وزارة التعليـــم العـــالـــي و الـبحـث الـعلمـــي
Ministère de l'Enseignement Supérieur et de la Recherche Scientifique
جـا معـة جـيـجـل
Université de Jijel
كليـة العلوم والتكنولوجيا
Faculté des Sciences et de la Technologie
قسم الهندسة الكهربائية
Département de Génie Electrique

Systèmes hybrides

Master II électrotechnique industriel
Module : Intelligence Artificielle

Les systèmes hybrides

- Exploitent les avantages respectifs de plusieurs paradigmes en combinant leurs algorithmes suivant une approche synergétique.
- Exemples de modèles pouvant être combinés :

  - Systèmes experts
  - Raisonnement à base de cas
  - Algorithmes et programmation génétique
  - Réseaux de neurones
  - Techniques de régression
  - Techniques statistiques
  - Systèmes à logique floue
  - Algorithmes de groupement
  - Techniques de simulation
  - …

Les systèmes hybrides ne sont pas tous égaux

- Un système hybride peut être bon ou mauvais, selon le choix et les rôles de ses composantes.
- Lotfi Zadeh est réputé avoir dit qu'un bon système hybride combine les qualité de la police britannique, de la mécanique allemande, de la cuisine française et du système bancaire suisse.
- Par contre, la cuisine britannique, la police française, la mécanique suisse et les finances allemand seraient un mauvais choix !
- Hybridations populaires
  - Systèmes neuro-flous, neuro-génétiques, flous-génétiques
  - Systèmes experts neuronaux

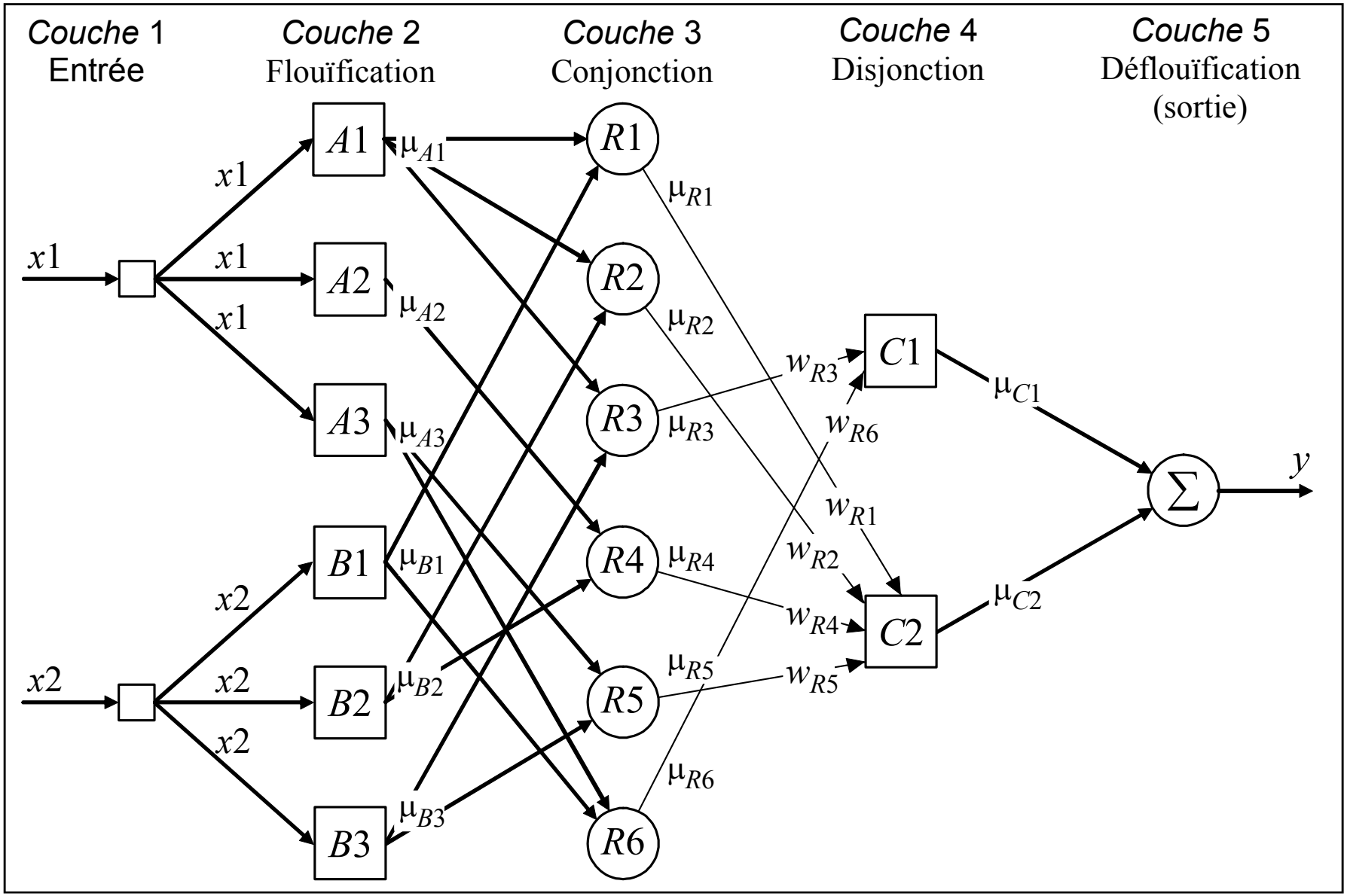
Types d'hybridation

- Modèles séquentiels
  - Entrée → Modèle 1 → Modèle 2 → sortie
  - Ex. : un module statistique passe ses résultats à un RNA.
  - Forme la plus faible d'hybridation.
- Modèles à auxiliaire
  - Entrée → Modèle 1 → Sortie
    ↑
    Modèle 2 (appelé par 1)
  - Ex. : AG qui règlent les poids d'un RNA.
  - Forme d'hybridation plus poussée.
- Modèles imbriqués
  - Entrée → Modèle 1 + Modèle 2 → sortie
  - Ex. : un système neuro-flou où le RNA est imbriqué dans le système flou.
  - Forme absolue d'hybridation
- Les modèles peuvent être combinés pour créer d'autres plus complexes.

Système neuro-flous

- Combine les traitement parallèle et les capacités d'apprentissage des réseaux de neurones avec les raisonnement quasi-humaines et les capacités d'explication des systèmes flous: Les RNA deviennent plus transparents, les systèmes flou acquièrent la capacité d'apprendre.
- La topologie du RNA est fonctionnellement équivalente à celle d'un modèle d'inférence flou, et on peut l'entraîner à :
  - Développer des règles floues SI-ALORS
  - Trouver les fonctions d'appartenance de variables d'entrées/sorties.
- Structure similaire à un PMC : 1 couche d'entrée, 1 couche de sortie et 3 couches cachées pour les fonctions d'appartenance et les règles.

Architecture d'un système neuro-flou



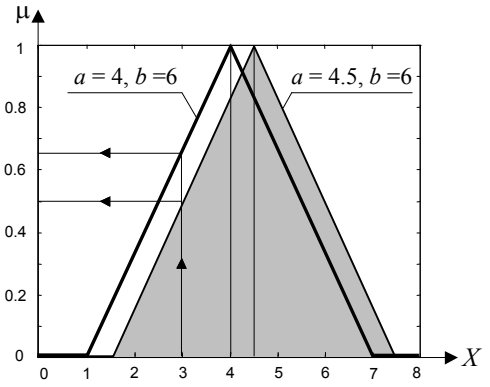
***Couche* 1** : Agit comme un tampon qui transmet simplement les données d'entrée à la couche de flouïfication. On a :

$$y_i^{(1)} = x_i^{(1)}$$

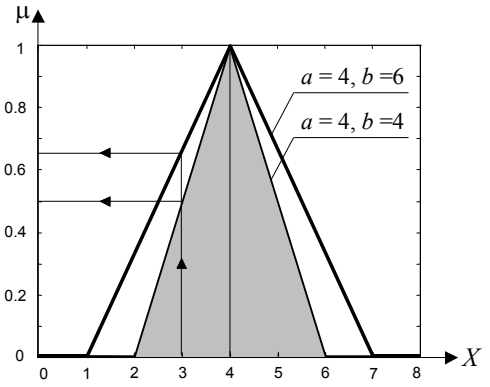
***Couche* 2** : Les neurones de cette couche réalisent chacun un ensemble flou qui servira dans les antécédents des règles. Chaque neurone reçoit une valeur d'entrée précise et génère son degré d'appartenance à l'ensemble flou représenté par le neurone.

Dans le cas d'ensembles flous triangulaires, on peut utiliser des fonctions d'appartenance de même forme qui sont définies par deux paramètres {*a*, *b*} :

$$y_i^{(2)} = \begin{cases} 0, & \text{if } x_i^{(2)} \le a - \dfrac{b}{2} \\ 1 - \dfrac{2\left|x_i^{(2)} - a\right|}{b}, & \text{if } a - \dfrac{b}{2} < x_i^{(2)} < a + \dfrac{b}{2} \\ 0, & \text{if } x_i^{(2)} \ge a + \dfrac{b}{2} \end{cases}$$



(*a*) Effect of parameter *a*.



(*b*) Effect of parameter *b*.

***Couche* 3** : Elle réalise les conjonctions des antécédents de règles floues. Chaque neurone dans la couche reçoit comme entrée les degrés d'appartenance définis dans la couche 2. L'intersection floue est réalisée avec l'opérateur produit:

$$y_i^{(3)} = x_{1i}^{(3)} \times x_{2i}^{(3)} \times \ldots \times x_{ki}^{(3)}$$

$$y_{R1}^{(3)} = \mu_{A1} \times \mu_{B1} = \mu_{R1}$$

***Couche* 4** : Elle réalise les disjonctions des antécédents de règles floues. Chaque neurone dans cette couche reçoit comme entrée les degrés d'appartenance définis dans la couche 3. L'union floue est réalisé à l'aide de l'opérateur :

$$y_i^{(4)} = x_{1i}^{(4)} \oplus x_{2i}^{(4)} \oplus \ldots \oplus x_{li}^{(4)}$$

$$y_{C1}^{(4)} = \mu_{R3} \oplus \mu_{R6} = \mu_{C1}$$

μ_{C1} représente la force combinée des conjonctions implémentées par les neurones *R*3 et *R*6.

***Couche* 5** : Chaque neurone dans cette couche prend les ensembles flous précédent, écrêtés (limités) par la force des règles correspondantes, et les combine en un seul ensemble flou. Ce dernier est alors déflouïfié par une méthode standard.

ANFIS: Adaptive Neuro-Fuzzy Inference System

- Modèle de génération automatique de règles floues basé sur le modèle d'inférence de Sugeno :

  IF x_1 is A_1 AND x_2 is A_2 . . . AND x_m is A_m
  THEN $y = f(x_1, x_2, \ldots, x_m)$

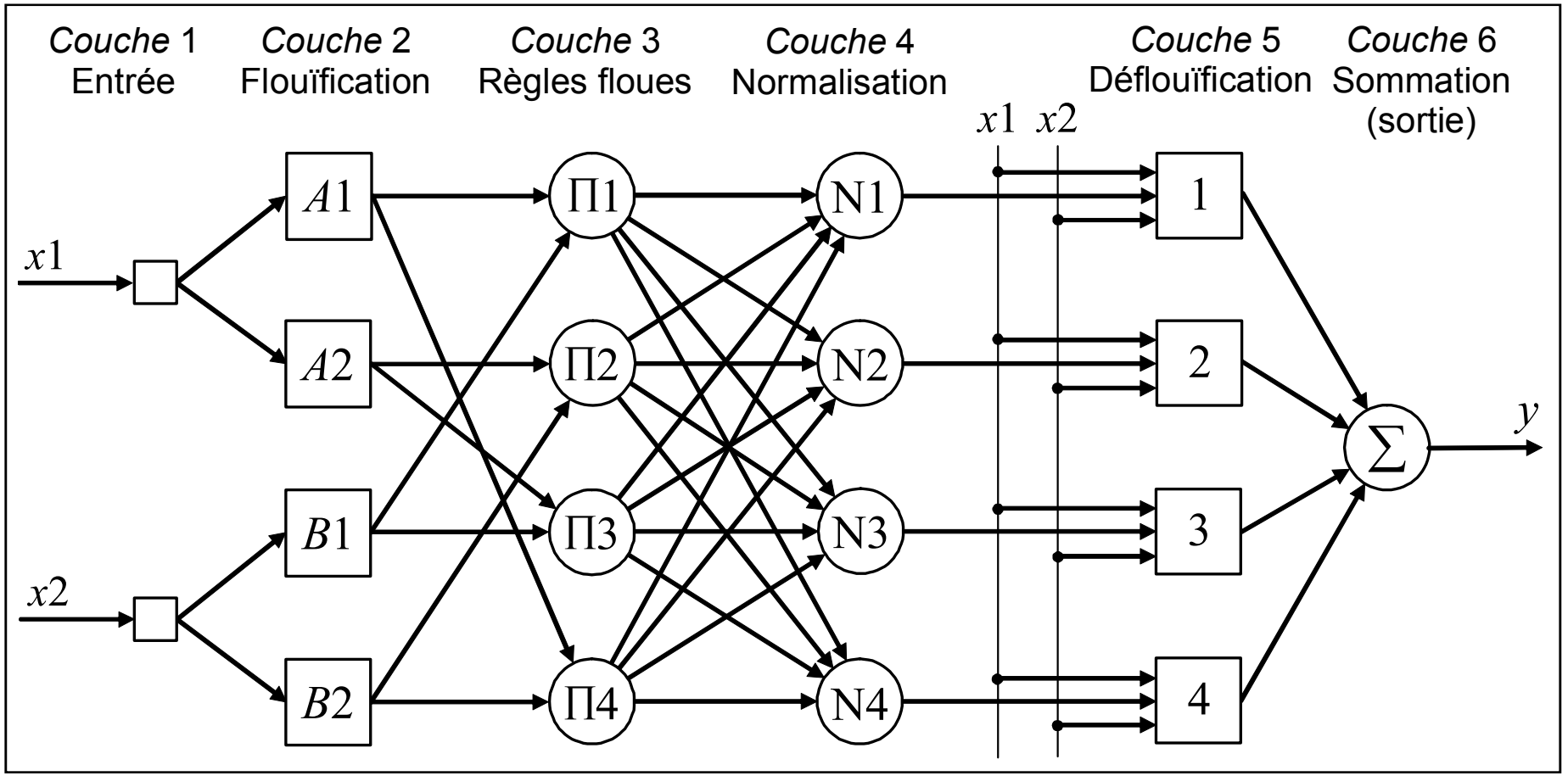
  où $x_1, x_2, \ldots, x_m$ sont des variables d'entrée et $A_1, A_2, \ldots, A_m$ sont des ensembles flous.

- Lorsque :
  - y= constante, on obtient un modèle de Sugeno d'ordre zéro.
  - y est une combinaison linéaire des entrées :

    $$y = k_0 + k_1 x_1 + k_2 x_2 + \ldots + k_m x_m$$

    on obtient un modèle de Sugeno de premier ordre.

Architecture du réseau ANFIS



Couche 1 : Tampon pour les données d'entrée.

Couche 2 : Neurones de flouïfication pour les antécédents des règles. Dans le modèle de Jang, les fonctions d'appartenance sont des gaussiennes.

Couche 3 : Chaque neurone correspond à une règle floue de Sugeno. il reçoit les sorties des neurones de flouïfication et calcule son activation. La conjonction des antécédents est réalisée avec l'opérateur produit. Ainsi, la sortie du neurone *i* de la couche 3 est donnée par :

$$y_i^{(3)} = \prod_{j=1}^{k} x_{ji}^{(3)} \quad \text{et} \quad y_{\Pi 1}^{(3)} = \mu_{A1} \times \mu_{B1} = \mu_1,$$

où μ_1 represente le degré de vérité de la *Règle 1*.

Couche 4 : Chaque neurone calcule le degré de vérité normalisé d'une règle floue donnée. La valeur obtenue représente la contribution de la règle floue au résultat final. Ainsi la sortie du neurone *i* de la couche 4 est :

$$y_i^{(4)} = \frac{x_{ii}^{(4)}}{\sum_{j=1}^{n} x_{ji}^{(4)}} = \frac{\mu_i}{\sum_{j=1}^{n} \mu_j} = \overline{\mu}_i$$

$$y_{N1}^{(4)} = \frac{\mu_1}{\mu_1 + \mu_2 + \mu_3 + \mu_4} = \overline{\mu}_1$$

Couche 5 : Chaque neurone *i* de cette couche est relié à un neurone de normalisation correspondant et aux entrées initiales du réseau. Il calcule le conséquent pondéré de la règle sous jacente comme étant

$$y_i^{(5)} = x_i^{(5)} \left[k_{i0} + k_{i1}\, x1 + k_{i2}\, x2\right] = \overline{\mu}_i \left[k_{i0} + k_{i1}\, x1 + k_{i2}\, x2\right]$$

où les Xi sont les entrées, et $k_{i\,0}$, $k_{i\,1}$ et $k_{i\,2}$ sont des paramètres du conséquent de la règle *i*.

Couche 6 : Comprend un seul neurone qui fournit la sortie de ANFIS en calculant la somme des sorties de tous les neurones de déflouïfication.

$$y = \sum_{i=1}^{n} x_i^{(6)} = \sum_{i=1}^{n} \overline{\mu}_i \left[k_{i0} + k_{i1}\, x1 + k_{i2}\, x2 \right]$$

Entraînement d'un réseau ANFIS

- Algorithme à deux temps où on estime d'abord les paramètres des conséquents par une technique de moindres carrés et ensuite les poids du réseau par une descente de gradient.
- Chaque époque d'entraînement comprend une passe avant et une passe arrière. Durant la passe avant, les patrons d'entrée servent à déterminer les sorties des neurones couche par couche, permettant de déterminer les valeurs des paramètres de conséquents en premier temps.
- Durant la passe arrière, l'algorithme de retropropagation d'erreur est appliqué pour régler les poids des différentes couches.

Détermination des paramètres des conséquents

- Partant de *P* paires d'apprentissage, on peut former *P* équations linéaires en fonction des paramètres des conséquents :

$$\begin{cases} y_d(1) = \overline{\mu}_1(1) f_1(1) + \overline{\mu}_2(1) f_2(1) + \ldots + \overline{\mu}_n(1) f_n(1) \\ y_d(2) = \overline{\mu}_1(2) f_1(2) + \overline{\mu}_2(2) f_2(2) + \ldots + \overline{\mu}_n(2) f_n(2) \\ \vdots \\ y_d(p) = \overline{\mu}_1(p) f_1(p) + \overline{\mu}_2(p) f_2(p) + \ldots + \overline{\mu}_n(p) f_n(p) \\ \vdots \\ y_d(P) = \overline{\mu}_1(P) f_1(P) + \overline{\mu}_2(P) f_2(P) + \ldots + \overline{\mu}_n(P) f_n(P) \end{cases}$$

où $\overline{\mu}_i$ est la valeur moyenne de μ_i, et $f_i()$ est la fonction de sortie dont on veut déterminer les paramètres.

- On peut écrire l'équation précédente sous la forme $\boldsymbol{y}_d = \mathbf{A}\,\boldsymbol{k}$, où $\boldsymbol{y}_d$ est un vecteur désiré de dimension $P \times 1$:

$$\boldsymbol{y}_d = \begin{bmatrix} y_d(1) \\ y_d(2) \\ \vdots \\ y_d(p) \\ \vdots \\ y_d(P) \end{bmatrix}, \quad \mathbf{A} = \begin{bmatrix} \bar{\mu}_1(1) & \bar{\mu}_1(1)x_1(1) & \cdots & \bar{\mu}_1(1)x_m(1) & \cdots & \bar{\mu}_n(1) & \bar{\mu}_n(1)x_1(1) & \cdots & \bar{\mu}_n(1)x_m(1) \\ \bar{\mu}_1(2) & \bar{\mu}_1(1)x_1(2) & \cdots & \bar{\mu}_1(2)x_m(2) & \cdots & \bar{\mu}_n(2) & \bar{\mu}_n(2)x_1(2) & \cdots & \bar{\mu}_n(2)x_m(2) \\ \vdots & \vdots & \cdots & \vdots & \cdots & \vdots & \vdots & \cdots & \vdots \\ \bar{\mu}_1(p) & \bar{\mu}_1(p)x_1(p) & \cdots & \bar{\mu}_1(p)x_m(p) & \cdots & \bar{\mu}_n(p) & \bar{\mu}_n(p)x_1(p) & \cdots & \bar{\mu}_n(p)x_m(p) \\ \vdots & \vdots & \cdots & \vdots & \cdots & \vdots & \vdots & \cdots & \vdots \\ \mu_1(P) & \bar{\mu}_1(P)x_1(P) & & \bar{\mu}_1(P)x_m(P) & & \bar{\mu}_n(P) & \bar{\mu}_n(P)x_1(P) & & \bar{\mu}_n(P)x_m(P) \end{bmatrix}$$

  et $\boldsymbol{k}$ est le vecteur des paramètres de conséquent inconnus de dimension $n(1 + m) \times 1$:

  $$\boldsymbol{k} = [k_{10}\ \ k_{11}\ \ k_{12} \ldots k_{1m}\ \ k_{20}\ \ k_{21}\ \ k_{22} \ldots k_{2m} \ldots k_{n0}\ \ k_{n1}\ \ k_{n2} \ldots k_{nm}]^T$$

- On a donc :

  $$\boldsymbol{k} = \mathbf{A}^{-1}\,\boldsymbol{y}_d \text{ (en pratique } \boldsymbol{k} = (\boldsymbol{A}^t\boldsymbol{A})^{-1}\boldsymbol{A}^t\,\boldsymbol{y}_d \text{)}$$

- Une fois le vecteur $\boldsymbol{k}$ déterminé, le vecteur de sortie du réseau $\boldsymbol{y}$ peut être calculé ainsi que le vecteur d'erreur associé, $\boldsymbol{e}$:

$$\boldsymbol{e} = \boldsymbol{y}_d - \boldsymbol{y}$$

- Lors de la passe arrière, l'algorithme de retropropagation d'erreur est appliqué pour mettre à jour les poids des antécédents des règles.

- Dans l'algorithme ANFIS de Jang, on optimise aussi bien les paramètres de antécédents que ceux des conséquents. Durant la passe avant, les paramètres des conséquents sont adaptés alors que les paramètres des antécédents sont maintenus constants ; durant la passe arrière, les rôles sont échangés.

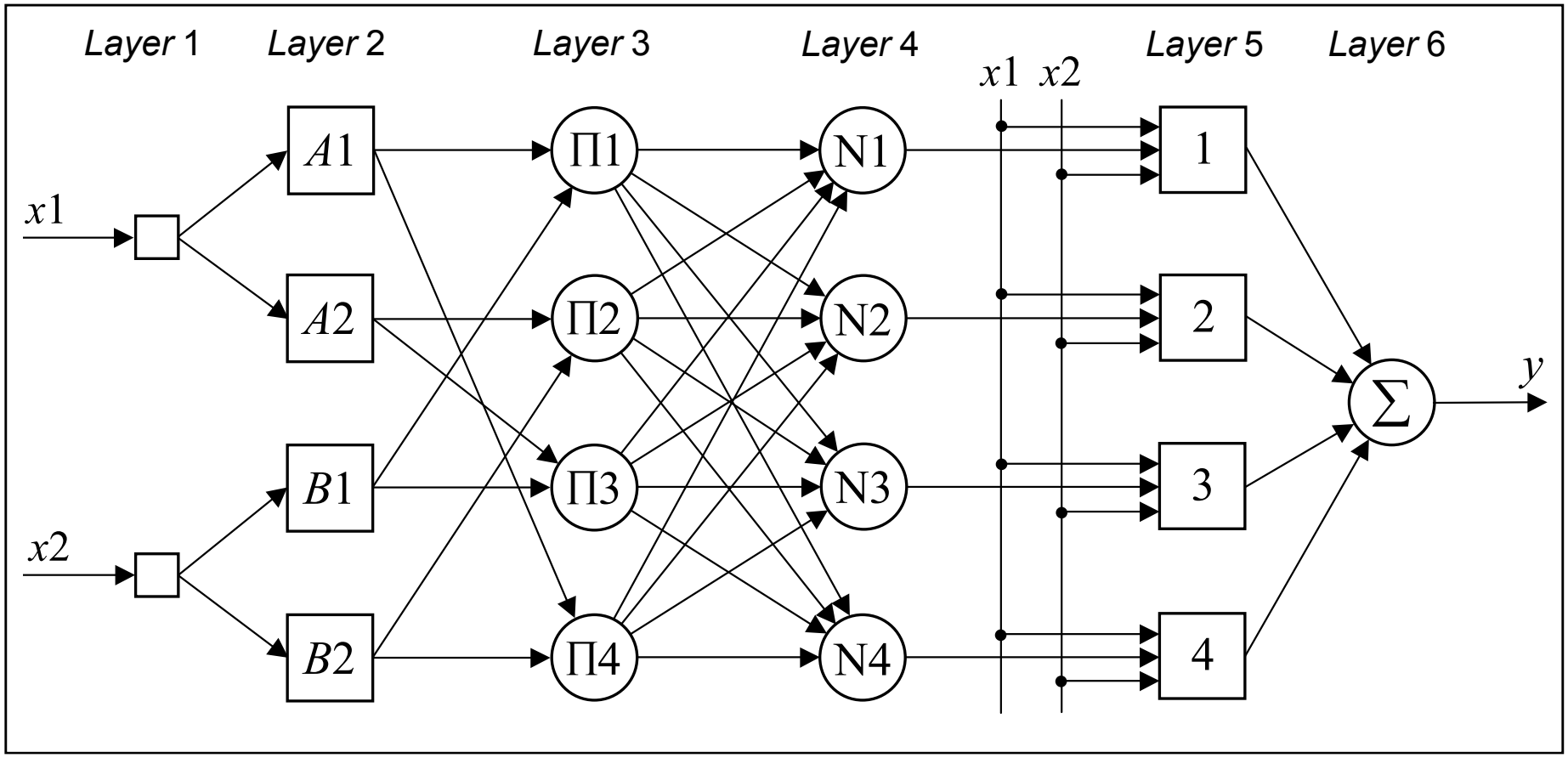
Approximation de fonctions avec ANFIS

- Ex. : suivre la trajectoire définie par la fonction non-linéaire définie par :

$$y = \frac{\cos(2\,x1)}{e^{x2}}$$

- Détermination de l'architecture :
  - Deux entrées, $x1$ and $x2$, et une sortie, y.
  - Chaque entrée possède deux valeurs linguistiques.
- Donc le réseau ANFIS possède quatre règles.

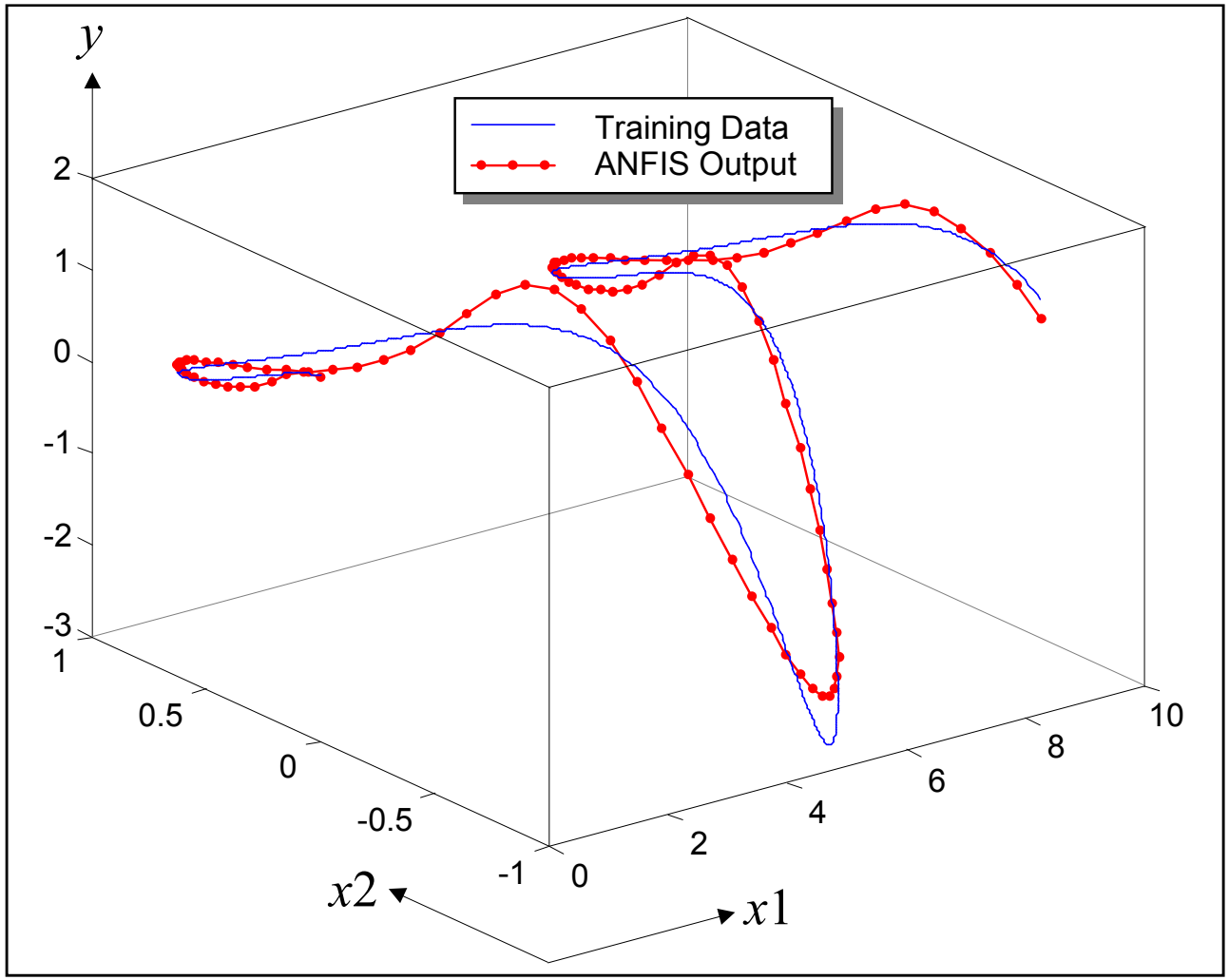
Modèle ANFIS avec quatre règles



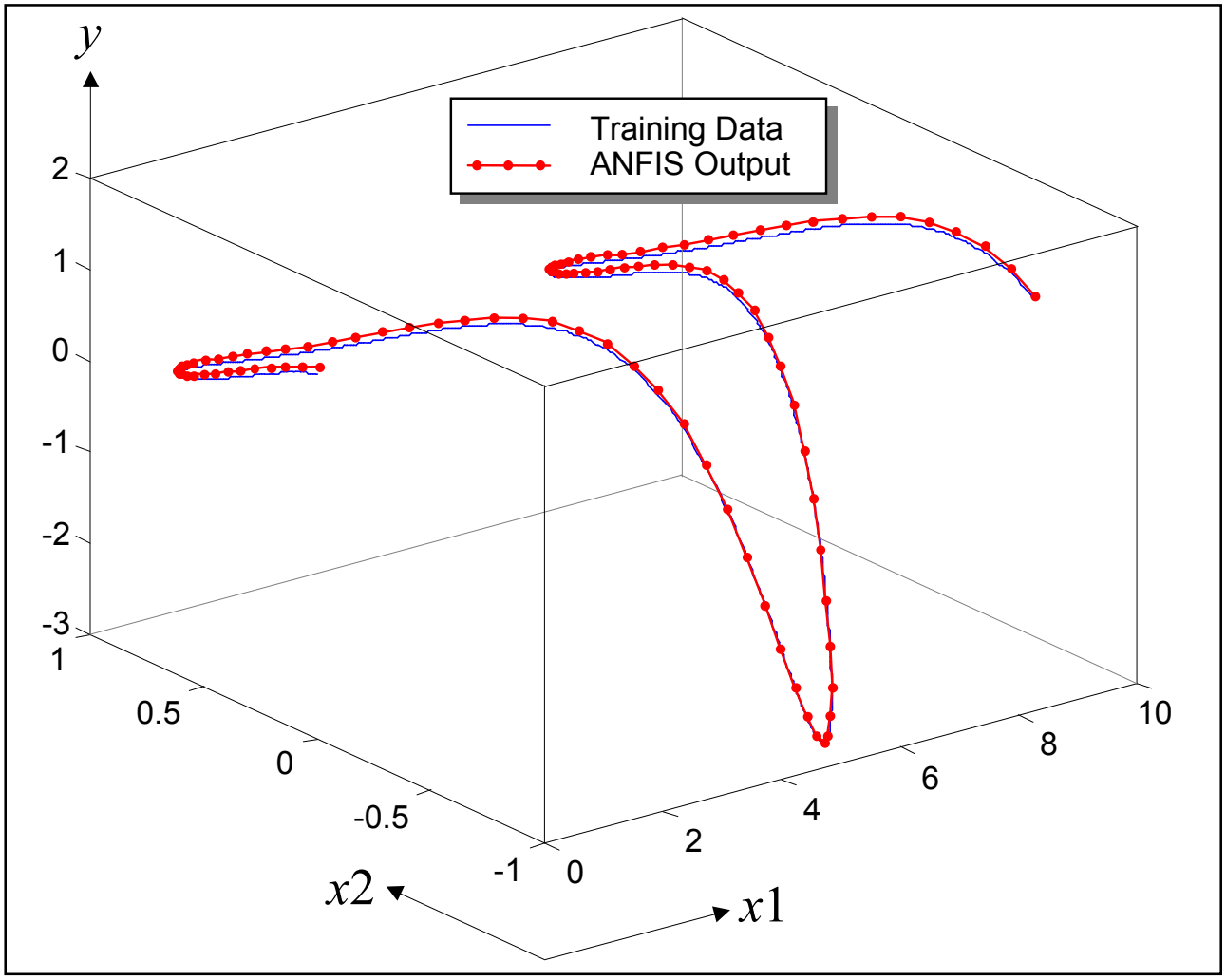
Apprentissage du réseau

- L'ensemble d'apprentissage comprend 101 échantillons représentés par 101 triplets [$x1$ $x2$ y_d].
- $X1$ va de 0 à 10 par pas de 0.1
- $X2$ = sin($x1$) pour donner des paires [$x1$ $x2$] qui sont raisonnablement distribuées.
- y_d, est déterminé en solvant l'équation.

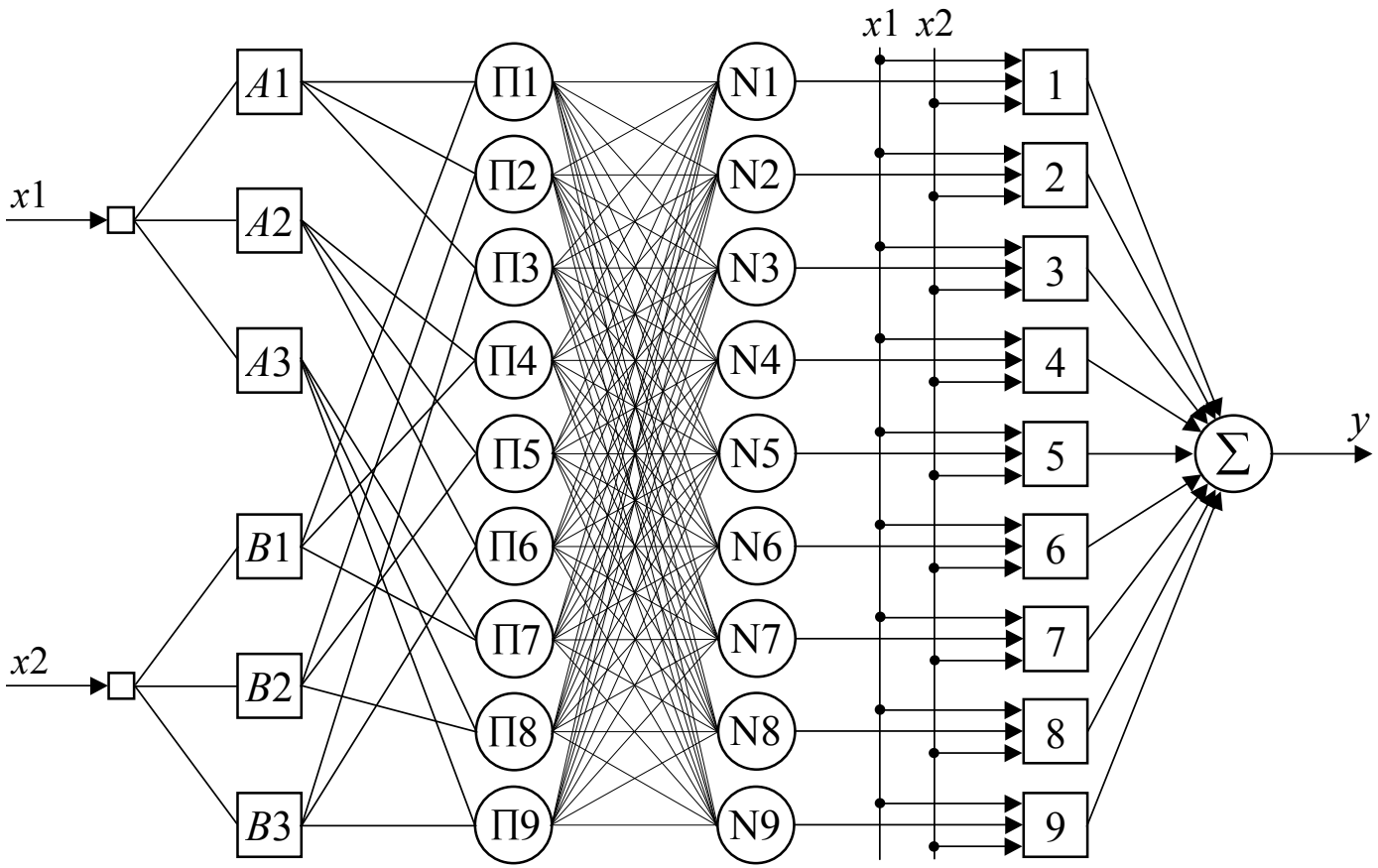
Apprentissage sur 1 période



Apprentissage sur 100 périodes

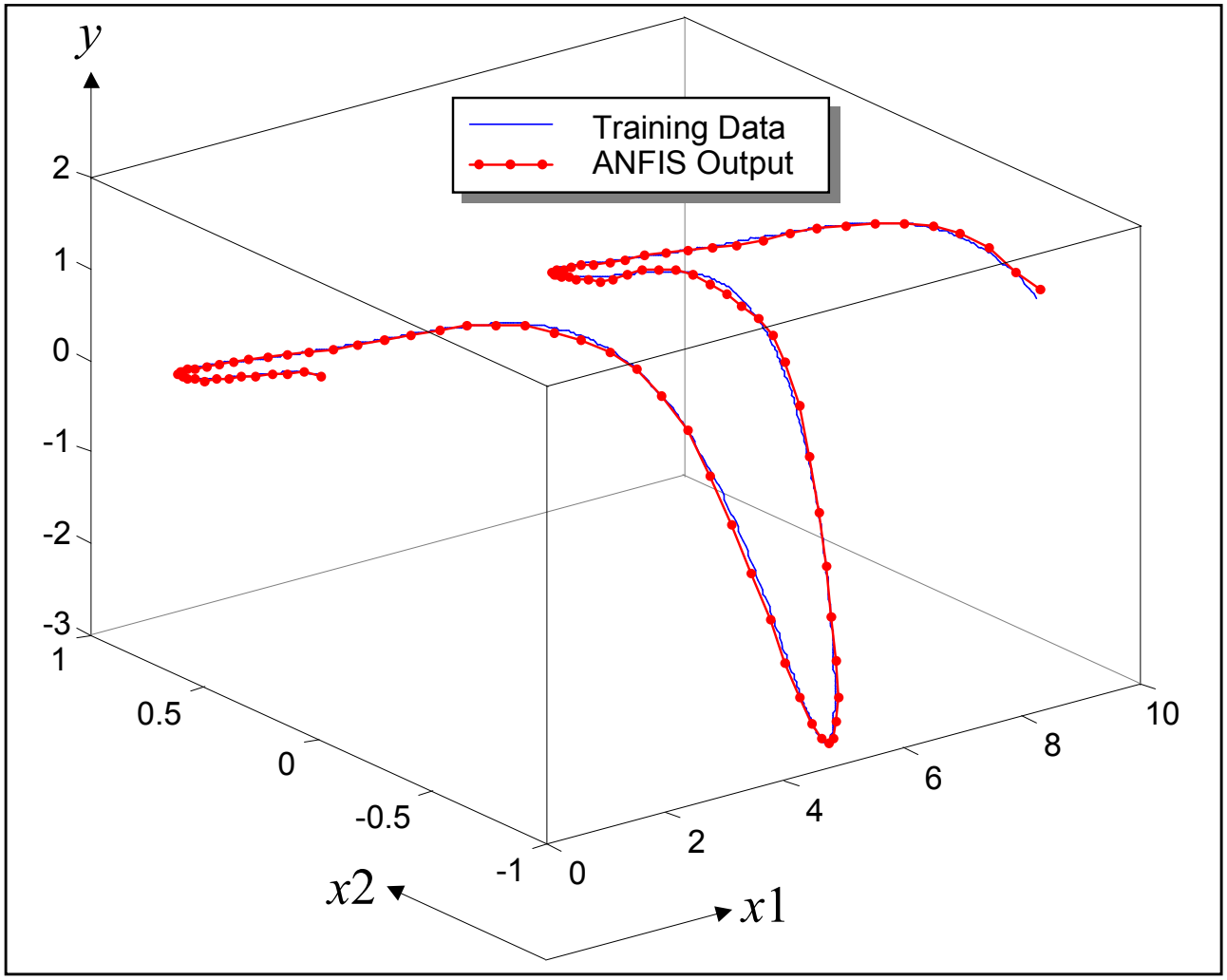


On peut améliorer la précision d'approximation en augmentant le nombre de valeurs linguisitques par entrée. Par exemple, pour 3 valeurs on obtient un réseau ANFIS à 9 règles :



Apprentissage sur 1 période

3 fonctions d'appartenance par variable d'entrée



Apprentissage sur 100 périodes

3 fonctions d'appartenance par variable d'entrée