

Annexe A

```

% =====
% ===== programme d'identification série parallèle de système1 =====
% ===== un réseaux de neurones à un seul couche cachée =====
% =====
clear all ;close all;clc
% identification sérer parallèle
% pour utiliser ce programme donner les variables suivants
Nvf=2; % nombre des variables
Nce=4; % nombre des neurones dans la couche d'entrée
Ncc=3; % nombre des neurones dans la couche cachée

w1=randn(Nvf,Nce);
w2=randn(Nce,Ncc); % les poids
w3=randn(1,Ncc);

w01=randn(1,Nce); w01j=zeros(size(w01)); % biais
w02=randn(1,Ncc); w02j=zeros(size(w02));
w03=randn(1,1); w03j=zeros(size(w03));
w1j=zeros(size(w1)); % initialisation
w2j=zeros(size(w2));
w3j=zeros(size(w3));
%-----%
% 0.01< mieu <0.25
mieu=0.02; % pas d'apprentissage
beta=0.00011 ; % moment
%-----%

%Génération des entrées de réseaux
for k=1:100
    u(k)=sin(2*pi*k/40);
end
yr(1)=0;
for k=1:99
    yr(k+1)=(yr(k)/(1+yr(k)^2))+u(k)^3;
end
X=[u,yr]; % vecteur d'entrée
[n,m]=size(X);
itiration =1; % nombre d'itération

%===== démarrage de programme d'apprentissage et rétropropagation =====
%=====
erreur=1;
while erreur>=10^-2
    for i=1:n
        sort_ds(i)=yr(i);
        x=X(i,:);
        %=====calcule la sort_act =====
        for a=1:Nvf
            mix_ent1(a,:)=x(a).*w1(a,:);
        end
    end
end

```

```

%-----%
for r=1:Nce
    s1=sum(mix_ent1(:,r))+w01(r);
    vec_sort1(r)=(1-exp(-s1))/(1+exp(-s1)); % fonction sigmoid
    der_sort1(r)=2*(exp(-s1))/(1+exp(-s1))^2; % dérivier sigmoid
end
%===== calcule les entrée de la couche 2 =====%
for b=1:Nce
    vec_ent2(b,:)=vec_sort1(b).*w2(b,:);
end

%===== calcule la sortie de la couche cachée =====%
for c=1:Ncc
    s2=sum(vec_ent2(:,c))+w02(c);
    vec_sort2(c)=(1-exp(-s2))/(1+exp(-s2));
    der_sort2(c)=2*(exp(-s2))/(1+exp(-s2))^2;
end
s3=sum(vec_sort2.*w3)+w03;
der_sort3=1;
%===== la sortie de réseaux est sort_act =====%
sort_act(i)=s3;
% calcul l'erreur
er=sort_ds-sort_act;
% adaptation des poids
% calcul les terme d'erreur "sigma"
sigma3=er(i)*der_sort3;
sigma2=der_sort2.*(sigma3.*w3);
for d=1:Nce
    som_w2(d)=sum(sigma2.*w2(d,:));
end

sigma1=som_w2.*der_sort1;
%=====
%===== adaptation des poids =====%
for q=1:Nce
    new_w1(:,q)=w1(:,q)+((mieu*sigma1(q))*x')+beta*(w1(:,q)-w1j(:,q));
    new_w01(q)=w01(q)+(mieu*sigma1(q))+beta*(w01(q)-w01j(q));
end
%-----%
for g=1:Ncc
    new_w2(:,g)=w2(:,g)+((mieu*sigma2(g)).*vec_sort1')+beta*(w2(:,g)-w2j(:,g));
    new_w02(g)=w02(g)+(mieu*sigma2(g))+beta*(w02(g)-w02j(g));
end
%-----%
new_w3=w3+(mieu*sigma3).*vec_sort2+beta*(w3-w3j);
new_w03=w03+(mieu*sigma3)+beta*(w03-w03j);
%=====
w1j=w1;          w01j=w01;
w2j=w2;          w02j=w02;
w3j=w3;          w03j=w03;
w1=new_w1;       w01=new_w01;
w2=new_w2;       w02=new_w02;
w3=new_w3;       w03=new_w03;

end

```

```
%===== %
    erreur=sum(er.^2)
%===== %

E(itiration)=erreur;

    figure(1);plot(E); title('errure')
    figure(2);
    hold on
    plot(sort_ds,'b'); plot(sort_act,'r');grid on
    pause(0.01)
    clf
    itiration = itiration +1 ;
end

disp('les valeur des poids')
format long;
w1
w2 % affichage des poids
w3
format;
disp('nombre ditiration')
itiration
figure(2);hold on          % plot la sortie désirer et la sortie de réseaux
plot(sort_ds,'b')
plot(sort_act,'r')
hold off
```

Annexe B

```

%=====
%===== programme d'identification série parallèle de système 3 =====
%===== un réseaux de neurones à deux couches cachée =====
%=====
clear all;close all;clc
%Génération des entrées de réseaux
for k=1:500
    u(k)=sin(2*pi*k/250);
end
for k=501:800
    u(k)=0.6*sin(2*pi*k/250)+0.2*cos(2*pi*k/25);
end
yr(1)=0;
for k=1:799
    yr(k+1)=u(k)+yr(k)/(1+yr(k)^2);
end
X=[u;yr];          % vecteur d'entrée
SD=yr;             % la sortie désirée
mieu=0.2;          %le pas d'apprentissage
betta=0.00001;     % moment

% programme a deus couches cachée

Nvf=2;             % nombre des entrée
Nce=7;             % nombre des neurones dans la couche d'entrée
Ncc1=13;           % nombre des neurones dans la couche cachée N 1
Ncc2=5;            % nombre des neurones dans la couche cachée N 2
Ncs=1;            % nombre des neurones dans la couche de sortie

%=====
w1=randn(Nce,Nvf);      w01=randn(Nce,1);
w2=randn(Ncc1,Nce);     w02=randn(Ncc1,1);
w3=randn(Ncc2,Ncc1);    w03=randn(Ncc2,1);
w4=randn(Ncs,Ncc2);     w04=randn(Ncs,1);
w1j=zeros(size(w1));    % initialisation des poids et biais
w2j=zeros(size(w2));
w3j=zeros(size(w3));
w4j=zeros(size(w4));
w01j=zeros(size(w01));
w02j=zeros(size(w02));
w03j=zeros(size(w03));
w04j=zeros(size(w04));
itiration=1;
erreur=1;
temp=cputime;          % calcule le tempsd'excution de programme
[n,m]=size(X);

```

```

while erreur >=10^-5
for i=1:m
    T(i)=i;
    X1=X(:,i);    % les entrées de réseaux

    for a=1:Nvf
        mix_ent1(a,:)=X1(a).*w1(:,a)';
    end

    %===== % la couche d'entrée %=====
    for b=1:Nce
        s1=sum(mix_ent1(:,b))+w01(b);
        vec_sort1(b)=(1-exp(-s1))/(1+exp(-s1));    %sigmoid;
        der_sort1(b)=2/(2+exp(-s1)+exp(s1));    %der_sigmoid;
    end
    %===== % la couche cachée N1 %=====
    for c=1:Nce
        mix_ent2(c,:)=vec_sort1(c).*w2(:,c)';
    end

    for d=1:Ncc1
        s2=sum(mix_ent2(:,d))+w02(d);
        vec_sort2(d)=(1-exp(-s2))/(1+exp(-s2));    %sigmoid;
        der_sort2(d)=2/(2+exp(-s2)+exp(s2));    %der_sigmoid;
    end
    %===== % la couche cachée N2 %=====
    for e=1:Ncc1
        mix_ent3(e,:)=vec_sort2(e).*w3(:,e)';
    end

    for f=1:Ncc2
        s3=sum(mix_ent3(:,f))+w03(f);
        vec_sort3(f)=(1-exp(-s3))/(1+exp(-s3));    %sigmoid;
        der_sort3(f)=2/(2+exp(-s3)+exp(s3));    %der_sigmoid;
    end
    %===== % la couche de sortie %=====
    for g=1:Ncc2
        mix_ent4(g,:)=vec_sort3(g).*w4(:,g)';
    end

    s4=sum(mix_ent4)+w04;
    vec_sort4=s4/(1+exp(-s4));    %sigmoid;
    der_sort4=1;%2/(2+exp(-s4)+exp(s4));    %der_sigmoid;

    %===== % la sortie obtenue %=====
    sort_act(i)=vec_sort4;    %la sortie actuelle de réseaux
    %===== % la sortie désirée %=====
    sort_ds(i)=SD(i);    % la sortie désirée de réseaux

    %===== % calcule l'erreur %=====
    er=sort_ds-sort_act;

```

```

%=====calcul les terme d'erreur "sigma" %=====
sigma4=er(i)*der_sort4; % terme d'erreure dans Ncs
sigma3=der_sort3.*(sigma4.*w4); % terme d'erreur dans Ncc2
for k=1:Ncc1
    som_w3(k)=sum(sigma3*(w3(:,k)));
end
sigma2=der_sort2.*(som_w3); % terme d'erreur dans Ncc1
for L=1:Nce
    som_w2(L)=sum(sigma2*w2(:,L));
end
sigma1=der_sort1.*som_w2; % terme d'erreur dans Nce

%=====adaptation des poids %=====
for M=1:Nce
    new_w1(M,:)=w1(M,:)+((mieu*sigma1(M))*X1)+beta*(w1(M,:)-w1j(M,:));
    new_w01(M)=w01(M)+(mieu*sigma1(M))+beta*(w01(M)-w01j(M));
end
for N=1:Ncc1
    new_w2(N,:)=w2(N,:)+((mieu*sigma2(N)).*vec_sort1)+beta*(w2(N,:)-w2j(N,:));
    new_w02(N)=w02(N)+(mieu*sigma2(N))+beta*(w02(N)-w02j(N));
end
for o=1:Ncc2
    new_w3(o,:)=w3(o,:)+((mieu*sigma3(o)).*vec_sort2)+beta*(w3(o,:)-w3j(o,:));
    new_w03(o)=w03(o)+(mieu*sigma3(o))+beta*(w03(o)-w03j(o));
end
new_w4=w4+(mieu*sigma3).*vec_sort3+beta*(w4-w4j);
new_w04=w04+(mieu*sigma4)+beta*(w04-w04j);
%=====
w1j=w1; w01j=w01;
w2j=w2; w02j=w02;
w3j=w3; w03j=w03;
w4j=w4; w04j=w04;
w1=new_w1; w01=new_w01;
w2=new_w2; w02=new_w02;
w3=new_w3; w03=new_w03;
w4=new_w4; w04=new_w04;
end
%=====
erreur=sum(er.^2)
%=====
figure(1);
hold on
plot(sort_ds,'b')
plot(sort_act,'r');grid on
pause(0.1)
clf

E(itiration)=erreur;
itiration=itiration+1;
end
figure(1);hold on
plot(T,SD,'b',T,sort_act,'r'),grid ,legend('système','modèle')
figure(2);plot(er);grid
%=====
disp('le temps d"excution en minut')
temps=(cputime-temp)/60
%=====

```