PETITHORY Damien
PIERNOT Jérôme

MT51

TP 1 :
Etude des groupes, sous-groupes et transformations de motifs

I. Outils de base

1. Rotation affine

Pour calculer l'image d'un point par la rotation de centre 0 et d'angle θ, on utilise la matrice de rotation :

$$M = \begin{pmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{pmatrix}$$

On peut alors définir la fonction Matlab comme cela (fichier rote.m):

```
% P1 et P2 sont des points (matrice ligne)
% theta est l'angle de la rotation
function [P2] = rote( theta, P1 )

%Matrice de rotation
M = [cos(theta) -sin(theta)
sin(theta) cos(theta)];

%On applique la transformation
P2 = M * P1';

%Transposée pour repasser en "point"
P2 = P2';
```

2. Symétrie affine

Pour calculer l'image d'un point par la symétrie orthogonale de miroir $(O, \vec{i})$, on utilise la matrice de symétrie :

$$M = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix}$$

On peut alors définir la fonction Matlab comme cela (fichier symetrie.m) :

```
% P1 et P2 sont des points (matrice ligne)
function [ P2 ] = symetrie( P1 )

    %Matrice de symetrie
    M = [1 0
         0 -1];

    %On applique la transformation
    P2 = M * P1';

    %Transposée pour repasser en "point"
    P2 = P2';
```

II. Etude du groupe diédral $(D_{2n}, \circ)$ pour n = 4

Pour n = 4, le groupe D_8 contient 8 éléments que l'on peut noter :

$$D_8 = \{\sigma_0, \sigma_1, \sigma_2, \sigma_3, \sigma_4, \sigma_5, \sigma_6, \sigma_7\}$$
$$= \{r^0, r^1, r^2, r^3, s \circ r^0, s \circ r^1, s \circ r^2, s \circ r^3\}$$
$$= \{s^\alpha \circ r^\beta \ / \ \alpha \in \{0;1\}, \beta \in \{0;1;2;3\}\}$$

1. Conversions décimal - quadratique

Le but est d'obtenir les α et β à partir de l'indice de l'élément dans le groupe et vice-versa.

On cherche donc l'écriture en base 4 de l'indice k de l'élément. Ainsi $k = \alpha.4^1 + \beta.4^0$, on en déduit donc que :

- α est le quotient entier de la division euclidienne de k par 4.
- β est le reste de la division euclidienne de k par 4.

On peut bien sûr généraliser à un n quelconque, pas nécessairement égal à 4. On écrit donc les fonctions Matlab suivante :

k → α et β (fichier `trad_dec_qu.m`)

```
function [ alpha, beta ] = trad_dec_qu( k, n )

    alpha = floor(k/n); %symetrie
    beta = mod(k,n); %rotation
```

α et β → k (`trad_qu_dec.m`)

```
function [ k ] = trad_qu_dec( alpha, beta, n )

    k = alpha*n + beta;
```

2. Table de la loi o sur D_8

e_i o $e_j = e_k$

$i \leftrightarrow (\alpha_i, \beta_i)$
$j \leftrightarrow (\alpha_j, \beta_j)$

$k \leftrightarrow (\alpha, \beta)$ avec $\begin{cases} \alpha \equiv \alpha_i + \alpha_j [2] \\ \beta \equiv \beta_j + (-1)^{\alpha_j} \beta_i [4] \end{cases}$

On peut donc écrire une fonction Matlab permettant d'effectuer la composition de 2 éléments d'un groupe diédral quelconque (fichier compose.m) :

```
% i et j sont les indices des éléments à composer
% n désigne le groupe diédral étudié
function [ k ] = compose( i, j, n )

    if i<2*n & j<2*n %Les indices doivent être valides
        %Conversion des indices des sigma en alpha et beta
        [alpha1, beta1] = trad_dec_qu(i,n);
        [alpha2, beta2] = trad_dec_qu(j,n);

        %Calcul des alpha et beta de la composee
        alpha = mod(alpha1+alpha2,2);
        beta = mod(beta2+((-1)^alpha2)*beta1,n);

        %On convertit les alpha et beta obtenus en k
        k = trad_qu_dec(alpha,beta,n);
    else
        %code d'erreur
        k = -1;
    end
```

A partir de cette fonction, il est possible de construire la table de la loi o sur un groupe diédral (fichier table.m) :

```
% n désigne le groupe diédral
function [ T ] = table( n )

    % On construit une matrice de 0 à la bonne taille
    T = zeros(2*n);

    % Puis on la remplie ...
    for i = 1:2*n
        for j = 1:2*n
            % Par les indices de la composé de 2 éléments
            T(i,j) = compose(i-1,j-1,n);
        end
    end
```

On obtient alors :

```
>> table(4)

ans =

     0     1     2     3     4     5     6     7
     1     2     3     0     7     4     5     6
     2     3     0     1     6     7     4     5
     3     0     1     2     5     6     7     4
     4     5     6     7     0     1     2     3
     5     6     7     4     3     0     1     2
     6     7     4     5     2     3     0     1
     7     4     5     6     1     2     3     0

>> table(5)

ans =

     0     1     2     3     4     5     6     7     8     9
     1     2     3     4     0     9     5     6     7     8
     2     3     4     0     1     8     9     5     6     7
     3     4     0     1     2     7     8     9     5     6
     4     0     1     2     3     6     7     8     9     5
     5     6     7     8     9     0     1     2     3     4
     6     7     8     9     5     4     0     1     2     3
     7     8     9     5     6     3     4     0     1     2
     8     9     5     6     7     2     3     4     0     1
     9     5     6     7     8     1     2     3     4     0

>>
```

3. Preuve : $(D_8, \circ)$ est un groupe non abélien

$(D_8, \circ)$ est un groupe ssi :

- o est une loi interne.
- o est associative.
- Il existe un élément neutre σ_e pour la loi o (i.e $\forall i \in \{0;..;7\} \quad \sigma_i \circ \sigma_e = \sigma_i$)
- Tout élément admet un symétrique (i.e $\forall i \in \{0;..;7\} \quad \sigma_i \circ \sigma_i^{-1} = \sigma_e$)

La loi o est interne puisque tous les éléments de la table sont des éléments du groupe. Donc la composé de 2 éléments quelconque du groupe appartient au groupe.

La loi o est toujours associative.

L'élément neutre est évident : c'est l'identité $\sigma_0 = s^0$ o r^0 = ide

Tous les éléments du groupe admettent un symétrique puisque l'on retrouve un 0 (indice de l'élément neutre) par ligne de la table. Cela veut dire qu'il est possible de trouver, pour chaque élément σ_i, un élément σ_j tel que $\sigma_i \circ \sigma_j = \sigma_0$

De plus, le groupe n'est pas abélien (non commutatif). Cela se voit très facilement avec la table puisque, par exemple $\sigma_1 \circ \sigma_6 = \sigma_5$ alors que $\sigma_6 \circ \sigma_1 = \sigma_7$

On sait que :

$e_i \circ e_j = e_k$

$i \leftrightarrow (\alpha_i, \beta_i)$
$j \leftrightarrow (\alpha_j, \beta_j)$
$k \leftrightarrow (\alpha, \beta)$ avec $\begin{cases} \alpha \equiv \alpha_i + \alpha_j [2] \\ \beta \equiv \beta_j + (-1)^{\alpha_j} \beta_i [n] \end{cases}$

On connaît (α_i, β_i) et $(\alpha, \beta) = (0,0)$, On peut donc trouver (α_j, β_j) par les formules suivantes :

$$\begin{cases} \alpha_j \equiv -\alpha_i [2] \\ \beta_j \equiv -(-1)^{\alpha_j} \beta_i [n] \end{cases}$$

Il est alors simple de créer une fonction Matlab permettant de trouver l'inverse d'un élément (fichier inverse.m) :

```
% i est l'indice de l'élément dont on recherche l'inverse
% n désigne le groupe diédral étudié
function [ j ] = inverse( i, n )

    %Conversion de l'indice du sigma en alpha et beta
    [alpha1, beta1] = trad_dec_qu(i,n);

   %Calcul des alpha et beta de la composee pour trouver le
    neutre
    alpha = mod(-alpha1,2);
    beta = mod(-beta1*(-1)^alpha,n);

    %On convertit les alpha et beta obtenus en k
    j = trad_qu_dec(alpha, beta, n);
```

A l'exécution on retrouve bien les valeurs présentes dans la table du groupe D8

```
>> inverse(1,4)

ans =

     3

>> inverse(6,4)

ans =

     6

>>
```

4. Sous-groupes de D_8

Pour trouver tous les sous-groupes d'un groupe diédral, il faut construire les sous-groupes engendrés par chacun des éléments du groupe, puis les sous-groupes engendrés par deux éléments du groupe.

Ainsi on écrit la fonction Matlab suivante (fichier sous_groupe.m) :

```
% n désigne le groupe diédral étudié
function [ S_Groupe ] = sous_groupe(n)

    % Creation de la solution
    S_Groupe = [];

    % Generation des sous groupes engendres ...
    % Par un élément ...
    for i = 0:2*n-1
       SG_engendre = remplir(sort(s_groupe_engendre(i,n)),2*n);
       % Si c'est un nouveau sous-groupe ...
       if appartient(SG_engendre,S_Groupe,2)==0
           % ... Alors on l'ajoute
           S_Groupe = [S_Groupe SG_engendre];
       end
    end

    % Puis par 2 éléments
    for i = 0:n-1
        for j = n:2*n-1
            SG_engendre = remplir(sort(s_groupe_engendre([i;j],n)),2*n);
            % Si c'est un nouveau sous-groupe ...
            if appartient(SG_engendre,S_Groupe,2)==0
                % ... Alors on l'ajoute
                S_Groupe = [S_Groupe SG_engendre];
            end
        end
    end
```

La fonction sort permet de trier les éléments du sous-groupe par ordre croissant afin de faciliter le test qui vise à vérifier si le sous-groupe que l'on vient de construire n'existe pas déjà.

La fonction remplir permet de compléter le sous-groupe avec des -1 pour que tous aient la même taille afin de les regrouper dans un tableau.

La fonction s_groupe_engendre permet de construire le sous-groupe engendré par une liste d'éléments du groupe (fichier s_groupe_engendre.m).

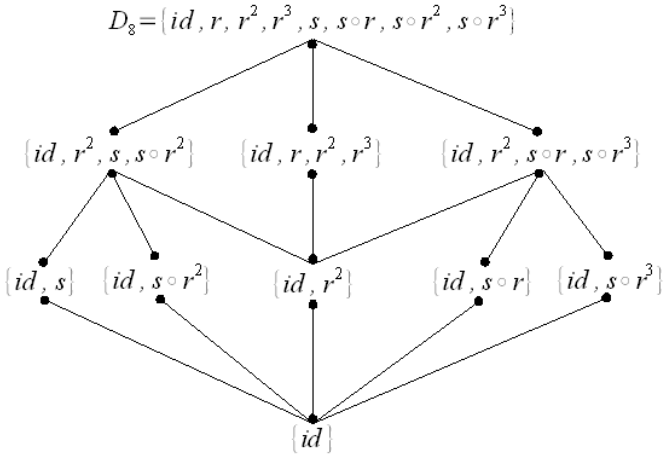
```
% i désigne le(s) élément(s) qui engendre(nt) le sous-groupe
function [ SG ] = s_groupe_engendre( i, n )

    % Initialisation de la solution
    SG = [0];

    % i n'est pas le neutre, il faut donc l'ajouter au sous-groupe
    if i ~= 0
        SG = [SG;i];
    end

    % On initialise la liste des éléments non encore traités
    new = SG;

    %Recherche des elements du sous groupe
    taille = length(new);
    %Tant qu'il y a encore des éléments à traiter
    while taille>0
        for k = 1:length(SG)
            % On compose cet élément avec tous ceux déjà présent dans
            le sous-groupe
            comp = compose(new(1),SG(k),n);
           % Si c'est un nouvel élément ...
            if ~appartient(comp, SG,1)
                % ... Alors on l'ajoute à la solution...
                SG = [SG;comp];
                 % On utilise le théorème de Lagrange pour arrêter la
                 fonction plus rapidement
                if length(SG)>n
                    SG = (0:(2*n-1))';
                    break;
                end
                % ... ainsi qu'a la liste des éléments à traiter
                new = [new;comp];
                taille = taille+1;
            end
        end

       % On supprime l'élément que l'on vient de traiter de la liste
       des éléments à traiter
        if taille>1
            new = new(2:taille);
        else
            new = [];
        end

        taille = taille-1;
    end
```

Grâce à ces fonctions on obtient alors la liste des sous-groupes d'un groupe diédral quelconque.

```
>> sous_groupe(4)

ans =

     0     0     0     0     0     0     0     0     0     0
    -1     1     2     4     5     6     7     1     2     2
    -1     2    -1    -1    -1    -1    -1     2     4     5
    -1     3    -1    -1    -1    -1    -1     3     6     7
    -1    -1    -1    -1    -1    -1    -1     4    -1    -1
    -1    -1    -1    -1    -1    -1    -1     5    -1    -1
    -1    -1    -1    -1    -1    -1    -1     6    -1    -1
    -1    -1    -1    -1    -1    -1    -1     7    -1    -1

>> sous_groupe(5)

ans =

     0     0     0     0     0     0     0     0
    -1     1     5     6     7     8     9     1
    -1     2    -1    -1    -1    -1    -1     2
    -1     3    -1    -1    -1    -1    -1     3
    -1     4    -1    -1    -1    -1    -1     4
    -1    -1    -1    -1    -1    -1    -1     5
    -1    -1    -1    -1    -1    -1    -1     6
    -1    -1    -1    -1    -1    -1    -1     7
    -1    -1    -1    -1    -1    -1    -1     8
    -1    -1    -1    -1    -1    -1    -1     9

>>
```

```
>> s_groupe_engendre(1,4)

ans =

     0
     1
     2
     3

>> s_groupe_engendre([2;5],4)

ans =

     0
     2
     5
     7

>>
```

On trouve donc 9 sous-groupes pour D8 qu'il est possible d'ordonner selon la relation d'inclusion. On obtient alors le diagramme de Hasse suivant :



III. Etude du groupe diédral $(D_{2n}, \circ)$ pour n quelconque

Les fonctions présenter tout au long de ce compte rendu sont déjà généralisées afin de pouvoir étudier l'ensemble des groupes diédral $(D_{2n}, \circ)$.

IV. Transfert de motifs

1. Sous-groupe opérant sur un motif

On utilise le motif M_0 du sujet de TP et on lui fait subir l'ensemble des transformations géométriques présentes dans chacun des sous-groupes obtenus précédemment.

C'est la fonction create_motif qui permet de réaliser ces opérations (fichier create_motif.m) :

```
% sg est le sous-groupe que l'on fait opérer sur le motif
% n désigne le groupe diédral étudié
% Xd et Yd sont les points caractérisant le motif
function [ ] = create_motif( sg, n, Xd, Yd )

    % On récupère les alpha et beta de chaque éléments de sg
    [alpha, beta] = trad_dec_qu(sg,n);

    taille = size(sg,1); %Nombre d'elements dans le sous groupe

    % Pour chaque éléments ...
    for i = 1:taille
        X = Xd;
        Y = Yd;

        % ... On effectue la rotation de chacun des points ...
        P = rote(beta(i,1)*2*pi/n,[X Y]);
        X = P(:,1);
        Y = P(:,2);

        % ... Puis la symetrie si il y en a une
        if alpha(i,1)==1
            P = symetrie(P);
            X = P(:,1);
            Y = P(:,2);
        end

        % Il ne reste plus qu'a tracer le motif
        line(X,Y);

    end
```

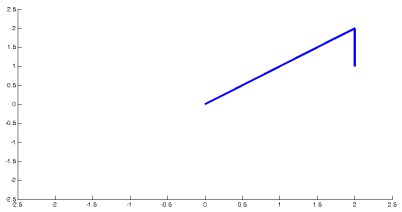
Après avoir défini :

X = [0 ;2 ;2] ; Y = [0 ;2 ;1] ;

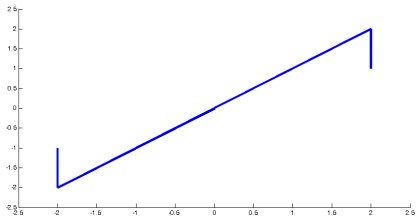
n = 4 ;

On obtient finalement les motifs suivants :

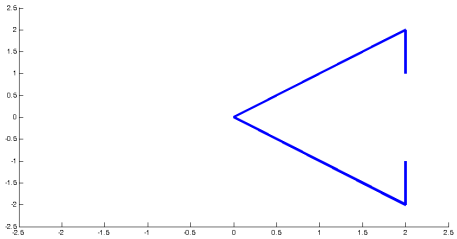
SG = s_groupe_engendre(0,n) ;



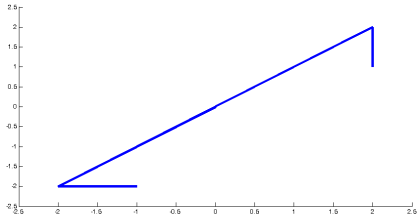
SG = s_groupe_engendre(2,n) ;



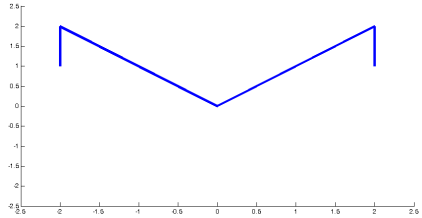
SG = s_groupe_engendre(4,n) ;



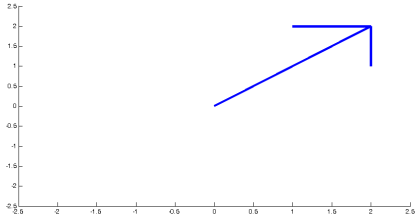
SG = s_groupe_engendre(5,n) ;



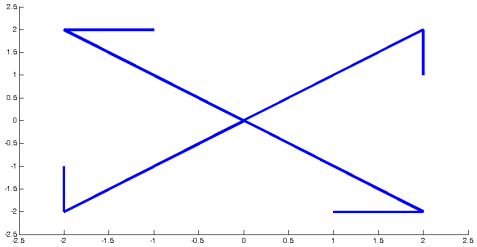
SG = s_groupe_engendre(6,n) ;



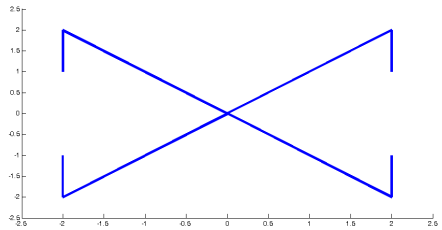
SG = s_groupe_engendre(7,n) ;



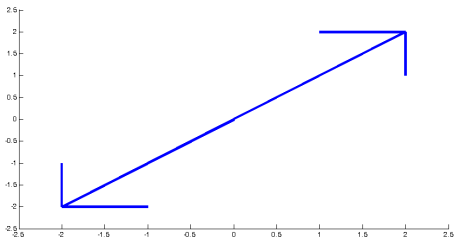
SG = s_groupe_engendre(1,n) ;



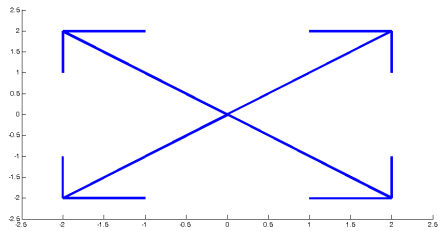
SG = s_groupe_engendre([2 ;4],n);



SG = s_groupe_engendre([2;5],n) ;



SG = s_groupe_engendre([0:8],n) ;



2. Transfert de formes de base

La rotation et la symétrie étant des isométries, elles conservent les longueurs, les angles et donc les formes.

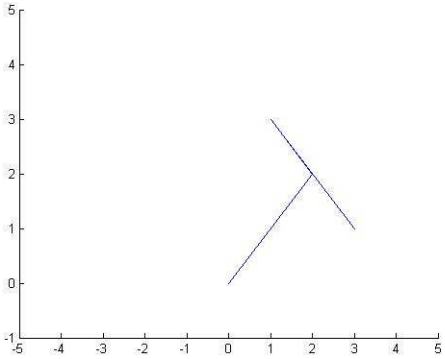
Ainsi pour faire opérer un groupe diédral sur une forme de base, il suffit de le faire opérer sur les éléments caractéristiques de cette forme.

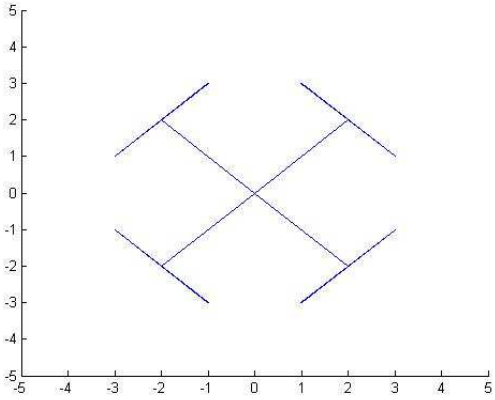
Par exemple, pour un cercle il suffit de faire opérer le groupe sur le centre et de retracer un cercle de même rayon que le cercle initial.

3. Prolongements

On peut définir d'autres motifs, et faire opérer le groupe D8 sur ces nouveaux motifs, par exemple :

```
>> X = [0;2;1;3];
>> Y = [0;2;3;1];
>> SG = [0;1;2;3;4;5;6;7];
>> create_motif(SG,4,X,Y)
```





```
>> X = [0;2;1.5;2.5];
>> Y = [0;2;3;3];
>> SG = [0;1;2;3;4;5;6;7];
>> create_motif(SG,4,X,Y)
```