, il faut l'intersection de 2 courbes et chaque courbe provenant de 2 satellites (pas nécessairement distincts). Il faut donc au moins 3 satellites. Si l'on avait pu obtenir la distance à chaque satellite, 2 auraient suffit.

5.2 Coniques

On considère uniquement les hyperboles et les ellipses. Les hyperboles sont les coniques définies comme les courbes dont la différence des distances à deux points donnés est fixe. Les ellipses celles dont la somme des distances est fixe.

On appelle *foyers* les 2 points participant à la définition de ces coniques.

Supposons que u et v soient les vecteurs reliant un des points de la conique à chacun des foyers.

5.2.1 Hyperboles

L'hyperbole (ou plutôt l'une des deux branches) se définit donc par $\|u\| - \|v\| = d$. Quel contrainte pour d ?

L'inégalité triangulaire donne : $\|u\| = \|v + (u - v)\| \leq \|v\| + \|u - v\|$, soit $\|u\| - \|v\| \leq \|u - v\|$. donc $d \leq \|u - v\|$, c'est dire que d est plus petit que la distance en les deux satellites (faire un dessin). Et en inversant u et v on voit que $|d| \leq \|u - v\|$.

Si $d = \|u - v\|$, l'hyperbole se réduit à la droite joignant les deux foyers privée du segment dont ils sont les extremités.

5.2.2 Ellipses

L'ellipse se définit donc par $\|u\| + \|v\| = d$. Quel contrainte pour d ?

L'inégalité triangulaire donne : $\|u - v\| =\leq \|u\| + \|v\| = d$, donc $d \geq \|u - v\|$, c'est dire que d est positif et plus grand que la distance en les deux satellites (faire un dessin).

Si $d = \|u - v\|$, l'ellipse se réduit au segment joignant les deux foyers.

5.2.3 Equation

Soit deux foyers (x_1, y_1) et (x_2, y_2) et une distance d. On note (x, y) un point de la conique. On a donc

$$\sqrt{(x-x_1)^2+(y-y_1)^2} \pm \sqrt{(x-x_2)^2+(y-y_2)^2} = d$$

Pour a et b deux réels positifs, si $\sqrt{a} \pm \sqrt{b} = d$ on a, en élevant au carré, $a + b \pm 2\sqrt{ab} = d^2$ (la réciproque étant fausse), soit $a+b-d^2 = \mp 2\sqrt{ab}$ donc en élevant encore au carré, $(a+b-d^2)^2 = 4ab$ (la réciproque est aussi fausse). Soit $(a-b)^2 - 2d^2(a+b) + d^4 = 0$.

Aussi si l'on choisit $a = (x-x_1)^2 + (y-y_1)^2$ et $b = (x-x_2)^2 + (y-y_2)^2$, on a $a - b = (x-x_1)^2 - (x-x_2)^2 + (y-y_1)^2 - (y-y_2)^2$ soit $a-b = (x_2-x_1)(2x-x_1-x_2) + (y_2-y_1)(2y-y_1-y_2)$ ainsi $a-b$ est un polynôme de degrès 1 en x et y. Et l'équation est donc un polynôme de degrès au plus 2 en x et y.

Développons en ne conservant que les termes d'ordre 2, $(a-b)^2 = (x_2-x_1)^2(4x^2) + (y_2 - y_1)^2(4y^2) + 2(x_2-x_1)(y_2-y_1)(2x2y) + \ldots$ L'équation devient donc $4[(x_2-x_1)^2(x^2) + (y_2-y_1)^2(y^2) + 2(x_2-x_1)(y_2-y_1)(xy)] - 2d^2[2x^2+2y^2] + \ldots = 0$ soit

$$4[(x_2-x_1)^2 - d^2]x^2 + 4[(y_2-y_1)^2 - d^2]y^2 + 8(x_2-x_1)(y_2-y_1)(xy) + \ldots = 0$$

Pour retrouver la nature (ellipse ou hyperbole), il faut faire une réduction de Gauss. Supposons d'abord que $(x_2-x_1)^2 - d^2 > 0$, on a donc

$$4\left(\sqrt{(x_2-x_1)^2-d^2}x + \frac{(x_2-x_1)(y_2-y_1)}{\sqrt{(x_2-x_1)^2-d^2}}y\right)^2 - 4\frac{(x_2-x_1)^2(y_2-y_1)^2}{(x_2-x_1)^2-d^2}y^2 + 4[(y_2-y_1)^2-d^2]y^2 + \ldots = 0$$

soit

$$4\left(\sqrt{(x_2-x_1)^2-d^2}x + \frac{(x_2-x_1)(y_2-y_1)}{\sqrt{(x_2-x_1)^2-d^2}}y\right)^2 - 4d^2\frac{(x_2-x_1)^2+(y_2-y_1)^2-d^2}{(x_2-x_1)^2-d^2}y^2 + \ldots = 0$$

Or pour une hyperbole (non dégénérée), on a vu que $d < \|u-v\|$ c'est-à-dire, ici, $(x_2-x_1)^2 + (y_2 - y_1)^2 - d^2 > 0$ et pour une ellipse (non dégénérée) $(x_2-x_1)^2 + (y_2-y_1)^2 - d^2 < 0$. Ainsi, puisqu'on a supposé que $(x_2-x_1)^2 - d^2 > 0$, pour une hyperbole, on obtient une différence de carrés, et pour une ellipse, une somme.

Reste à observer les autres cas : si $(x_2-x_1)^2 - d^2 < 0$, on mutiplie l'équation par -1 et on utilise $\sqrt{d^2-(x_2-x_1)^2}$ puis on continue de même. Si $(x_2-x_1)^2 - d^2 = 0$, on commence par y et on continue de même. Enfin si $(x_2-x_1)^2 - d^2 = 0$ et $(y_2-y_1)^2 - d^2 = 0$ (qui n'est possible que pour $d \leq \|u-v\|$ c'est-à-dire dans le cas d'une hyperbole, l'équation devient $8(x_2-x_1)(y_2-y_1)(xy) + \ldots = 0$ que l'on peut aussi écrire

$$(x_2-x_1)(y_2-y_1)[(x+y)^2 - (x-y)^2]/4 + \ldots = 0$$

qui est une différence ce carré (on a bien $|x_2-x_1||y_2-y_1| = d^2 \neq 0$).

Ceci nous conduit à la définition suivante :

Définition 42 *L'équation d'une hyperbole (resp. ellipse) non dégénérée est un polynôme du second degrès à deux variables dont la réduction de Gauss s'écrit comme une différence (resp. somme) de deux carrés.*

Exemple : si on considère l'équation d'hyperbole bien connue, $y = 1/x$, soit $xy = 1$, elle s'écrit aussi $(x+y)^2 - (x-y)^2 = 4$.

Exemple : (Réduction de Gauss complète, y compris les ordres inférieurs) $2x^2 + 7xy + 6y^2 + x + 9y + 3 = 0$. On écrit d'abord tous les termes contenant x, dans un seul carré

$$2(x - 7y/4 + 1/4)^2 - 49y^2/8 - 7y/4 - 1/8 + 6y^2 + 9y + 3 = 0$$

soit

$$2(x-7y/4+1/4)^2-y^2/8+29y/4+23/8=0$$

et on écrit maintenant tous les termes contenant y dans un carré

$$2(x-7y/4+1/4)^2-1/8(y+29)^2+29^2/8+23/8=0$$

et l'on retrouve la différence de 2 carrés. En posant $X=x-7y/4+1/4$ et $Y=y+29$, on a ainsi l'hyperbole $2X^2-1/8(Y)^2+(29^2+23)/8=0$ dont on connait les axes.

5.3 Observation

Par la suite on supposera que les satellites ont pour position $(2,1)$, $(2,0)$ et $(2,-1)$ et que le recepteur gps est en $(1,0)$.

Etablissons léquation de l'hyperbole ayant pour foyer les deux premiers satellites et passant par $(1,0)$. La distance entre l'obervateur et le premier satellite est $\sqrt{2}$ et pour le deuxième 1, soit

$$\sqrt{(x-2)^2+(y-1)^2}-\sqrt{(x-2)^2+(y-0)^2}=\sqrt{2}-1$$

et si l'on effectue les calculs comme dans le paragraphe précédent, on a

$$[(y-1)^2-y^2]^2-2(\sqrt{2}-1)^2(2(x-2)^2+(y-1)^2+y^2)+(\sqrt{2}-1)^4=0$$

soit

$$(-2y+1)^2-2(3-2\sqrt{2})[2(x-2)^2+2y^2-2y+1]+17-12\sqrt{2}=0$$

et en developpant les termes en y

$$-4(3-2\sqrt{2})(x-2)^2+4(-2+2\sqrt{2})(y^2-y)+12-8\sqrt{2}=0$$

soit

$$-4(3-2\sqrt{2})(x-2)^2+8(\sqrt{2}-1)(y-1/2)^2+14-10\sqrt{2}=0$$

(qui est bien l'équation d'une hyperbole).

Et l'équation de la branche passant par l'observateur est

$$y=1/2-\sqrt{10\sqrt{2}-14+4(3-2\sqrt{2})(x-2)^2}/\sqrt{8(\sqrt{2}-1)}$$

et de même (par symétrie) pour les deuxième et troisième satellites

$$y=-1/2+\sqrt{10\sqrt{2}-14+4(3-2\sqrt{2})(x-2)^2}/\sqrt{8(\sqrt{2}-1)}$$

(on peut vérifier que $(1,0)$ et, par symétrie, $(3,0)$ sont solutions des deux équations).

On peut donc faire le tracé avec octave (on supposera la terre un disque de rayon 1 centré en $(0,0)$) :

```
clear

function y=h1(x)
  y=1/2-sqrt(10*sqrt(2)-14+4*(3-2*sqrt(2))*(x-2)^2)/sqrt(8*(sqrt(2)-1))
end

function y=h2(x)
  y=-1/2+sqrt(10*sqrt(2)-14+4*(3-2*sqrt(2))*(x-2)^2)/sqrt(8*(sqrt(2)-1))
end

x=-1:.01:1;
dessus=sqrt(1-x.^2);
dessous=-dessus;

w=-5:.01:5;
plot(w,h1(w),w,h2(w),x,dessus,x,dessous)
axis ([-5, 5, -5, 5], "square")
```

5.4 Résolution numérique approchée

5.4.1 Equation scalaire

Il s'agit de trouver le (les) point d'intersection des deux branches d'hyperboles :

$$y = 1/2 - \sqrt{10\sqrt{2} - 14 + 4(3 - 2\sqrt{2})(x-2)^2}/\sqrt{8(\sqrt{2}-1)}$$

$$y = -1/2 + \sqrt{10\sqrt{2} - 14 + 4(3 - 2\sqrt{2})(x-2)^2}/\sqrt{8(\sqrt{2}-1)}$$

C'est-à-dire de résoudre $f(x) = 0$ avec

$$f(x) = 1 - \sqrt{10\sqrt{2} - 14 + 4(3 - 2\sqrt{2})(x-2)^2}/\sqrt{8(\sqrt{2}-1)} - \sqrt{10\sqrt{2} - 14 + 4(3 - 2\sqrt{2})(x-2)^2}/\sqrt{8(\sqrt{2}-1)}$$

Dichotomie

On peut observer que $f(0) = -0.86689$ et $f(2) = 0.58579$ et utiliser le théorême :

Théoreme 43 *(valeurs intermédiaires) Soit f une fonction continue sur un intervalle $[a, b]$, alors pour tout $c \in [f(a), f(b)]$, il existe au moins un d tel que $f(d) = c$.*

Ce qui conduit à l'algorithme de Dichotomie :

Algorithme 44 *Si $f(a)f(b) < 0$, on calcule $c = (a+b)/2$. Si f(c)=0, fin du calcul. Si $f(a)f(c) < 0$, on remplace b par c, si $f(a)f(c) > 0$ c'est que $f(c)f(b) < 0$, on remplace a par c. Puis on recommence.*

L'algorithme converge puisqu'à l'étape n on a $|b - a| \leq C/2^n$.
Ceci conduit au programme octave suivant

```
clear

function y=f(x)
  y=1/2-sqrt(10*sqrt(2)-14+4*(3-2*sqrt(2))*(x-2).^2)/sqrt(8*(sqrt(2)-1))-(-1/2+
sqrt(10*sqrt(2)-14+4*(3-2*sqrt(2))*(x-2).^2)/sqrt(8*(sqrt(2)-1)));
end

tol=0.01
a=0
b=1.5
c=(a+b)/2
compteur=1

while(abs(b-a)>tol & abs(f(c))>tol)
  if (f(a)*f(c)<0)
    b=c;
  endif
  if (f(a)*f(c)>0)
    a=c;
  endif
  c=(a+b)/2
  compteur=compteur+1
end
c
f(c)
```

Ce qui donne 6 intérations.

Méthode de Newton et sécante

Si f est C^1, on approche la solution de $f(x) = 0$ en remplaçant f par sa tangeante (dessin).

En partant d'un point a, on approche donc $f(x)$ par $f(a) + f'(a)(x - a)$ et on considère donc qu'une approximation de la solution de $f(x) = 0$ est $a - f(a)/f'(a)$ (si $f'(a) \neq 0$).

Algorithme 45 *(Newton) Si f est C^1, on part de a que l'on remplace par $a - f(a)/f'(a)$. Puis on recommence.*

Ceci a pour inconvénient de devoir connaitre la dérivé de f ce qui n'est pas facile en général. Une solution est de remplacer $f'(a)$ par un taux de variation de f :

Algorithme 46 *(Sécante) On part de a et b donnés, que l'on remplace par $a - f(a)(a-b)/(f(a) - f(b))$ et a respectivement. Puis on recommence.*

```
clear

function y=f(x)
  y=1/2-sqrt(10*sqrt(2)-14+4*(3-2*sqrt(2))*(x-2).^2)/sqrt(8*(sqrt(2)-1))-(-1/2+
sqrt(10*sqrt(2)-14+4*(3-2*sqrt(2))*(x-2).^2)/sqrt(8*(sqrt(2)-1)));
end

tol=0.01
a=0
b=1.5
compteur=0

while(abs(b-a)>tol & abs(f(a))>tol)
  if (abs(f(a)-f(b))>0)
    taux=(f(a)-f(b))/(a-b);
    b=a;
    a=a-f(a)/taux;
  else
    a=a+tol;
  endif
  a
  compteur=compteur+1
end
a
f(a)
```

Deux intérations en partant de 0 et $1,5$. Mais cela peut aussi diverger.

5.4.2 Systèmes d'équations

On n'utilise plus ici les équations du polynomiales de degrès 2 des hyperboles mais seulement la différence de distance entre les deux paires de satellites. Il s'agit donc de résoudre un système de deux équations non-linéaires (et non polynomiales).

Dichotomie

Pour une fonction à valeur dans $\mathbf{R}^N$, le théoreme des valeurs intermediaires ne s'applique pas : il est facile de construire un contre exemple où les deux coordonnées ne s'annulent pas en même temps.

Méthode de Newton et sécante

Comme il s'agit d'un sytème, ce n'est plus la dérivée (d'une fonction scalaire) qui peut être utilisée mais la différentielle :

Définition 47 *Une fonction f de $\mathbf{R}^N$ dans $\mathbf{R}^M$ est dite différentiable en $x \in \mathbf{R}^N$ s'il existe D_x linéaire de R^N dans $\mathbf{R}^M$ telle que pour $h \in \mathbf{R}^N$:*

$$\lim_{\|h\|\to 0} \frac{1}{\|h\|}\|f(x+h) - f(x) - D_x(h)\| = 0$$

Dans ce cas on appelle D_x différentielle de f en x notée $Df(x)$. La différentielle est une matrice $M \times N$ et $Df(x)_{ij} = \partial f_i(x)/\partial x_j$ si $x = (x_1, \ldots, x_N)$.

Exemple : $x^2 + y^2$.

La méthode de Newton fonctionne de la même façon que pour une équation scalaire : on approche $f(a+h)$ par $f(a) + Df(a)(h)$. Résoudre $f(a+h) = 0$ devient donc $f(a) + Df(a)(h) = 0$ soit $h = -Df(a)^{-1}f(a)$, et une meilleure approximation que a est donc $a - Df(a)^{-1}f(a)$.

D'où l'algorithme :

Algorithme 48 *(Newton vectoriel) Si f est différentiable avec une différentielle continue, on part de a que l'on remplace par $a - Df(a)^{-1}f(a)$. Puis on recommence.*

Ceci a (toujours) pour inconvénient de devoir connaitre la dífférentielle de f ce qui n'est pas facile en général. Une solution est de remplacer $\partial f_i(a)\partial x_j$ par un taux de variation de f : $(f_i(a) - f_i(a_1, \ldots, a_{j-1}, b_j, a_{j+1}, \ldots, a_N))/(a_j - b_j)$

Algorithme 49 *(Sécante) On part de a et b donnés, on calcule*

$$D_i j = (f_i(a) - f_i(a_1, \ldots, a_{j-1}, b_j, a_{j+1}, \ldots, a_N))/(a_j - b_j)$$

et l'on remplace a et b par $a - D^{-1}f(a)$ et a respectivement. Puis on recommence.

Ceci nous donne le code octave :

```
clear

function z=hyperboles(x)
  z(1,1)=sqrt((x(1,1)-2)^2 + (x(2,1)-1)^2) - sqrt((x(1,1)-2)^2
+ (x(2,1)-0)^2) - (sqrt(2)-1);
  z(2,1)=sqrt((x(1,1)-2)^2 + (x(2,1)+1)^2) - sqrt((x(1,1)-2)^2
+ (x(2,1)-0)^2) - (sqrt(2)-1);
end

xp(1,1)=0.5
xp(2,1)=0.5
hyperboles([xp])

x(1,1)=0.6
x(2,1)=0.6
hyperboles([x])

for i=1:5
dhyp1 = (hyperboles([x(1,1);x(2,1)]) - hyperboles([xp(1,1);x(2,1)]))/(x(1,1) -  xp(1,1));
dhyp2 = (hyperboles([x(1,1);x(2,1)]) - hyperboles([x(1,1);xp(2,1)]))/(x(2,1) -  xp(2,1));

dhyp = [dhyp1,dhyp2]

xp=x;
```

```
x = x - dhyp \ hyperboles([x])

hyperboles([x;x])
end
```

On peut observer de nombreux problèmes de convergence notamment à en cas de points aux voisinages des satellites. Les courbes ne sont pas dérivables en les satellites :

```
clear

function z=hyperboles1(x)
  z=sqrt((x(1,1)-2)^2 + (x(2,1)-1)^2) - sqrt((x(1,1)-2)^2 + (x(2,1)-0)^2) -(sqrt(2)-1);
end

function z=hyperbole2(x)
  z=sqrt((x(1,1)-2)^2 + (x(2,1)+1)^2) - sqrt((x(1,1)-2)^2 + (x(2,1)-0)^2) -(sqrt(2)-1);
end

y=-1:.1:3;
for i=1:41
  for j=1:41
    hyp1(i,j)=hyperboles1([y(i);y(j)]);
  end
end
mesh(y,y,hyp1)
```

Point fixe

Une autre méthode pour traiter le cas des systèmes est de traiter les équations séparement : s'il y a N équations à N inconnues : on peut considerer comme fixes, toutes les inconnues sauf la première et donc résoudre la première équation avec la première coordonnées de x : $f_1(t_1, x_2, \ldots, x_n) = 0$ ce qui donne t_1, puis la seconde avec la deuxième coordonnées de x : $f_2(t_1, t_2, x_3, \ldots) = 0$ ce qui donne t_2 etc. On peut alors reprendre avec la première équation.

Voici le code octave (on utilise ici fsolve par soucis de simplicité) :

```
clear

global t

function z=f1(x,y)
  z=sqrt((x-2)^2+(y-1)^2)-sqrt((x-2)^2+(y-0)^2)-(sqrt(2)-1);
end

function z=f2(x,y)
  z=sqrt((x-2)^2+(y+2)^2)-sqrt((x-2)^2+(y-0)^2)-(sqrt(5)-1);
end

function z=f1y(x)
  global t
  z=f1(x,t);
end

function z=f2y(x)
  global t
  z=f2(x,t);
end
function z=f1x(y)
```

```
  global t
  z=f1(t,y);
end

function z=f2x(y)
  global t
  z=f2(t,y);
end

t=2
for i=1:10
  t=fsolve("f1x",0)
  t=fsolve("f2y",1.5)
end
```

On constate qu'avec la disposition symétrique des satellites ($(2,1)$, $(2,0)$, $(2,-1)$), la méthode oscille sans converger. Si l'on choisit $(2,2)$, $(2,0)$, $(2,-1)$ la méthode diverge et si l'on choisit $(2,1)$, $(2,0)$, $(2,-2)$ la méthode converge.

Ceci n'est pas très robuste.

5.4.3 fsolve

On pouvait directement utiliser $fsolve$ pour résoudre les système :

```
clear

function z=hyperboles(x)
  z(1)=sqrt((x(1)-2)^2 + (x(2)-1)^2) - sqrt((x(1)-2)^2 + (x(2)-0)^2) - (sqrt(2)-1);
  z(2)=sqrt((x(1)-2)^2 + (x(2)+1)^2) - sqrt((x(1)-2)^2 + (x(2)-0)^2) - (sqrt(2)-1);
end

fsolve ("hyperboles",[1,1])
```

Bien programmée par octave cette commande est bien plus robuste.