

Projet de Maths

Programme Scilab

```
disp(modulo(M+N,2),"M+N=")
disp(modulo(M*N,2),"M.N=")
```

$\text{modulo}(x,2)$: « Reste de la division de x par 2 »

Lorsque x est pair, $\text{modulo}(x,2)=0$

Lorsque x est impair, $\text{modulo}(x,2)=1$

De plus $\text{modulo}(x+y,2)$: « Reste de la division de $x+y$ par deux »

$\Rightarrow 0+0=0$, 0 est pair donc $\text{modulo}(0+0,2)=0$

$\Rightarrow 0+1=1$, $1+0=1$, 1 est pair donc $\text{modulo}(1+0,2)=1$

$\Rightarrow 1+1=2$, 2 est pair donc $\text{modulo}(1+1,2)=0$

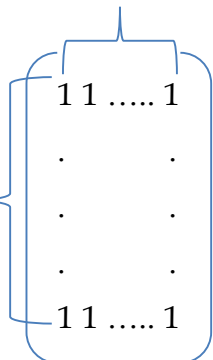
Toutes les matrices de ce sujet appartiennent à $(\mathbb{Z}/2\mathbb{Z})^n$: « Tous les éléments de la matrice sont les restes de la divisions de chaque élément par 2 »

```
P=[[0, 1, 1];[ 1, 1, 0]; ones(2,3)]
G=[eye(3,3);P]
```

Ones(x,y)=

y fois

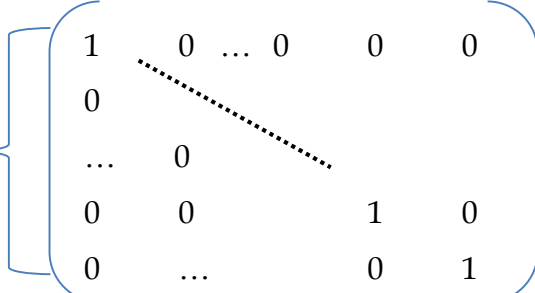
x fois



The diagram shows a matrix of ones. A horizontal bracket above the matrix is labeled 'y fois'. A vertical bracket to the left of the matrix is labeled 'x fois'. The matrix is represented as a grid of 1s and dots.

Eye(x,x)= I_x

x fois



The diagram shows an identity matrix. A vertical bracket to the left of the matrix is labeled 'x fois'. The matrix is represented as a grid of 1s and 0s, with a diagonal line of dots indicating the identity elements.

$$P_{=}$$

$$G =$$

```
function y=f(M); y=modulo(G*M,2); endfunction
```

On défénis la fonction f tel que $f(M)=G^*M$ (sur $\mathbb{Z}/2\mathbb{Z}$ biensur)

```
for x=0:1
for y=0:1
for z=0:1
disp(f([x;y;z]))
end
end
end
```

Pour x allant de 1 à 0, on prend pour y allant de 1 à 0, z allant de 1 à 0

Créant ainsi (dans l'ordre)

x	0	0	0	0	1	1	1	1
y	0	0	1	1	0	0	1	1
z	0	1	0	1	0	1	0	1

On affiche alors l'image de tous les M créés par la fonction f

function [colonne1, colonne2]=syndrome(H)

La fonction syndrome renvoie les deux colonnes du tableau des syndromes, la colonne syndromes et la colonne erreurs

Selon la matrice H.

nbligne=size(H,1)

On mémorise le nombre de ligne que possède H

nbelement=(2^nbligne)

On veut multiplier une matrice erreur E avec la matrice H

Alors la matrice syndrome S obtenue doit avoir autant de colonne que H a de lignes.

C'est donc une combinaison de nbligne éléments , chacune pouvant prendre deux valeurs (0 ou 1). Donc il y a $2^{nbligne}$ combinaisons possibles.

Rappel de ce qu'on veut faire :

16
syndrom

La moitié des 16 commence par 0 et l'autre par 1.

1	1	1	1
1	1	1	0
1	1	0	1
1	1	0	0
1	0	1	1
1	0	1	0
1	0	0	0
1	0	0	0
0	1	1	1
0	1	1	0
0	1	0	1
0	1	0	0
0	0	1	1
0	0	1	0

Ensuite on fait la moitié de la moitié pour y mettre des 1 et l'autre moitié prendra des 0

Et on recommence le procédé pour cpt allant de 1 à 4

Ainsi on prend $16/2^{cpt}$
Ici $16/2^3$

0	0	0	1
0	0	0	0

On commence par remplir $16/2^{\text{cpt}}$ de 1 puis $16/2^{\text{cpt}}$ de 0 à partir de là où on a pas encore remplis, et on répète l'action jusqu'à ce que toute la colonne soit rempli.

Pour alterner 1 et 0 on fait compteur modulo 2 et on incrémente compteur.

Fin Rappel

On a dit que la matrice syndrome avait autant de colonne que H avait de ligne.

for a=(1:nbligne)

On fait donc la manipulation ci-dessus de la colonne 1 jusqu'à nbligne (le nombre de colonne pour S)

rempli=0;

On a rien rempli à la base.

for b=(1:2^a)

On remplit la colonne par tranche de 2, une fois de 1 et une fois de 0.

Plus avance dans la colonne et plus ces tranches de 2 sont petites et donc plus on répète l'action

Pour la colonne 1, on fait la manip deux fois, donc 2^1 , et rempli à chaque fois la moitié

Pour la colonne 2, on fait la manip quatre fois, donc 2^2 , et on remplit à chaque fois le quart

Ect ...

Donc on fais la manip de une à $2^{n^{\circ}\text{colonne}}$ fois

colonne1(rempli+1:rempli+nbelement/(2^a),a)= modulo(b,2)
rempli=((nbelement/(2^a))*b)

On remplit la colonne 1 donc le tableau des syndromes, dans la colonne a, et à partir de là où on a fini de remplir, donc rempli+1, jusqu'à une partie de nbélément (moitié,quart,...)

C'est donc nbélément/2, si on est à la première colonne, nbélément/4, si c'est la deuxième etc...

On dis jusqu'à où on a rempli pour que l'instruction suivante commence à partir de là où on a fini de remplir. Exemple : on a rempli 1 fois la moitié, deux fois la moitié, 1 le quart...

```
nbcolonne=size(H,2)
```

On multiplie la matrice Erreur E avec la matrice H : $E \times H$

Pour cela il faut que E possède autant de colonne que H .

```
colonne2=zeros(nbelement,nbcolonne);
```

On associe à un syndrome, une et une seule erreur. La colonne deux aura donc autant d'éléments que la colonne1 et chacune de ces éléments sont des matrices de taille 1,nbcolonne.

Partant du point initial on dit que chaque erreur est 0.

```
rempli=1  
poids=1  
position=nbcolonne+1
```

On va commencer par trouver la première ligne, la variable 'rempli' prend donc 1.

On part du poids 1 (On a un seul '1' dans l'erreur)

'position' est la position du dernier '1' fixe, cf. Projet Scilab.docx, au début il y en a pas, on considère qu'il est en dehors de la matrice. On le déplace en partant de la fin.

```
while (rempli<nbelement & poids<nbcolonne)
```

tant qu'on a pas rempli toutes les lignes et que le nombre de 1 dans la matrice ne dépasse pas son nombre de colonne.

```
cpt=position-1
```

le compteur cpt représente la position du 1 mobile, il part de là où il n'y a pas de '1' fixe. Donc position-1

```
while (rempli<nbelement & (cpt>0))
```

Tant qu'on n'a pas rempli toutes les lignes et que le '1' mobile ne se déplace pas au-delà de la matrice.

```
erreur=zeros(nbcolonne,1)
```

La matrice erreur est une ligne de 0

```
erreur(cpt,1)=1
```

On lui ajoute le '1' mobile.

```
if(poids>1) then
    for (a=1:(poids-1))
        erreur(position-a+1,1)=1
    end
end
```

Si le poids est supérieur à 1, on lui ajoute aussi autant de 1 qu'il en faut en partant de la position du dernier '1' fixe et en rajoutant des 1 à la suite.

```
synd=modulo(H*erreur,2)
```

On regarde le syndrome qu'il produit avec la matrice H

```
exist=%F
```

On part de l'hypothèse qu'il n'existe pas

```
compteur3=1
while ((~exist) & (compteur3<nbelement))
    ...
    compteur3=compteur3+1
end
```

On regarde si le syndrome qu'il produit est dans le tableau en partant de 1 et en allant jusqu'à nbélément

Pour cela

```
exist=(isequal(colonne1(compteur3,:),synd') &
(isequal(colonne2(compteur3,:),zeros(1,nbcolonne))))
```

isequal renvoie vrai si deux matrices sont exactement pareil sinon faux.

Il faut que le syndrome qu'on a trouvé soit égale au syndrome qui est dans le tableau ET que l'erreur qui y est déjà associé soit égale à 0 (il n'y a pas d'erreur associé)

```
if(exist) then
    colonne2(compteur3-1,:)=erreur'
    rempli=rempli+1
end
```

Si le syndrome existe et qu'on peut le mettre alors on le rajoute et on informe qu'on a rempli une ligne

```
cpt=cpt-1
```

On déplace le '1' mobile d'un pas vers le début (vu qu'on part de la position du '1' fixe et que les '1' fixes se trouvent à droite du '1' mobile)

```
if (position==2 | position==nbcolonne+1) then // Modification apporté
    poids=poids+1
    position=nbcolonne
else
    position=position-1;
```

end

Si le dernier '1' fixe est à la deuxième position (la première étant occupé par le '1' mobile) ou si il est à sa position initial (donc en dehors de la matrice) alors on passe au poids suivants et on repositionne le dernier '1' fixe à la fin.

Sinon on le décale d'un pas vers la gauche

```
function M=correction(r)
    [SYNDROME,ERREUR]=syndrome(H)
```

Pour le programme suivant (qui corrige un mot reçu 'r'), j'importe le tableau des syndromes puis j'applique l'énoncé.

A=S(r)

B calcule S(r)

```
exist=%F
cpt=1
while (~exist & cpt<=16) // tant qu'il n'existe pas
    exist=(isequal(SYNDROME(cpt,:),A')) //on compare S( r) à chaque ligne du tableau syndrome
    cpt=cpt+1
end
```

Il cherche dans le tableau le vecteur d'erreurs de poids minimal correspondant

```
if (exist) then
    e=ERREUR(cpt-1) // on prend l'erreur correspondant au syndrome
    c=modulo(r+e,2)
else
    disp("introuvable")
    c=r
end
```

Enfin, il en déduit le mot de code comme à la question 11. Donc on calcule le mot c en fonction de r et e

```
for a=(1:3) on recrée tous les images de f (même méthode)
    rempli=0;
    for b=(1:2^a)
```

```

Mtemp(rempli+1:rempli+((2^3)/(2^a)),a)= modulo(b,2)
rempli=((2^3)/(2^a))*b
end
end

trouve=%F
cpt=1
disp(Mtemp(cpt,:), "1=")
while (~trouve & cpt<2^3) // On cherche de la même manière l'antécédent de c par f
    F=f(Mtemp(cpt,:))

    trouve=(isequal(c,F))
    cpt=cpt+1
end

```

Comme la fonction f est injective, il peut déterminer l'unique antécédent de cet élément du code par, f et enfin retrouver le message de A.

```

if (trouve) then
    M=Mtemp(cpt-1,:)
else
    disp("non trouvé")
    M=zeros(3,1)
end

```

Si on a trouvé, on affiche le mot

Ne vous fiez pas au apparence, le programme est assez simple :

```

function G=transforme(G1)
    nbligne=size(G1,1)
    nbcolonne=size(G1,2)
    final=[eye(3,3);P] // On veut obtenir comme resultat final, G
    disp(G1,"G1=")
    disp(final,"Resultat attendu=")
    if (nbligne==7 & nbcolonne==3) then // il faut donc que la matrice G1 soit de la même taille
        G=G1
        cpt=2
        while (~isequal(G(:,1),final(:,1)) & cpt<=3) // tant que la colonne 1 de G1 n'est pas celle du
résultat attendu et qu'on a pas dépasser la colonne 3
            G(:,1)=modulo(G(:,1)+G(:,cpt),2) // on additionne la première colonne avec les autres
            cpt=cpt+1
        end

        cpt=1
        while (~isequal(G(:,2),final(:,2)) & cpt<=3) // tant que la colonne 2 de G1 n'est pas celle du
résultat attendu et qu'on a pas dépasser la colonne 3

            G(:,2)=modulo(G(:,2)+G(:,cpt),2) // on additionne la deuxième colonne avec les autres

```



```

    cpt=cpt+2
end
cpt=1
    while (~isequal(G(:,3),final(:,3)) & cpt<3) // tant que la colonne 3 de G1 n'est pas celle du
résultat attendu et qu'on a pas dépasser la colonne 3
        G(:,3)=modulo(G(:,3)+G(:,cpt),2) // on additionne la troisième colonne avec les autres

    cpt=cpt+1
end
if (~isequal(G,final)) then
    disp("conversion impossible")
end
else
    disp("Erreur taille matrice")
end

```