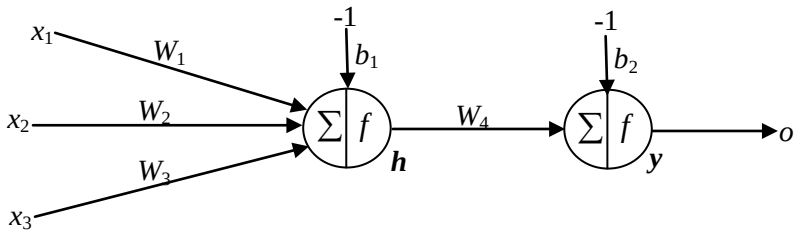
TD N°2 Identification et commande par réseau de neurone

Exercice 1 :

Soit un réseau de neurone de type perceptron multicouche avec une couche cachée d'un seul neurone, et une seule couche de sortie. La fonction d'activation utilisée pour ce cas est de type sigmoïde représenté dans la figure suivante :

.



Pour la base de donné suivante qui présente une fonction booléenne

x_1	x_2	x_3	o_d
1	1	0	1
0	1	1	1

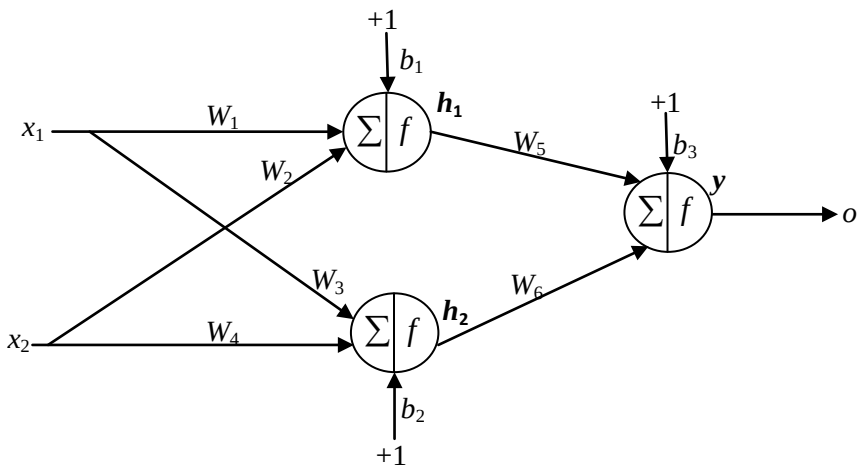
- Déterminer la valeur de l'erreur Quadratique

- Donner les poids et les biais du réseau après une époque d'apprentissage Avec les conditions initiales suivantes :

W_1= 0.5, W_2= 0.4, W_3= 0.3, W_4= 0.6, b_1=0.8, b_2=0.3.

Exercice 2 :

Soit un réseau de neurone de type perceptron multicouche avec une couche d'entrée, une cachée et une couche de sortie, La fonction d'activation utilisée pour ce cas est de type sigmoïde représenté dans la figure suivante :



Pour la base de donné suivante :

x_1	x_2	o_d
0.1	0.3	0.03

- Déterminer la valeur de l'erreur Quadratique
- En utilisant l'algorithme de rétropropagation de l'erreur donner les poids W_1,...,W_5 après une époque d'apprentissage avec les conditions initiale suivantes :

W_1	W_2	W_3	W_4	W_5	W_6	b_1	b_2	b_3
0.5	0.1	0.62	0.2	-0.2	0.3	0.4	-0.1	1.83

Et un taux d'apprentissage $\eta = 0.01$.

Exercice 3 :

Soit le système discret donné par la fonction de transfert suivante :

$$H(z) = z^{-1}\frac{b_1 + b_2 z^{-1}}{1 - a_1 z^{-1}}$$

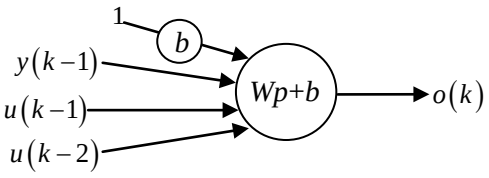
Avec

$b_1 = 1$, $b_2 = -0.8$, $a_1 = -0.5$

et l'équation de récurrence liant la sortie $y(k)$ à l'entrée $u(k)$

$$y(k) = a_1\, y(k-1) + b_1\, u(k-1) + b_2\, u(k-2)$$

Le but de cet exercice est l'identification des paramètres (b_1, b_2, a_1) on utilisant une base d'apprentissage qui contient 500 points d'entrés sortie ($u(k)$,$y(k)$) et un résaux de neurone exprimé par la figure suivante :



Les entrés du réseau $p = \left[y(k-1) \quad u(k-1) \quad u(k-2) \right]$

1- Déterminer le critère à minimiser
2- Déterminer la loi d'adaptation des poids (W, b)
3- Ecrire un programme en Matlab qui permet d'effectuer cette identification on utilisant un signal d'entré u riche en fréquence présenter par le sous programme suivante :

```
u=(-1).^(1:10);

for i=11:500
   u(i)=-u(i-7)*u(i-10);
end
```

Exercice 4 :

On considère un système ayant la fonction de transfert suivante :

$$H(z) = \frac{z^{-1}\left(0.3 + 0.05\, z^{-1}\right)}{1 - 0.7\, z^{-1}}$$

Auquel on appliquera un signal de commande $u(t)$ riche en fréquence que l'on superpose à une valeur constante égal à 5 correspondant à un point de fonctionnement du processus par le sous programme suivante :

```
u=5*ones(1,200);
sig=(-1).^(1:10);

for i=11:200
   sig(i)=-sig(i-7)*sig(i-10);
end
u=u+sig ;
```

À partir des données $u(k)$, $y(k)$, on calcule les variations $\Delta u(k)=u(k)-u(k-1)$ et $\Delta y(k)=y(k)-y(k-1)$ et avec l'utilisation les fonction d'activation type sigmoïde unipolaire et pour éviter la saturation des poids du réseau on doit normaliser les différents signaux $u(k)$, $y(k)$, $\Delta u(k)$, $\Delta y(k)$ comme suit :

a_y=0.35, *b_y*=-1.58, *a_u*=0.4, *b_u*=-1.5, *a_dy*=0.6836, *b_dy*=0.5, *a_du*=0.2, *b_du*=0.5

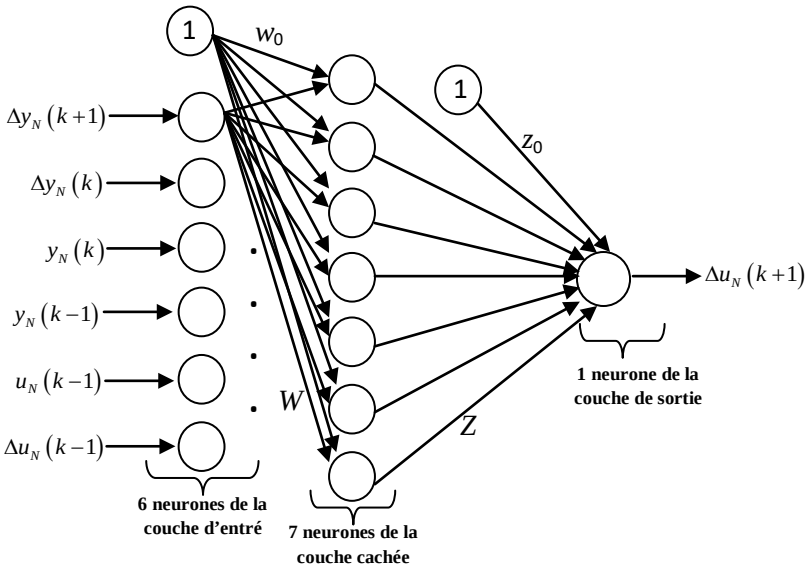
*du_N=a_du*du+b_du ;*

*dy_N=a_dy*dy+b_dy;*

*u_N=a_u*u+b_u ;*

*y_N=a_y*y+b_y ;*

Notre réseau de neurone contient 6 neurones au couche d'entré et 7 neurone au couche caché et 1 neurone au couche de sortie, a pour but de fournir, à chaque instant d'échantillonnage. La variation de commande $\Delta u(k) = u(k) - u(k-1)$ comme montre la figure suivante :

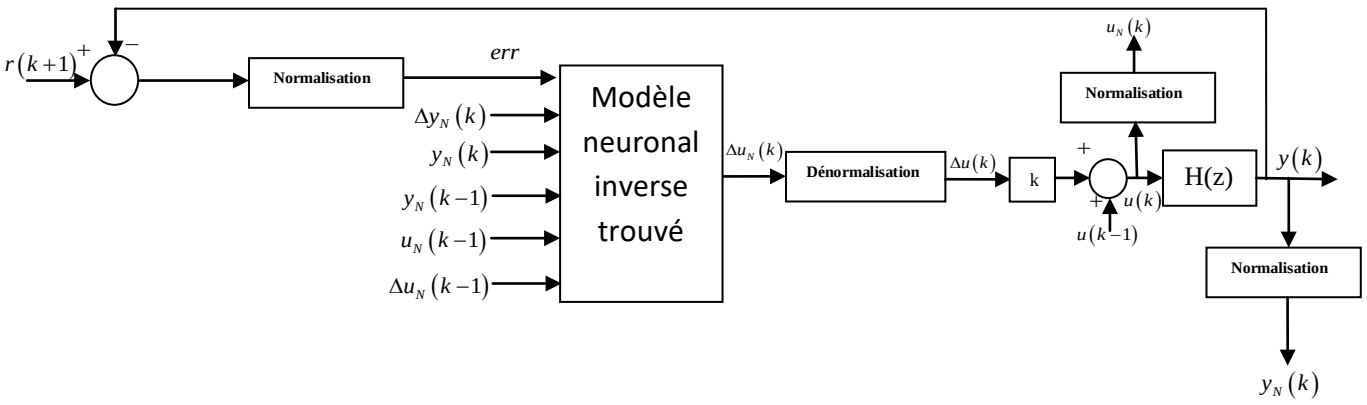


1- Déterminer le critère à minimiser
2- Déterminer la loi d'adaptation des poids (W, w_0, Z, z_0)
3- Ecrire un programme en Matlab qui permet d'effectuer l'identification inverse du système

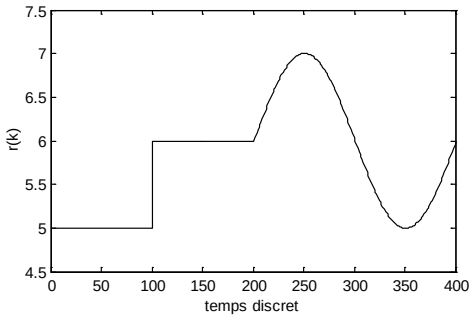
Remarque : pour la fonction d'activation de la couche cachée et la couche de sortie on prend

$$f(x) = \frac{1}{1+e^{-x}}$$

Pour réaliser un système de commande avec le modèle inverse neuronal trouvé, la première cellule de la couche d'entré recevra la valeur normalisé de l'écart entre la valeur future *r*(k+1) de la consigne et la valeur courante de la sortie *y*(t) du processus comme montre le schéma suivant :



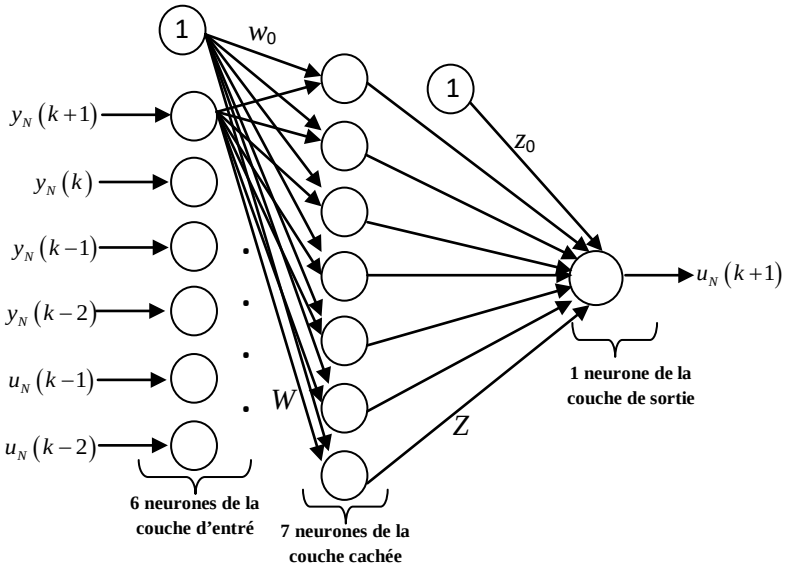
1- Ecrire un programme en Matlab qui permet de réalise ce système de commande en prenant le signale de référence qui est présenté par la figure suivante :




```
palier1=5*ones(1,100);
palier2=6*ones(1,100);

r1=[palier1 palier2];
t=0:pi/100:2*pi;
r2=6+sin(t);
r=[r1 r2];
```

Une autre structure de réseau modélisant le modèle inverse du processus est la suivante :



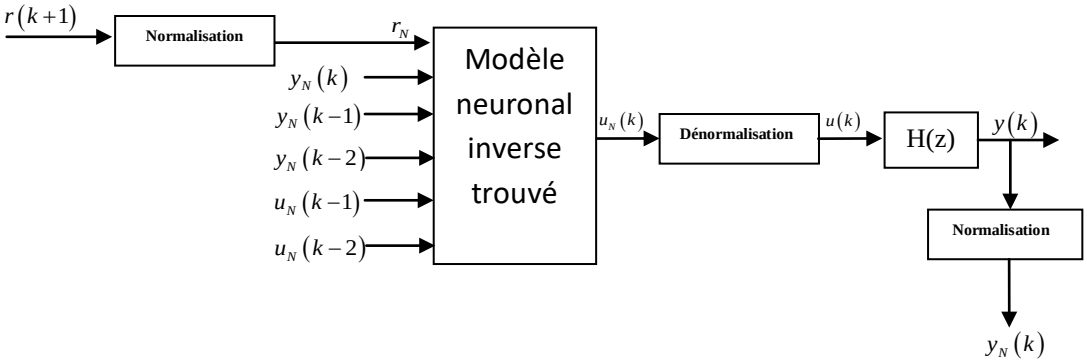
Les signaux d'entrée ne contiennent pas de variation et en sortie on cherche à obtenir directement le signal de commande normalisé.

1- Ecrire un programme en Matlab qui permet d'effectuer l'identification inverse du système

Remarque : pour la fonction d'activation de la couche cachée et la couche de sortie on prend

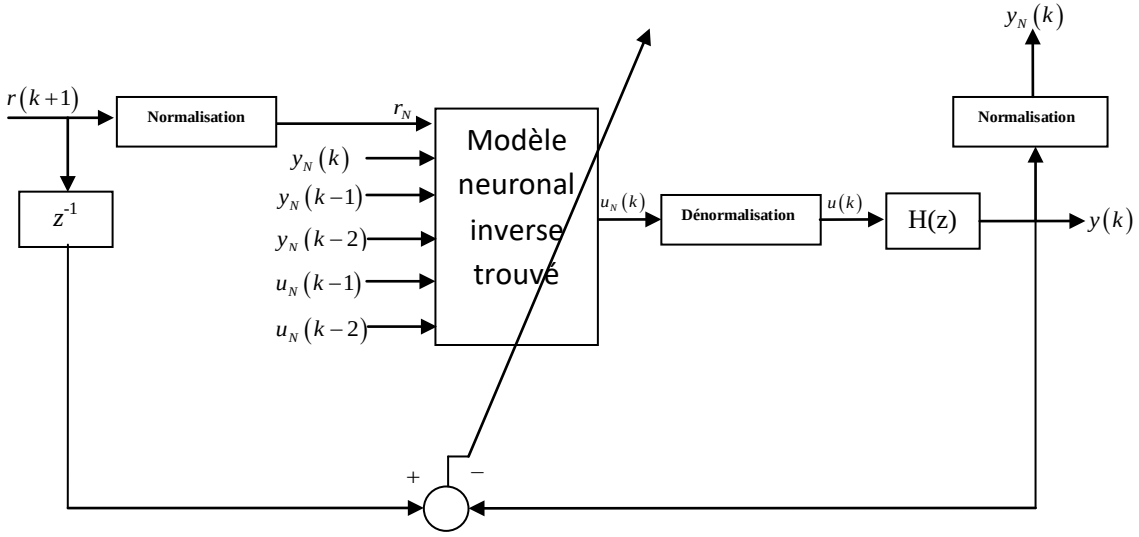
$$f(x)=\frac{1}{1+e^{-x}}$$

Dans la loi de commande, la première cellule de la couche d'entrée reçoit la valeur future du signal de consigne, après la dénormalisation la sortie du réseau constituera le signal de commande qui sera directement appliqué au processus comme montre la figure suivante :



1- Ecrire un programme en Matlab qui permet de réalise ce système de commande en prenant le même signal de référence du cas précédant

Pour améliorer les résultats précédant, on désire d'appliquer la commande neuronal adaptative, on rétropropage l'erreur de poursuite à travers le réseau pour modifier les différents poids et biais, afin d'annuler l'erreur en régime permanant le schéma de la commande adaptative est présenter par la figure suivante.



1- Ecrire un programme en Matlab qui permet de réalise ce système de commande adaptative en prenant le même signal de référence du cas précédant

<u>Exercice 5 :</u>

Ecrire un programme en Matlab qui permet l'approximation de la fonction $sin(x)$ avec $-2 \le x \le 2$ on utilisant neural network toolbox

Quelque fonction Matlab à utilisé :

1- Calcul de la sortie du model discret :

$$H(z) = \frac{z^{-1}\left(0.3 + 0.05\, z^{-1}\right)}{1 - 0.7\, z^{-1}}$$

```
num=[0.3 0.05];
den=conv([1 0],[1 -0.7]);
y=(dlsim(num,den,u))';
```

2- **newff**: création d'un réseau de neurone
net.trainParam.epochs: définition le nombre d'itération
net.trainParam.goal: définition de l'erreur
train : apprentissage du réseau

3- logsig : fonction type sigmoïde $logsig(x) = \frac{1}{1+e^{-x}}$