Ecole Nationale Supérieure
des Télécommunications

Master ESCO
Parcours STN

Décodage des réseaux de points

par

MROUEH Lina lina.mroueh@enst.fr
BADR Maya maya.badr@enst.fr

Encadrant M. BELFIORE Jean-Claude.

Février 2006

Plan

Plan 2
1. Introduction 3
2. Préliminaire 3
2.1 - Réseau de points-Lattice 3
2.2 - Matrice génératrice d'un réseau de points 3
2.3 - Représentation en réseau de points des systèmes MIMO 3
2.4 - Closest–point 4
2. 5 - Région de Voronoi 4
3. Algorithmes de décodage 5
3.1 - Décodeur sphérique (SD) 5
3.1.1 – Principe du décodeur sphérique 5
3.1.2 - Organigramme de l'algorithme 7
3.1.3 - Exemple du déroulement de l'algorithme 8
3.2 - Algorithme de Shnorr-Euchner (SE) 8
3.2.1 – Principe du Shnorr-Euchner 8
3.2.2 - Organigramme de l'algorithme 11
3.2.3– Implémentation Matlab 13
3.3 - Comparaison du SD et du SE 16
3.4 – Complexités et performances 17
Références 18

1. Introduction

Le but de ce projet est d'utiliser la représentation des codes en réseaux de points pour le décodage. Les décodeurs de réseaux de points les plus connus sont le décodeur sphérique et le Schnorr-Euchner. Les deux algorithmes qu'on va détailler dans ce projet se reposent sur le critère ML qui consiste à maximiser la vraisemblance.
Sur un canal gaussien AWGN, le critère ML consiste à minimiser la distance entre le point reçu et le point décodé ce qui équivalent à trouver le point du réseau le plus proche du point reçu.

Dans ce projet, on va commencer par définir des notions du réseau de points, du problème de closest point. On présente par la suite les deux algorithmes de décodage : le sphère decoding et l'algorithme du Schnorr-Euchner.

2. Préliminaire

2.1 - Réseau de points-Lattice

Un **réseau de points** Λ est un sous-groupe discret de l'espace euclidien R^n, de rang $p \leq n$.

$$\Lambda = \{ \sum_{i=1}^{p} a_i v_i ; a_i \in Z \}$$

C'est donc un ensemble engendré par p vecteurs $\mathbf{v}_1, \mathbf{v}_2 \ldots \mathbf{v}_p$ de R^n, la famille de ces vecteurs constitue une **base du réseau de points, de dimension p.**

2.2 - Matrice génératrice d'un réseau de points

La **matrice génératrice** du réseau de points est une matrice à entrées réelles dont les lignes sont linéairement indépendantes sur R.
Le réseau de points de rang p, engendré par une matrice génératrice notée **M** dont les colonnes sont les vecteurs $\mathbf{v}_1, \mathbf{v}_2 \ldots \mathbf{v}_p$, sera noté Λ_M.

Ce réseau pourra être vu comme une transformation linéaire appliquée au réseau Z^p, en effet tout vecteur **x** de ce réseau s'écrit sous la forme **x = z.M** avec **z' =** $(z_1, z_2, ..., z_p) \in Z^p$.

$$\Lambda_M = \{ \mathbf{M.z} ; \mathbf{z} \in Z^p \}$$

2.3 - Représentation en réseau de points des systèmes MIMO

Considérons le système à n_t antennes à l'émission et n_r à la réception, le mot de code **r** reçu s'écrit, pour $n_t = n_r = n$, sous la forme :

$$\mathbf{y}_n = \mathbf{H}_{n*n} \mathbf{x}_n + \mathbf{z}_n$$

Pour obtenir la représentation en réseau de points d'un système MIMO, on doit séparer les parties réelles et imaginaires des vecteurs et de la matrice **H** et on obtient :

$$\mathbf{y'} = [\ \Re(\mathbf{y}^T), \Im(\mathbf{y}^T)]$$

$$= [\ \Re(\mathbf{x}^T), \Im(\mathbf{x}^T)]\ .\ \begin{bmatrix} \Re(H^T) & -\Im(H^T) \\ \Im(H^T) & \Re(H^T) \end{bmatrix} + [\Re(\mathbf{z}^T), \Im(\mathbf{z}^T)]$$

$$= \mathbf{u}\,\mathbf{M}_H + \mathbf{z'} \quad \text{avec } \mathbf{u} \in Z^{2n} \text{ et } \mathbf{z'} \in R^{2n}$$

La probabilité d'avoir deux colonnes de **H** dépendantes est nulle du fait que les coefficients de **H**, correspondant aux évanouissements entre les antennes émettrices et les antennes réceptrices, sont indépendants. La matrice est donc de rang plein, de même la matrice $\mathbf{M}_H$ est de rang plein = 2n.
Le réseau de points équivalent représentant le système MIMO est donc de dimension 2n, de matrice génératrice $\mathbf{M}_H$ **.**

Si on utilise un code spatio-temporel **ST**, en supposant **H** fixe durant la transmission d'un mot de code (canal quasi-statique) :

$$\mathbf{Y}_{n*T} = \mathbf{H}_{n*n}\ \mathbf{X}_{n*T} + \mathbf{Z}_{n*T}$$

où T est la longueur temporelle du code, **X** étant le mot de code à transmettre, **H** la matrice de transfert du canal et **Z** le bruit gaussien.

2.4 - Closest–point

Pour un réseau de points Λ donné et un point $\mathbf{y} \in \mathrm{R}^n$ reçu, le point le plus proche est un vecteur $\hat{c}$ vérifiant :

$$\| \mathbf{y} - \hat{c} \| \leq \| \mathbf{y} - \mathbf{c} \| \quad \forall\, \mathbf{c} \in \Lambda$$

où $\| . \|$ correspond à la distance euclidienne.

On peut ramener le problème du closest point au critère ML sur un canal AWGN :

$\mathbf{Y} = \mathbf{H}.\mathbf{X} + \mathbf{Z}$, qui consiste à chercher $\hat{X} = \arg(\min_{X \in C} \| Y - H.X \|^2)$,

avec c = H.X ; X étant un point d'une constellation $\in Z^n$.
H matrice qui génère le réseau des points.

2. 5 - Région de Voronoi

La région de Voronoi d'un point **c** d'un réseau de points Λ est l'ensemble de tous les vecteurs de R^n qui peuvent être décodés en **c** :

$$\nu(\mathbf{c}) = \{\mathbf{y} \in \mathrm{R}^n\ ; \| \mathbf{y} - \mathbf{c} \| \leq \| \mathbf{y} - \mathbf{c'} \| \quad \forall\, \mathbf{c'} \in \Lambda\ \}$$

3. Algorithmes de décodage

La recherche du point le plus proche étant la base du décodage, plusieurs méthodes ont été développées pour résoudre ce problème qui dépend de la structure du réseau de points et qui devient alors plus simple quand le réseau est plus structuré. Dans notre cas on cherche à résoudre le problème du point le plus proche dans le cas général.

Une approche commune à ce problème est d'identifier une certaine région de R^n dans laquelle le point optimal pourra se trouver, tester ensuite tous les points du réseau appartenant à cette région et réduire sa taille dynamiquement jusqu'à trouver le point rechercher.
Il existe deux principales branches relatives à la recherche du point le plus proche dans la littérature. La première est celle de **Pohst** considérant comme région de recherche une hyper sphère et la deuxième est celle de **Kannan** considérant comme région de recherche un parallélépipède rectangle. La méthode de **Kannan** apparaît comme une méthode théorique alors que celle de **Pohst** comme une méthode pratique. Pour cela on s'intéressera à cette dernière méthode dans laquelle deux stratégies de recherche ont été développées correspondants aux deux décodeurs : décodeur par sphères **SD** et décodeur de Schnorr-Euchener **SE**.

Concernant le **SD,** le principe est de chercher le point le plus proche dans une sphère de rayon donné, centrée sur le point reçu. Si on ne trouve aucun point le rayon est agrandi avant de recommencer la recherche. Le parcours des points se fait de la surface de la sphère vers l'intérieur.

Le principe de l'algorithme **SE** est d'effectuer des projections successives sur les hyperplans du réseau de points afin de trouver le point le plus proche. Dans ce cas le parcours des points dans la sphère se fait du centre vers l'extérieur.
Nous aborderons, dans la suite, les principes de ces deux algorithmes en détails.

3.1 - Décodeur sphérique (SD)

3.1.1 – Principe du décodeur sphérique

L'équation qui donne la représentation en réseaux d'un système MIMO est donnée par :

$$\mathbf{Y}^T = \mathbf{s}^T \mathbf{M}^T + \mathbf{w}^T$$

On va considérer le réseau de points engendré par la matrice M, donc $\mathbf{x} = \mathbf{s}^T . \mathbf{M}^T$ sera le point du réseau émis sur le canal gaussien s étant un point d'une constellation$\in Z^n$.
Le bruit W est un bruit gaussien $N(0, \sigma^2)$.

Le décodeur choisit le mot de code qui minimise la métrique $\| \mathbf{y}^T - \mathbf{x} \|$, soit :

$$\min_{x \in \Lambda} \| \mathbf{y}^T - \mathbf{x} \| = \min_{w \in y - \Lambda} \| \mathbf{w} \|$$

Le problème revient donc à chercher le point le plus proche de z dans le réseau de points translaté $\mathbf{y} - \Lambda$

On définit alors les points **x, y** et **w** par :

$$\mathbf{x} = \mathbf{s}^T . \mathbf{M}^T, \qquad \mathbf{u} = \mathbf{s}^T \in Z^n$$
$$\mathbf{y} = \rho . \mathbf{M}, \qquad \rho = (\rho_1, ..., \rho_n) \in R^n$$
$$\mathbf{w} = (\rho - \mathbf{u}) . \mathbf{M} = \xi . \mathbf{M}, \qquad \xi = (\xi_1, ..., \xi_n) \in \mathrm{R}^n$$

L'algorithme de décodage sphérique consiste à chercher parmi les points de la lattice (réseaux de points) les points qui se trouvent à l'intérieur d'une sphère de rayon $\sqrt{C}$ centré sur le point reçu **y.**

Cette sphère sera transformée en une ellipsoïde centré à l'origine du nouveau repère dont les coordonnées sont définis par ξ.
Cela se traduit par l'inéquation suivante :

$$\|\mathbf{w}\|^2 = \mathrm{Q}(\xi) = \xi \mathbf{M}\mathbf{M}^T \xi^T = \xi \mathbf{G} \xi^T = \sum_{i=1}^{n} \sum_{j=1}^{n} g_{ij} \xi_i \xi_j \le C$$

Après avoir procédé à une décomposition QR de la matrice génératrice, on obtient à partir de cette inéquation pour chaque composante u_i, une borne inférieure et une borne supérieure.

Le calcul est détaillé dans l'article [2], on aura alors:

$$-\sqrt{\frac{C}{q_{nn}}} + \rho_n \le u_n \le \sqrt{\frac{C}{q_{nn}}} + \rho_n$$

$$\mathrm{b}_{\mathrm{inf},i} = -\sqrt{\frac{T_i}{q_{ii}}} + \mathrm{S}_i \le \mathrm{u}_i \le \sqrt{\frac{T_i}{q_{ii}}} + \mathrm{S}_i = \mathrm{b}_{\mathrm{sup},i}$$

où : $\mathrm{S}_i = \rho_i + \sum_{l=i+1}^{n} q_{il} \xi_l, \quad i = 1,...,n$

$$\mathrm{T}_{i-1} = \mathrm{C} - \sum_{l=i}^{n} q_{ll} (\xi_l + \sum_{j=l+1}^{n} q_{lj} \xi_j)^2 = T_i - q_{ii} (S_i - u_i)^2$$

$$q_{ii} = r_{ii}^2 \ , i = 1,...,n \quad \text{et} \quad q_{ij} = \frac{r_{ij}}{r_{ii}} \ , i = 1, ...,n \ ; \ j = i+1,...,n$$

Lorsque un point du réseau de points est trouvé, la distance au carrée qui le sépare du centre (le point reçu) est donnée par :

$$\hat{d}^2 = C - T_1 + q_{11} (S_1 - u_1)^2$$

Afin de trouver le point le plus proche, l'algorithme parcourt l'intervalle $I_i = [b_{\mathrm{inf},i}, b_{\mathrm{sup},i}]$.

Pour chaque combinaison de composantes u_i trouvée, la distance $\hat{d}^2$ est évaluée. Chaque point trouvé vérifiant $\hat{d}^2 \leq C$ est stocké.
La recherche continue jusqu'à ce que tous les points se trouvant dans la sphère soient parcourus.

L'algorithme de recherche implique que le rayon de la sphère ainsi que les bornes $b_{\text{inf},i}, b_{\text{sup},i}$ pour $i = 1,\ldots,n$ soient mis à jour dynamiquement au cours de la recherche chaque fois qu'un point est trouvé, i.e. $C = \hat{d}^2$.

3.1.2 - Organigramme de l'algorithme

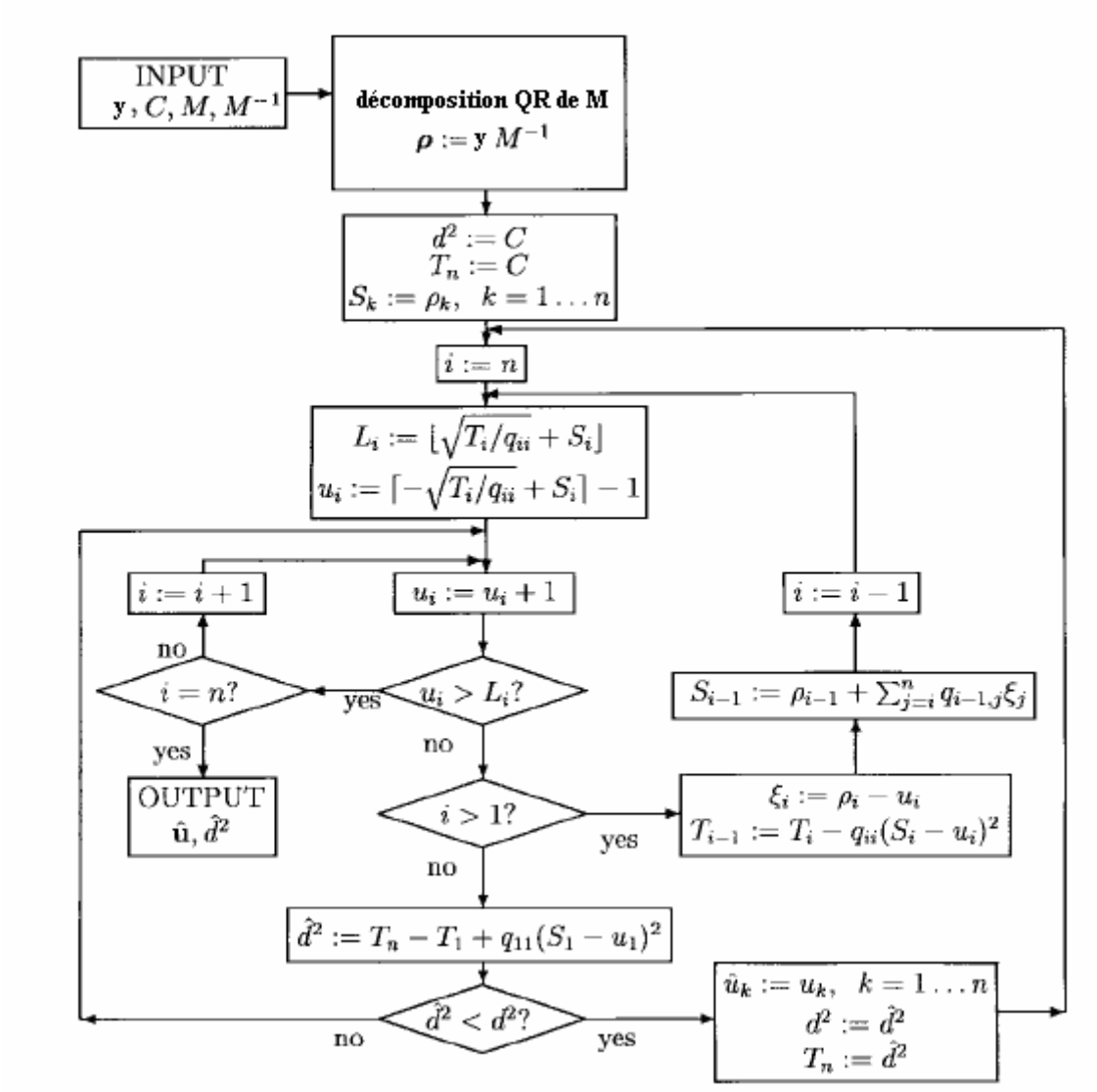


Figure 1 : Organigramme du décodeur par sphères.

3.1.3 - Exemple du déroulement de l'algorithme

Cas d'un réseau de dimension 2 où on aura 2 composantes u_1 et u_2 :

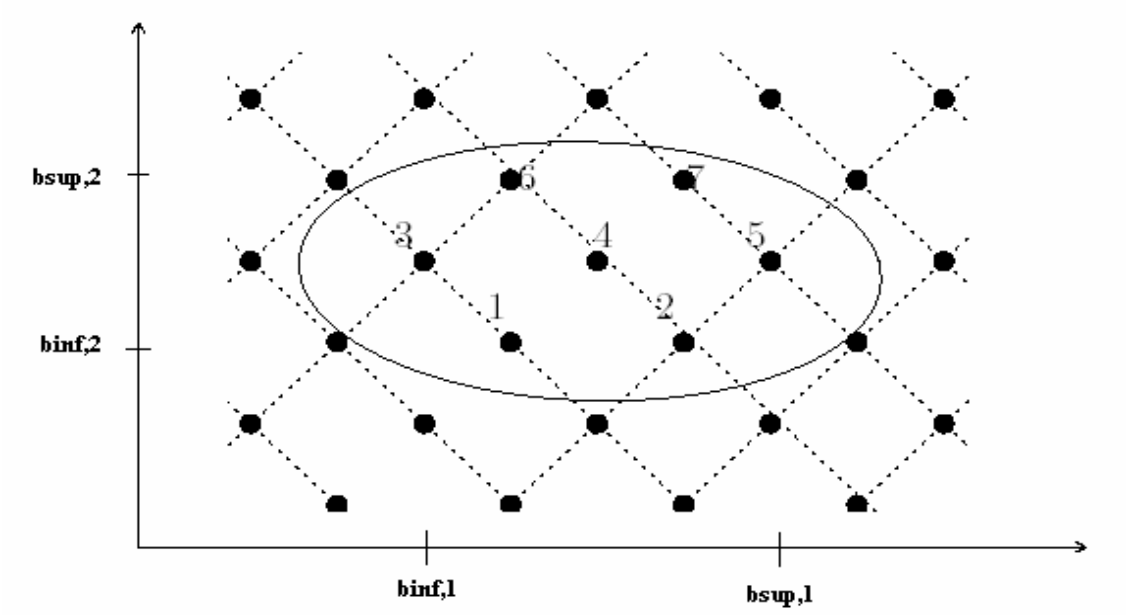


Figure 2 : Exemple SD des points du réseau de points Z^2

En un premier lieu, on cherche à borner u_2, et ensuite u_1. Une fois le point **u** est trouvé, on calcule la distance qui le sépare du point **y**. Celle-ci est comparée au rayon de la sphère :
Si $\hat{d}^2 < C \Rightarrow$ Il faut ajuster le rayon de la sphère et par suite les bornes ; Sinon, on passe au point suivant (sur la même ligne) tant que u1< $b_{sup,1}$, si cette condition n'est pas satisfaite on passe à la ligne juste au dessus (à condition qu'on ne dépasse pas la borne supérieure).

Nous constatons alors que les points sont parcourus de bas en haut, et de gauche à droite. Cette algorithme s'arrête lorsque c'est plus possible ni de passer au point suivant sur la même ligne, ni passer à la ligne au dessus.
Dans ce cas, on n'aura dans la sphère que le point recherché qui est le point le plus proche au point reçu.

3.2 - Algorithme de Shnorr-Euchner (SE)

3.2.1 – Principe du Shnorr-Euchner

Reprenons l'équation qui donne la représentation en réseaux d'un système MIMO:

$$\mathbf{Y= M.s+ w}$$

Pour appliquer l'algorithme de Shnorr-Euchner SE pour décoder les réseaux de points nécessite une base triangulaire du réseau. Pour cela, nous allons procéder à une décomposition QR de la matrice génératrice du réseau de points M.
M =QR où R est une matrice triangulaire supérieure et Q est une matrice orthogonale.
On aura alors :

$$\mathbf{y= QR\ s\ + w}$$

On multiplie alors les membres de l'égalité par Q, on obtient un réseau de points équivalent, et l'équation devient alors :

$$\mathbf{x = Q^T . y\ = R . s + Q^T\ .w\ _1 = R. s + w_1}$$

Vu la forme triangulaire de la matrice génératrice des points, on peut voir le réseau de points généré par $\mathbf{R}_1$ comme étant une superposition de couches.

En effet, la matrice peut être écrite sous la forme : $\mathbf{R}_1 = \begin{bmatrix} R_2 \\ r_n \end{bmatrix}$; $\mathbf{R}_2$ étant la matrice constituée des n-1 lignes de $\mathbf{R}_1$.

Ainsi, le réseau de points en dimension n peut être vue comme une réunion d'infinité de réseau engendré $\mathbf{R}_2$ par de dimension n-1, appelées couches.

Le vecteur $\mathbf{r}_n = \mathbf{r}_{//} + \mathbf{r}_{\perp}$ avec :

$\mathbf{r}_{//} = (r_{n1}, r_{n2}, \ldots, r_{n,n-1}, 0)$ est dans l'espace engendré par $\mathbf{R}_2$

$\mathbf{r}_{\perp} = (0, 0, \ldots, 0, r_{n,n})$ est orthogonale à toutes les couches.

Le principe de l'algorithme SE est de chercher le point le plus proche dans une hyper sphère centré sur le point reçu.

Soit $\mathbf{y} \in R^n$ le point reçu à décoder. L'algorithme SE permet de déterminer le point $\hat{y}$ appartenant au réseau de points générés par $\mathbf{R}_1$ le plus proche de **y**.
La première étape de cet algorithme consiste à trouver le **Babai Point**.

Recherche du Babai Point

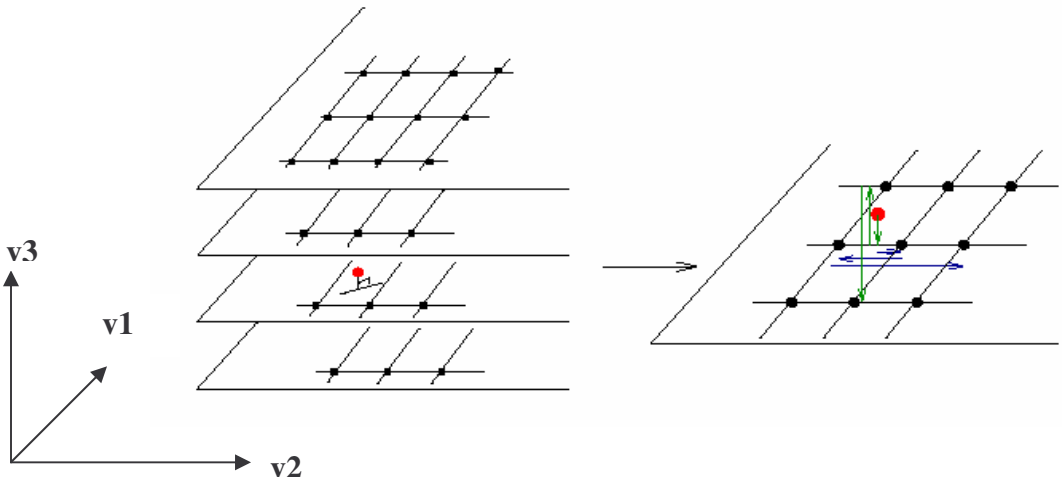


Figure 3 : Projections successives sur les hyperplans

Le Babai point est le premier point trouvé par l'algorithme.
Le vecteur y reçu possède les composantes $\rho_1, \rho_2, ..., \rho_n$ dans l'espace engendré par $\mathbf{R}_1$ tel que :

$$\mathbf{y} = \rho \,.\, \mathbf{R}_1$$
$$\rho = (\rho_1, ..., \rho_n) \in R^n$$

Par suite,

$$y_i = \rho_i . r_{i,i} + \sum_{j=i+1}^{n} \rho_j . r_{i,j}$$

Pour trouver le point de Babai par le calcul, on projette alors le vecteur y reçu sur $\mathbf{r}_\perp$ (i.e v3 sur la figure), et on choisit alors la couche la plus proche (z = u3 $\in Z$).

Soit, $\hat{u}_n = \dfrac{y.r_\perp}{// r_\perp //^2} = \dfrac{y_n}{r_{nn}} = \rho_n$ la projection du vecteur y sur $\mathbf{r}_\perp$, on détermine par la suite la couche la plus proche du point projeté, tel que :

$$u_n = \left\langle \hat{u}_n \right\rangle$$

Où $\langle z \rangle$ est l'entier le plus proche de z $\in \Re$.

La distance de y à la couche t_n est donné par :

$$d_n = / t_n - \hat{u}_n / . // r_\perp //$$

Pour calculer $\hat{u}_{n-1}$, il faut considérer le réseau de points en dimension n-1 engendré par $\mathbf{R}_2$ (espace v1,v2 sur la figure) on va utiliser la décision qu'on a fait sur $\hat{u}_n$.

En utilisant la décision sur u_n, on déduit

$$\rho_{n-1} = \frac{1}{r_{n-1,n-1}} (y_{n-1} - u_n . r_{n,n-1}) .$$

$$u_{n-1} = \left\langle \rho_{n-1} \right\rangle \text{ et } d_{n-1} = \left[u_{n-1} - \rho_{n-1} \right] r_{n-1,n-1}$$

Par récurrence, on pourra alors calculer toutes les composantes u_i du Babai point.

La distance Euclidienne entre le Babai Point et le point reçu est donc égale à :

$$d^2 = \sum_{i=1}^{n} d_i^2$$

Ce point n'est pas nécessairement le point le pus proche du point observé.
Il dépend à la fois de y, de la lattice et de la base qui n'est pas nécessairement orthogonale utilisé pour représenter la lattice.

Le SE consiste alors à chercher le point le plus proche dans une hypersphère centré sur le point reçu et de rayon la distance du point reçu au Babai Point

En partant du Babai Point on doit parcourir tous les points qui sont à l'intérieur de la sphère associé au réseau de points. On doit alors tourner autour de chaque composante du Babai point en zigzaguant. Le changement de sens au cours de zigzag est déterminé par le signe de d_i.

3.2.2 - Organigramme de l'algorithme

Dans cet algorithme, k est la dimension de la couche courante.

Cas A : A chaque fois que l'algorithme trouve une composante u_k, il passe à la couche k-1, à condition que la distance ne dépasse pas la distance minimale courante.

Cas C : Dans le cas où la distance entre le point reçu et le point décodé dépasse la distance minimale courante, on passe à la couche supérieure en effectuant un step en avant qui n'est autre que le zigzag. Si la couche examinée était égale à n, donc le point en mémoire sera le point décodé.

Cas B : Dans ce cas, l'algorithme a franchit tous les layers et on arrive à trouver un point de la lattice pour lequel la distance est inférieure à la distance minimale courante.
Ce point de réseau sera stocké, et la distance sera mise à jour.

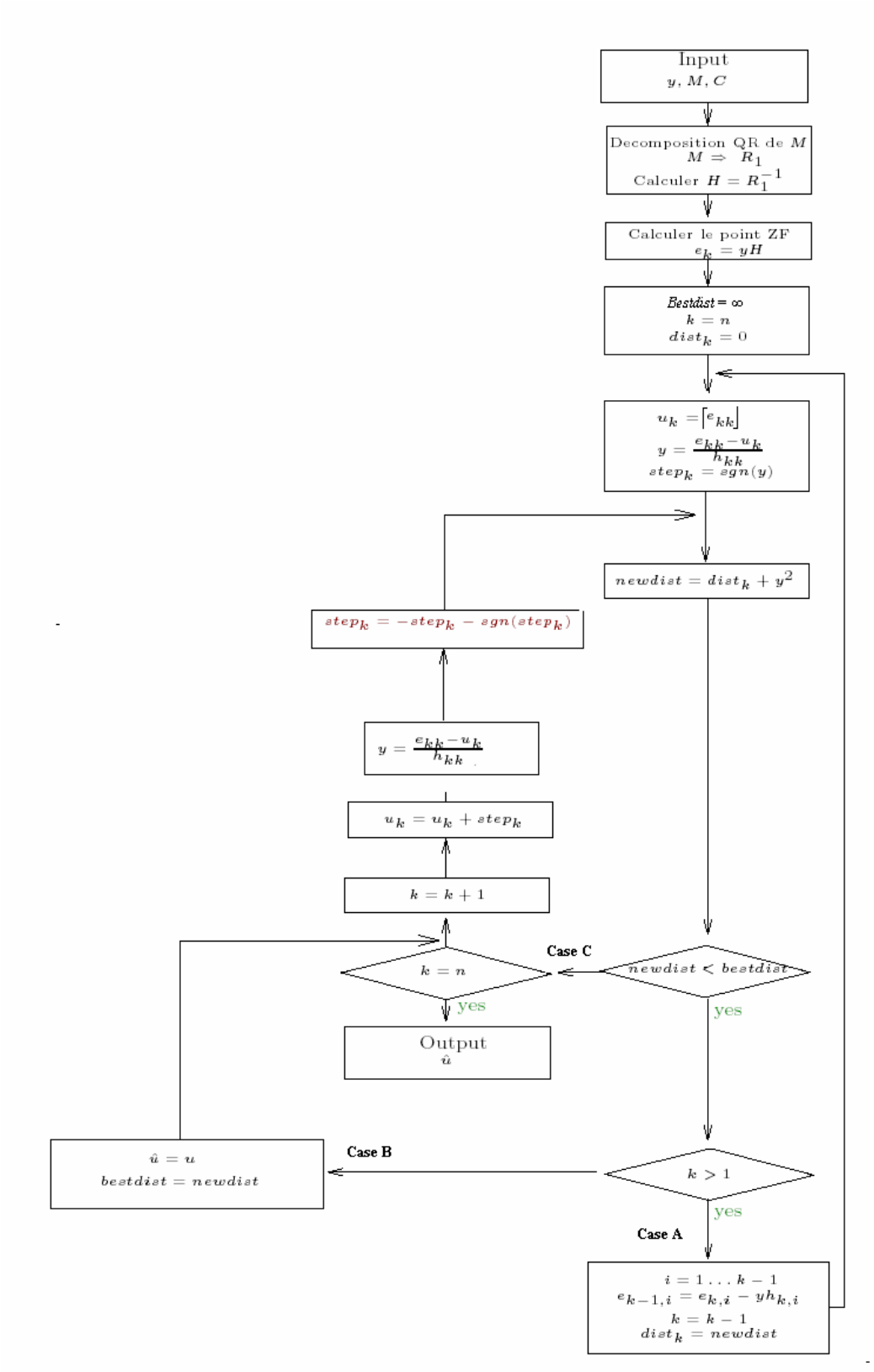


Figure 4 : Organigramme du Schnorr-Euchner

3.2.3– Implémentation Matlab

Soit un réseau de point engendré par une matrice **G triangulaire inférieure** dont les éléments de la diagonale sont **réels positifs**.
Soit y un point $\in R^n$ le point à décoder $\Lambda(G)$; Le SE(x,H) avec $\mathbf{H = G^{-1}}$ permet de trouver le point u tq c =x.G de $\Lambda(G)$ est le plus proche de y.
Soit en reformulant le problème d'une autre façon, on peut dire que le SE cherche le $c \in \Lambda(G)$ qui minimise $\| y - c \|$; Le SE nous donne x tq c = G.x.

Si l'on veut appliquer cet algorithme dans le cas de **décodage ML**, on considére le modèle $\mathbf{Y}_{n*1} = \mathbf{M}_{n*n}.\mathbf{X}_{n*1} + \mathbf{Z}_n$ (1). Il faudra aussi minimiser $\| y - M.x \|$
Il faut alors supposer que c'est M qui est la matrice génératrice ici. Mais le problème est que M n'est pas nécessairement triangulaire inférieure
Solution : **Décomposition QR**. avec $M = Q_1 R$ où Q_1 est une matrice orthonormale càd elle satisfait $Q_1 . Q_1^T = I$.

Comment s'assurer que tous les éléments de la diagonale soient positifs ?

La décomposition QR ne prévoit pas si les éléments de la diagonale sont positifs ou pas.
Pour que l'on puisse bien appliquer SE(x,H), il faudra bien s'assurer que les éléments de la diagonales soit positif, pour cela :

- Il faut multiplier par -1 toutes les colonnes de R dont l'élement de la diagonale est négative.
- Multiplier par -1 les lignes de Q_1 correspondantes.

On désignera dans la suite par $G3 = R^T$ et par $Q = Q_1^T$ donc, $M^T = G3.Q$

Or $Y = M.X + Z \Rightarrow Y^T = X^T.M^T + Z^T$;
Par suite : $Y^T = X^T G3.Q + Z^T \quad \Rightarrow \quad Y^T Q^T = X^T.R^T + Z^T Q^T$

Soit $Y_1^T = Y^T Q^T = X^T.R^T + Z^T Q^T$

Donc comme paramètre pour l'algorithme SE(x1,H) , on aura :
$x1 = Y_1^T = Y^T Q^T$ et $G = R \Rightarrow \mathbf{H = G3^{-1}}$

x = SE (y, H) point émis

Donc si l'on génère un bruit gaussien par la fonction varn et on variant la variance sigma(càd on varie le SNR tel que sigma = $10^{-\frac{Psig\_db - SNR(db)}{10}}$), en supposant que la puissance d'émission est constante(= 0db), on arrive à tracer la probabilité d'erreur en fonction de SNR

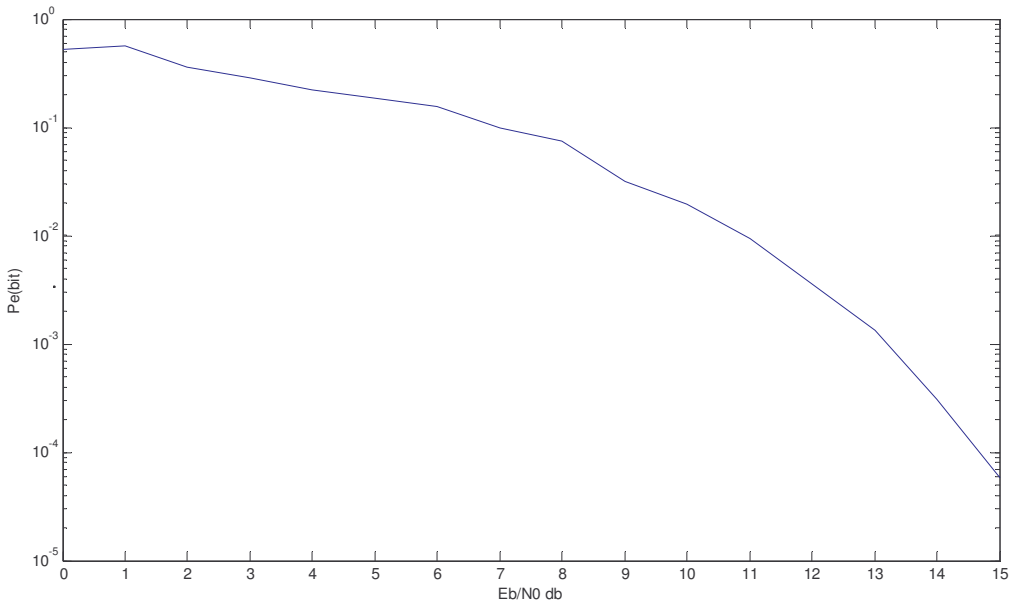
On remarque bien que dans l'algorithme SE, la probabilité d'erreur diminue en fonction du SNR.

```
%-------------------------------------------------------------------
%--                                                               --
%--              Implémentation de l'algorithme                   --
%--                   Schnorr-Euchnerr                            --
%--                   Opt = SE(x,H)                               --
%--           Opt- Point de la lattice le plus proche de x        --
%--           x - Point reçu                                      --
%--           H - Matrice triangulaire inférieure                 --
%--                à coefficients positifs                        --
%--                                                               --
%-------------------------------------------------------------------

function opt = SE(x,H)
bestdist = inf;
n = size (H, 1);
k = n;
dist(k)= 0;
e(k,:) =x*H ;
u(k)= round(e(k,k));
y = 1/abs(H(k,k))*(e(k,k)-u(k));

if (y == 0)
  step(k) = -1;
else
  step(k)=sign(y);
end;

while(1)
    newdist = dist(k)+ y*y;
    if newdist < bestdist
        if k > 1
            for i = 1: k-1
                e(k-1,i)=e(k,i)-y*H(k,i);
            end
            k = k-1;
            dist(k)=newdist;
            u(k)= round(e(k,k));
            y = 1/abs(H(k,k))*(e(k,k)-u(k));
            if (y == 0)
                step(k) = -1;
            else
                step(k)=sign(y);
            end
        else
            output = u;
            bestdist=newdist;
            k=k+1;
            u(k)=u(k)+step(k);
            y = 1/abs(H(k,k))*(e(k,k)-u(k));
            step(k)= -step(k)-sign(step(k));
        end
    else
        if k == n
            opt = output ;
            break;
        else
            k=k+1;
            u(k)=u(k)+step(k);
            y = 1/abs(H(k,k))*(e(k,k)-u(k));
            step(k)=-step(k)-sign(step(k));
        end
    end
end
```

```
%-------------------------------------------------------------%
%--              Modélisation pour le canal MIMO            --%
%--                 Y - Vecteur complexe                    --%
%--      M - Matrice génératrice à coefficient réels.       --%
%--                   Y = M.X +Z                            --%
%--       Z Bruit Gaussien de variance sigma                --%
%--                                                         --%
%-------------------------------------------------------------%

clear;

% Input des données et paramètres
pas = 0.1;

% Entrer le paramètre X pour lequel vous voulez faire le test
X = [3 2 5 0];

%Puissance du signal (1mw -> 0db)
Psig_db = 0;

% Matrice et décomposition en QR
M = [-1 -1 0 0 ; 1 -1 0 0; 0 1 -1 0 ; 0 0 1 -1];
[Q1 R] = qr(M);
G3= R';
Q = Q1';
M1 = G3*Q;

% Il faut que les h[k,k] soient tous >0 sinon il faut qu'on
% multiplie par -1 les colonnes ou l'élément de la diagonale est
% négative et par -1 la colonne de Q correspondante.

for j = 1 :size(M,1)
    if G3(j,j)<0
        G3(:,j)=G3(:,j)*(-1);
        Q(j,:)=Q(j,:)*(-1);
    end;
end;

H = inv(G3);
SNR = 0:pas:25;
ind = 0;

% Puissance du signal = 1mw ; SNR = - sigma_db;
for SNR_count = 0:pas:25

    ind =ind+1;
    count =0;
    for i=1:50
    % Le point reçu Y = Bruit Gaussien
    sigma = 10^(Psig_db - SNR_count/10);
    Y = X* G3*Q + sigma*randn(1,4);

    % Fonction SE(X, H) avec Re_Y = Q'.Y et Im_Y = Q'.Y
    Y1 =  Y * Q';

    % Decodé = SE(X, R)
    X_decoded = SE(Y1 , H);

    if [X_decoded ] == X
        count =count +1;
    end
    end
    pe(ind) = 1 - count/50;

 end;

 disp ('Pour un Y recu bruité :')
 Y
 disp ('correspondant à un X du réseau de point engendré par M:')
```

```
X

disp ('Lorsque SNR devient grand, en appliquant SE')
X_decoded

plot(SNR,pe)
```

```
Pour un Y recu bruité (Bruit blanc Gaussien) :
Y =
   -4.9996    1.0009   -3.0045    5.0002
correspondant à un X du réseau de point engendré par M:
X =
     3     2     5     0
Lorsque SNR devient grand, en appliquant SE
X_decoded =
     3     2     5     0
```



On remarque que lorsque SNR dépasse un certain seuil (= 15 dB), la probabilité d'erreur devient presque nul.

3.3 - Comparaison du SD et du SE

Dans cette partie, nous nous proposons de comparer les deux algorithmes de décodage des réseaux de points, SD et SE, qu'on vient de présenter en se basant sur la stratégie de recherche.

Nous avons vu que les deux algorithmes parcourent les points du réseau se trouvant à l'intérieur d'une sphère centrée sur le point reçu mais avec des stratégies différentes.

Stratégies de parcours :

Dans le cas du SD, le parcours des points se fait de la surface de la sphère vers l'intérieur. Par contre dans le cas du SE, l'idée consiste à tourner autour de chaque composante du Babai point en zigzaguant, de l'intérieur de la sphère vers l'extérieur.

Rayon de la sphère :

L'astuce de base du SD est le choix du rayon de la sphère duquel dépend la convergence de l'algorithme. Ce rayon doit être nécessairement initialisé au début de l'algorithme. Il y a donc un compromis du fait que pour gagner du temps on a intérêt de choisir un petit rayon mais dans ce cas on risque de ne pas trouver notre point rechercher dans la sphère.
Par ailleurs, dans le cas du SE et avant de trouver le Babai point, on peut choisir le rayon initial de la sphère égal à l'infini ce qui assure la convergence du SE.

3.4 – Complexités et performances

Du point de vue pratique, pour étudier et comparer la complexité du SE et SD, on doit étudier la complexité des deux phases précodage et recherche composant les deux algorithmes.

Les résultats de cet étude, accomplie dans [4], nous permet de dire que les phases de précodage du SD et SE aboutissent pratiquement à la même complexité. La complexité des phases de recherche, mesurée par le temps de calcul, est déterminée en fonction du nombre d'antennes à l'émission par simulation [4]. On en déduit que le SE dispose d'une phase de recherche plus rapide que celle du SD.

La complexité totale étant la somme des complexités des deux phases, nous concluons ainsi que la complexité totale de l'algorithme SE est inférieure à celle du SD.

Références

[1] **Closest Point Search in Lattices.**
Erik Agrell, Thomas Eriksson, Alexander Vardy, and Kenneth Zeger.

[2] **A Universal Lattice Code Decoder for Fading Channels**
Emanuele Viterbo, *Member, IEEE*, and Joseph Boutros, *Member, IEEE*

[3] **Lattice Code Decoder for Space-Time Codes**
Oussama Damen, Ammar Chkeif, and Jean-Claude Belfiore

[4] **Nouvelles constructions algébriques de codes spatio-temporels atteignant le compromis ”multiplexage-diversité” (chapitre 4)**
Ghaya Rekaya-Ben Othman