Système d'inférence floue : application au contrôle
Approximation d'une fonction de contrôle

1 Présentation générale

L'objectif de ce TP est de se familiariser avec l'écriture de règles en logique floue. Il s'agit en particulier de comprendre l'influence des différents paramètres (fonction d'appartenance des différents termes linguistiques, type d'inférence...) dans la forme finale de la fonction de contrôle réalisée.

Afin de pouvoir visualiser aisément la surface correspondant à la fonction f générée par un ensemble de règles floues, on s'intéresse au cas simple suivant :

$$f : [0,1] \times [0,1] \mapsto [0,1]$$

Les variables d'entrée de la fonction de contrôle f sont donc au nombre de 2 : x et y. La variable de sortie s'appelle z. Ces variables prennent leurs valeurs sur $\mathbb{R}$.

À chaque variable réelle x, y et z est associée une variable linguistique, respectivement X, Y et Z.

2 Description des règles

Les règles du contrôleur flou permettant d'associer les variables linguistiques X et Y à Z doivent être décrites selon un langage particulier dans un fichier comprenant trois parties principales (cf. support de cours pages 100 et 102) :

1. **Configuration** : cette partie est optionnelle. Elle permet de spécifier :
   - Le type de connectif `et`. Les valeurs possibles sont `min`, `produit`, `produit_borne` et `produit_drastique`. Le type de connectif `et` permet d'indiquer quelle doit être la norme triangulaire utilisée.
   - Le type de connectif `ou`. Les valeurs possibles sont `max`, `somme`, `somme_bornee` et `somme_drastique`. Le type de connectif `ou` permet d'indiquer quelle doit être la co–norme triangulaire utilisée.
   - Le type de connectif `ainsi_que`. Les valeurs possibles sont `somme` et `max`.
   - L'implication utilisée `implication`. Les valeurs possibles sont `larsen` et `mamdani`.
   - Le type de décodage `decodage`. Les valeurs possibles sont `max`, `moyenne_max` et `centre`. **REMARQUE :** en cas d'incompétence (fonction d'appartenance de la variable de sortie en dessous d'un certain seuil) la valeur prise par la variable de sortie est 0.

   Des valeurs par défaut sont prises si un des champs n'est pas renseigné.

2. **Description des valeurs linguistiques** : cette partie a pour but de définir la fonction d'appartenance du sous-ensemble flou correspondant à chaque valeur linguistique. Les fonctions d'appartenance peuvent avoir les formes suivantes :
   (a) triangle(x_1, x_2, x_3).
   (b) trapeze(x_1, x_2, x_3, x_4).
   (c) rampe_bas(x_1, x_2).
   (d) rampe_haut(x_1, x_2).

   La forme des fonctions d'appartenance ainsi que le rôle des différents paramètres sont regroupés sur la figure 1.

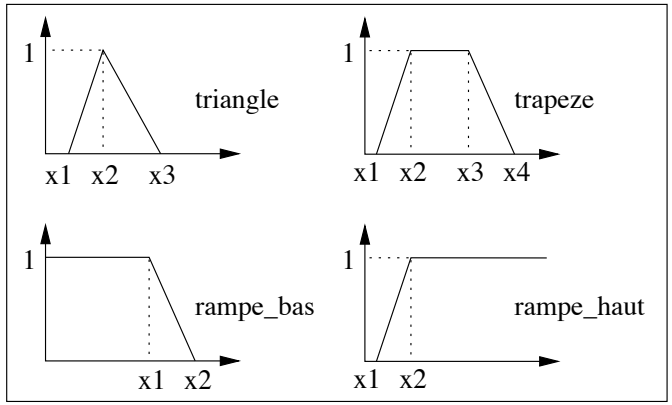

FIG. 1 – Types disponibles pour les fonctions d'appartenance

3. **Description des règles** ; ceci est un exemple de règle :

   si X est A alors Y est B

   On peut ajouter les quantificateurs pas, peu et tres.
   Exemple :

   si X est pas A alors Y est B

   On peut combiner des conclusions (uniquement avec et).
   Exemple :

   si X est A alors Y est B et Z est C

   On peut combiner des propositions avec et et ou.
   Rappel : le et est prioritaire par rapport au ou.
   Exemples :

   si X est A ou Y est B alors Z est C

   si X est A et Y est B alors Z est C

   si X est A ou Y est B et Z est C alors T est D

La grammaire complète du langage permettant d'interpréter les règles floues est la suivante :

```
PROGRAM ->DEFCONFIG DEFLINGUIS DEFRULES

/*------------------------------------------------------------------------*/
/*                                                                        */
/* CONTROLLER CONFIGURATION                                               */
/*                                                                        */
/*------------------------------------------------------------------------*/


DEFCONFIG ->CONFIG AO LISTCONFIG AF
| /* empty */



LISTCONFIG ->LISTCONFIG CONFIG
| CONFIG


CONFIG ->AND EQ IDENT PV
| OR EQ IDENT PV
| IMPLIQ EQ IDENT PV
| ALSO EQ IDENT PV
| DEFUZZ EQ IDENT PV


/*------------------------------------------------------------------------*/
/*                                                                        */
/* LINGUISTIC DEFINITIONS                                                 */
/*                                                                        */
/*------------------------------------------------------------------------*/


DEFLINGUIS ->LINGUIS  AO LISTLINGUIS AF



LISTLINGUIS ->LISTLINGUIS LINGUIS
| LINGUIS


LINGUIS ->IDENT EQ TRIANGLE PO
REAL V REAL V REAL
PF PV
```



```
| IDENT EQ TRAPEZE PO
REAL V REAL V REAL V REAL
  PF PV
| IDENT EQ UP_RAMP PO
REAL V REAL
  PF PV
| IDENT EQ DOWN_RAMP PO
REAL V REAL
  PF PV


/*------------------------------------------------------------------------*/
/*                                                                        */
/* RULES DEFINITIONS                                                      */
/*                                                                        */
/*------------------------------------------------------------------------*/


DEFRULES ->RULES AO LISTRULES AF



LISTRULES ->LISTRULES RULES
| RULES


RULES ->  IF PROP THEN LISTACTIONS PV


PROP ->PROPELEM
| PROP AND PROP
| PROP OR PROP


PROPELEM ->IDENT IS LISTMOD IDENT


LISTMOD ->LISTMOD MOD
                | /* EMPTY */


LISTACTIONS ->ACTION
                | LISTACTIONS AND LISTACTIONS


ACTION ->  IDENT IS IDENT
```

```
DIGIT ->[0-9]
INTEGER ->"-"?{digit}+
REAL ->"-"?{digit}+[.]{digit}*
LETTER ->[_a-zA-Z]
IDENT ->{letter}({letter}|{digit})*
%%


CONFIGURATION   -> configuration
LINGUISTIQUE    -> linguistique
RULES           -> regles
TRIANGLE        -> triangle
TRAPEZE         -> trapeze
UP_RAMP         -> rampe_haut
DOWN_RAMP       -> rampe_bas
DEFUZZ          -> decodage
IF              -> si
IS              -> est
THEN            -> alors
MOD             -> tres |
                   peu |
                   pas
AND             -> et
OR              -> ou
IMPLIQ          -> implication
ALSO            -> ainsi_que
AO              -> {
AF              -> }
PO              -> (
PF              -> )
PV              -> ;
V               -> ,
EQ              -> :=
```

3 Génération de la surface

À partir d'un fichier de règles, il est possible de générer la surface correspondante ($z = f(x, y)$) grâce à la commande `fonction`.

```
fonction [-v] fuzzyFile
```

Ce programme génère un fichiers de points : `surface` qui peut être visualisé avec `gnuplot`.

4 Visualisation de la surface

Le fichier produit, `surface`, peut être exploité par l'utilitaire de visualisation `gnuplot` qui est un programme de visualisation de données 2D et 3D. Il s'appuie sur une interface en mode texte et possède un certain nombre de commandes prédéfinies. Une aide en ligne est disponible à tout moment grâce à la commande `help` seule ou suivie d'un nom. Un historique des commandes précédentes est géré et est accessible grâce aux touches `Ctrl-P` pour reculer et `Ctrl-N` pour avancer. Enfin, il n'est pas nécessaire de taper les commandes en entier, `gnuplot` étant capable, s'il n'y a pas conflit, de reconstuire le nom complet à partir des premières lettres (par exemple `with line` peut être saisi `w l`).

Exemple de session de visualisation :

`gnuplot> set para`
Abréviation de "`set parametric`". Indique à `gnuplot` que le fichier de données contient une liste de points X, Y, Z.
`gnuplot> splot 'surface' w l`
Trace la surface correspondant aux points contenus dans le fichier "`surface`". La visualisation de la surface est réalisée grâce à des lignes (commande `w l`).
`gnuplot> quit`
Sans commentaire ...

Il est possible de changer le point de vue de la visualisation grâce à la commande `set view <rotX>,<rotZX>` où `rotX` représente l'angle en degré autour de l'axe X (axe horizontal sur l'écran) et `rotZ` l'angle en degré autour de l'axe Z (axe vertical à l'écran). Remarque : il est possible de changer à chaque commande les deux valeurs simultanément ou seulement une des deux valeurs. Exemples :

- `set view 15,30`
- `set view 15` (ne change que la rotation par rapport à X)
- `set view ,30` (ne change que la rotation par rapport à Z)

Après l'utilisation de `set view`, il est nécessaire de relancer une commande `splot` afin de remettre à jour l'affichage.

5 Problème à résoudre

1. Écrire un jeu de règles permettant de décrire la surface de la figure 2. Remarque : la rupture du plan vertical au plan incliné se produit pour $x = 0.5$.
2. Changer le type d'inférence et observer le résultat produit.
3. Modifier le fichier de la question 1 afin de *creuser* le plan incliné (cf. figure 3).
4. Modifier une nouvelle fois les règles de la question 1 afin d'obtenir la surface décrite sur la figure 4.

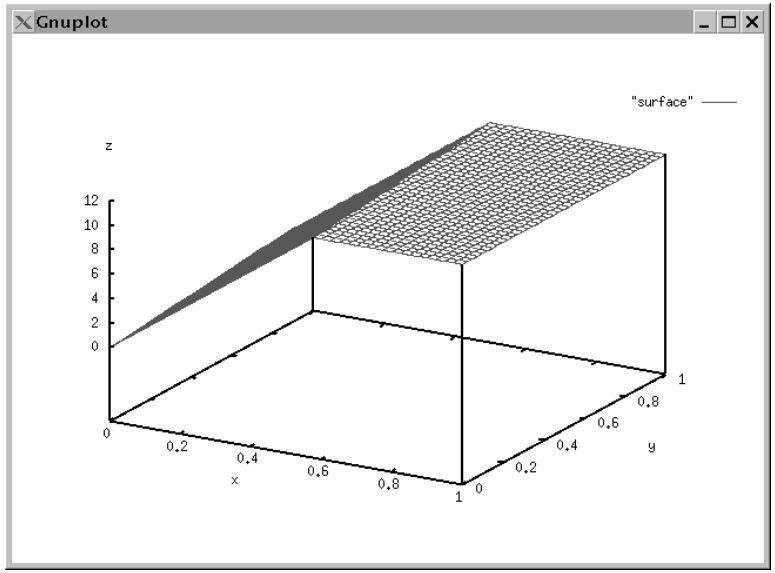

FIG. 2 –

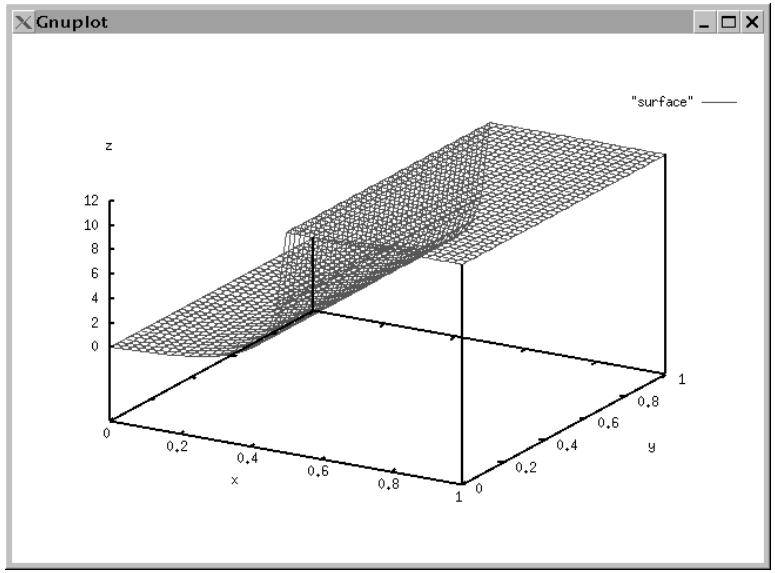

FIG. 3 –

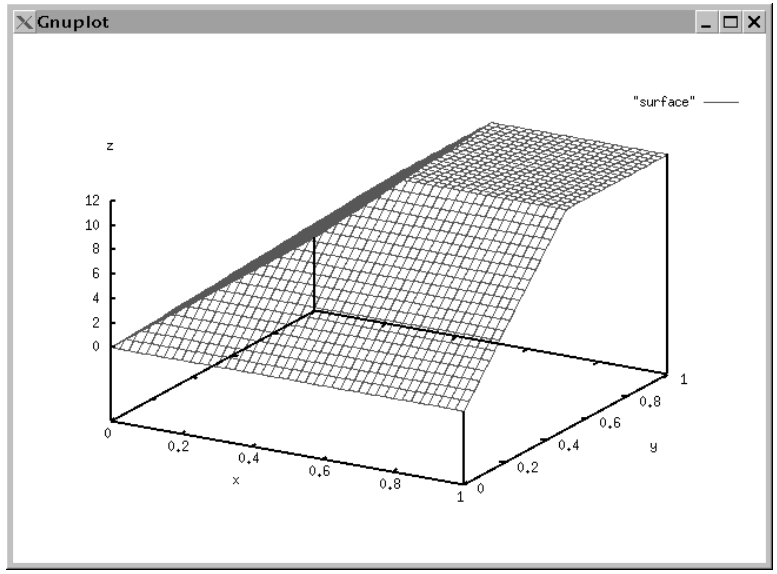

FIG. 4 –