

<b>Uniwersytet Zielonogórski</b>	Wykonali:	Grupa:	Nr ćwiczenia: 4	Ocena:
<b>Laboratorium systemów obliczeń inteligentnych</b>				
Temat ćwiczenia: Identyfikacja systemów dynamicznych za pomocą wielowarstwowych sieci neuronowych.		Prowadzący:	Data wyk. ćw.	Data odd. spr.

**Zadanie 1**

Zaprojektować sieć neuronową oraz przeprowadzić jej uczenie w celu modelowania obiektu dynamicznego opisanego następującym wzorem

$$y(k) = f[y(k-1), y(k-2), y(k-3), u(k-1), u(k-2)],$$

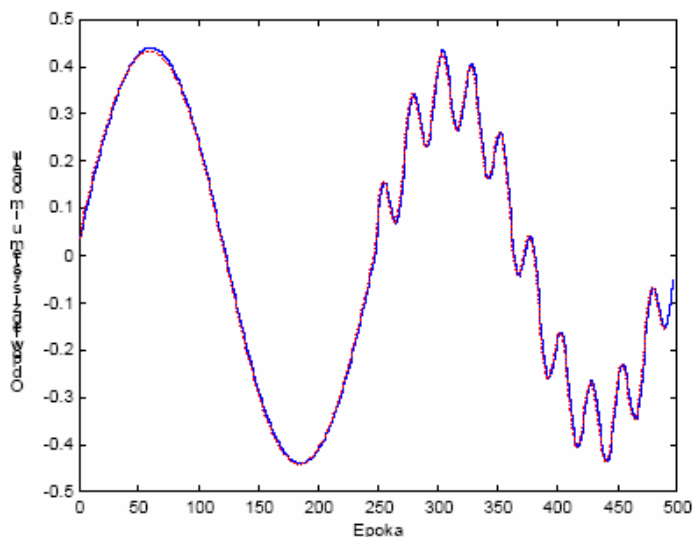
gdzie funkcja nieliniowa  $y(k) = f(\bullet)$  jest określona wzorem

$$f(x_1, x_2, x_3, x_4, x_5) = \frac{x_1 x_2 x_3 x_5 (x_3 - 1) + x_4}{1 + x_2^2 + x_3^2}$$

- W procesie identyfikacji zastosować neuronowy model równoległy
- W procesie identyfikacji zastosować neuronowy model szeregowo-równoległy (porównać otrzymane rezultaty)
- Zbadać wpływ doboru architektury sieci neuronowej na jakość identyfikacji (liczba warstw sieci, liczba neuronów w warstwie, dobór funkcji aktywacji neuronów)

Do testowania modelu wykorzystać sygnał

$$\begin{array}{ll} \sin(2\pi k / 250) & \text{dla } k > 250 \\ 0.8 \sin(2\pi k / 250) + 0.2 \sin(2\pi k / 25) & \text{dla } k \leq 250 \end{array}$$



**Ad a)**

Do nauki sieci wykorzystaliśmy poniższy skrypt

```

siec=newff([-1 1;-1 1;-1 1;-1 1;-1 1],[5
1],{'tansig','tansig'},'traingdx');

siec.trainparam.epochs=2000; %maksymalna liczba iteracji
siec.trainparam.goal=0;      %zadana jakosc
siec.trainparam.show=200;    %czestotliwosc wyswietlania

%generowanie losowego sygnalu wejsciowego
u=rand(1,100)*(0.5)*2;

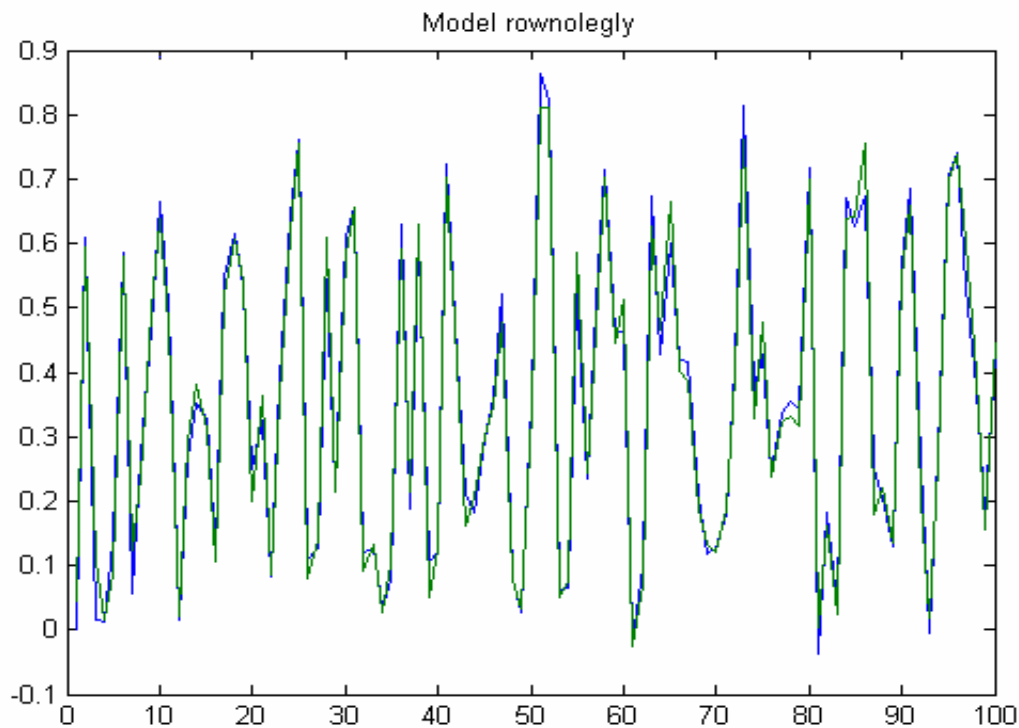
%przyjete wartosci poczatkowe
wart_pocz=0;
x1=wart_pocz;
x2=wart_pocz;
x3=wart_pocz;
x4=wart_pocz;
x5=wart_pocz;

%obliczanie wartosci funkcji y(k)
y=[];
for i=1:length(u),
    y=[y,(x1*x2*x3*x5*(x3-1)+x4)/(1+x2^2+x3^2)];
    x3=x2;
    x2=x1;
    x1=y(i);
    x5=x4;
    x4=u(i);
end
x1=[wart_pocz,y(1:99)];
x2=[wart_pocz,wart_pocz,y(1:98)];
x3=[wart_pocz,wart_pocz,wart_pocz,y(1:97)];
x4=[wart_pocz,u(1:99)];
x5=[wart_pocz,wart_pocz,u(1:98)];

%uczenie sieci
siec.trainparam.lr=maxlinlr([x1;x2;x3;x4;x5],y);
siec=train(siec,[x1;x2;x3;x4;x5],y);

%model rownolegly
x1=wart_pocz;
x2=wart_pocz;
x3=wart_pocz;
x4=wart_pocz;
x5=wart_pocz;
y_mod=[];
for i=1:length(u),
    y_mod=[y_mod,sim(siec,[x1;x2;x3;x4;x5])];
    x3=x2;
    x2=x1;
    x1=y_mod(i);
    x5=x4;
    x4=u(i);
end
figure;
plot([1:100],y,[1:100],y_mod);
title 'Model rownolegly';

```



Rys.1 Uczenie modelu równoległego nieliniowego systemu dynamicznego

---- sygnał uczący  
 ---- odpowiedź sieci

Do testowania sieci wykorzystaliśmy poniższy skrypt:

```
%generowanie sygnału testującego
u_test=[sin(2*pi*[0:249]/250),0.8*sin(2*pi*[250:499]/250)+0.2*sin(2*pi*[250:499]/25)];

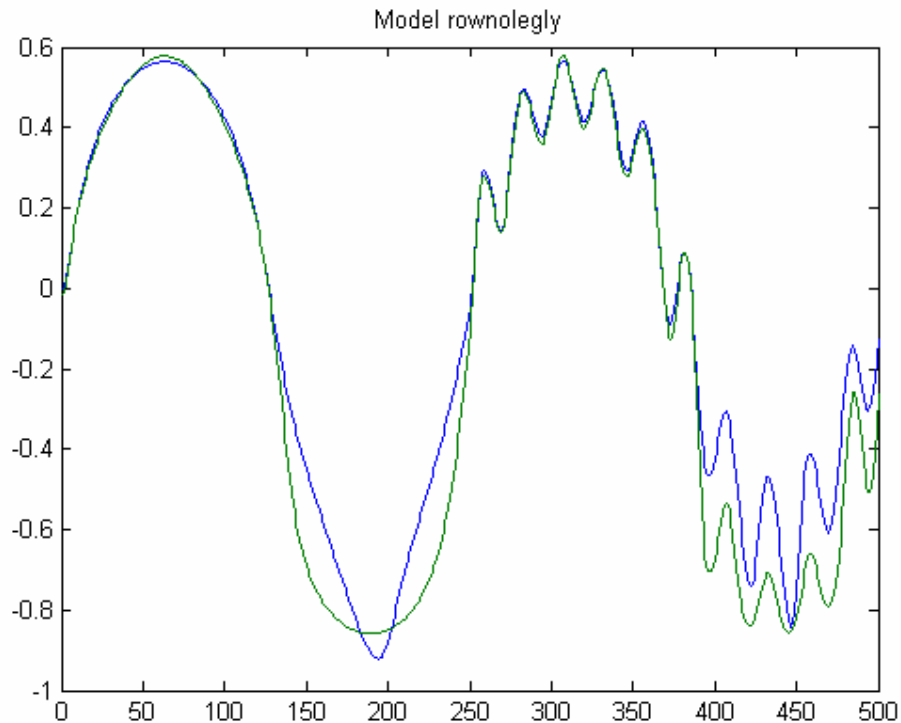
%przyjete wartosci poczatkowe
wart_pocz=0;
x1=wart_pocz;
x2=wart_pocz;
x3=wart_pocz;
x4=wart_pocz;
x5=wart_pocz;

%obliczanie wartosci funkcji
y=[];
for i=1:length(u_test),
    y=[y,(x1*x2*x3*x5*(x3-1)+x4)/(1+x2^2+x3^2)];
    x3=x2;
    x2=x1;
    x1=y(i);
    x5=x4;
    x4=u_test(i);
end
x1=[wart_pocz,y(1:499)];
x2=[wart_pocz,wart_pocz,y(1:498)];
x3=[wart_pocz,wart_pocz,wart_pocz,y(1:497)];
x4=[wart_pocz,u_test(1:499)];
```

```
x5=[wart_pocz,wart_pocz,u_test(1:498)];

%model rownolegly
x1=wart_pocz;
x2=wart_pocz;
x3=wart_pocz;
x4=wart_pocz;
x5=wart_pocz;
y_mod=[];
for i=1:length(u_test),
    y_mod=[y_mod,sim(siec,[x1;x2;x3;x4;x5])];
    x3=x2;
    x2=x1;
    x1=y_mod(i);
    x5=x4;
    x4=u_test(i);
end
figure;
plot([1:500],y,[1:500],y_mod);
title 'Model rownolegly';
```

Wynik testu przedstawia poniższy rysunek:



Rys.2 Testowanie modelu równoległego nieliniowego systemu dynamicznego

---- sygnał uczący  
---- odpowiedź sieci

**Ad b)**

Do nauki sieci wykorzystaliśmy poniższy skrypt

```

siec=newff([-1 1;-1 1;-1 1;-1 1;-1 1],[5
1],{'tansig','tansig'},'traingdx');

%ustawienie parametrow uczacych
siec.trainparam.epochs=2000; %maksymalna liczba iteracji
siec.trainparam.goal=0; %zadana jakosc
siec.trainparam.show=200; %czestotliwosc wyswietlania

%generowanie losowego sygnalu wejsciowego
u=rand(1,100)*(0.5)*2;

%przyjete wartosci poczatkowe
wart_pocz=0;
x1=wart_pocz;
x2=wart_pocz;
x3=wart_pocz;
x4=wart_pocz;
x5=wart_pocz;

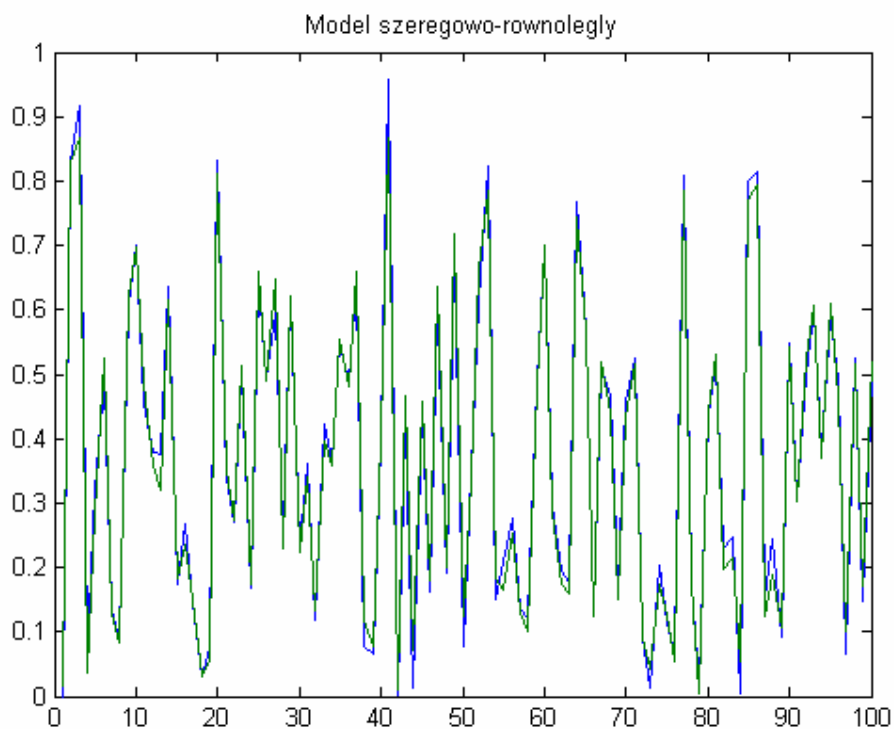
%obliczanie wartosci funkcji y(k)
y=[];
for i=1:length(u),
    y=[y,(x1*x2*x3*x5*(x3-1)+x4)/(1+x2^2+x3^2)];
    x3=x2;
    x2=x1;
    x1=y(i);
    x5=x4;
    x4=u(i);
end
x1=[wart_pocz,y(1:99)];
x2=[wart_pocz,wart_pocz,y(1:98)];
x3=[wart_pocz,wart_pocz,wart_pocz,y(1:97)];
x4=[wart_pocz,u(1:99)];
x5=[wart_pocz,wart_pocz,u(1:98)];

%uczenie sieci
siec.trainparam.lr=maxlinlr([x1;x2;x3;x4;x5],y);
siec=train(siec,[x1;x2;x3;x4;x5],y);

%model szeregowo-rownolegly
y_mod=sim(siec,[x1;x2;x3;x4;x5]);
figure;
plot([1:100],y,[1:100],y_mod);
title 'Model szeregowo-rownolegly';

```

Wynik nauczania sieci przedstawia poniższy rysunek:



Rys.3 Uczenie modelu szeregowo-równoległego nieliniowego systemu dynamicznego

---- sygnał uczący  
 ---- odpowiedź sieci

Do testowania sieci wykorzystaliśmy poniższy skrypt:

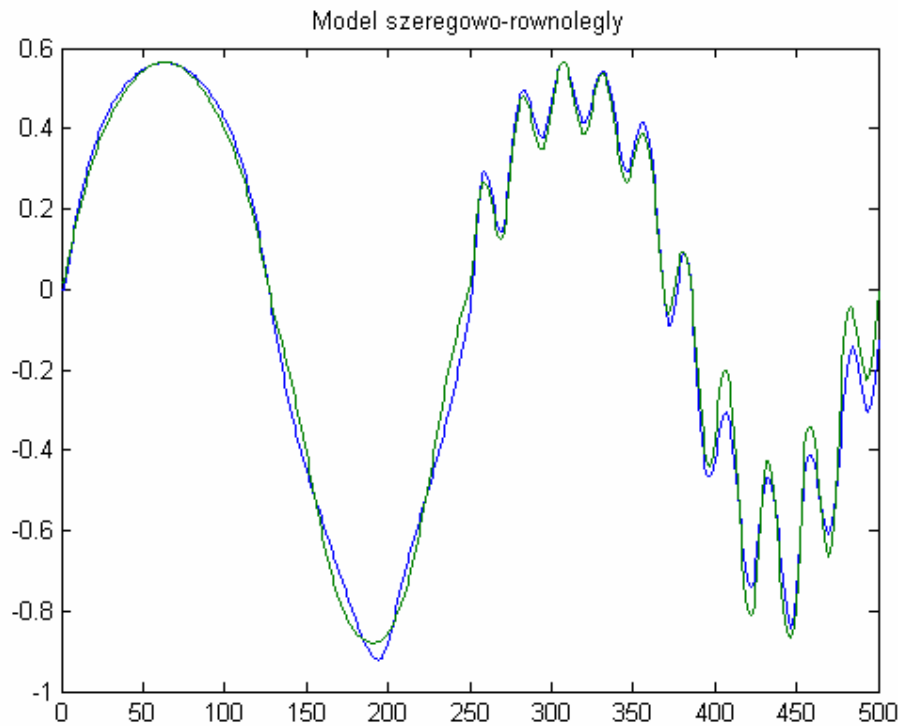
```
%testowanie sieci
%generowanie sygnału testującego
u_test=[sin(2*pi*[0:249]/250),0.8*sin(2*pi*[250:499]/250)+0.2*sin(2*pi*[250:499]/25)];

%przyjete wartosci poczatkowe
wart_pocz=0;
x1=wart_pocz;
x2=wart_pocz;
x3=wart_pocz;
x4=wart_pocz;
x5=wart_pocz;

%obliczanie wartosci funkcji
y=[];
for i=1:length(u_test),
    y=[y, (x1*x2*x3*x5*(x3-1)+x4)/(1+x2^2+x3^2)];
    x3=x2;
    x2=x1;
    x1=y(i);
    x5=x4;
    x4=u_test(i);
end
x1=[wart_pocz,y(1:499)];
x2=[wart_pocz,wart_pocz,y(1:498)];
```

```
x3=[wart_pocz,wart_pocz,wart_pocz,y(1:497)];  
x4=[wart_pocz,u_test(1:499)];  
x5=[wart_pocz,wart_pocz,u_test(1:498)];  
  
%model szeregowo-rownolegly  
y_mod=sim(siec,[x1;x2;x3;x4;x5]);  
figure;  
plot([1:500],y,[1:500],y_mod);
```

Wynik testu przedstawia poniższy rysunek:



Rys.4 Testowanie modelu szeregowo-równoległego nieliniowego systemu dynamicznego

--- sygnał uczący  
--- odpowiedź sieci

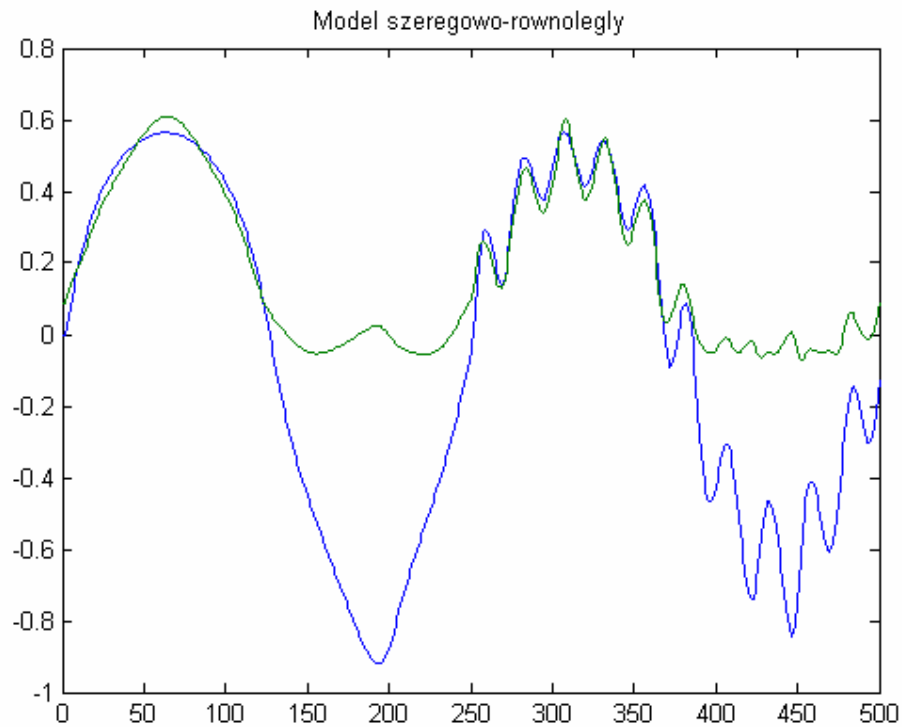
**Ad c)**

- zmiana liczby warstw sieci

Do badania wpływu zmiany liczby warstw sieci na proces uczenia sieci wykorzystaliśmy następujący rodzaj sieci:

```
siec=newff([-1 1;-1 1;-1 1;-1 1;-1 1],[5 6 5 1],{'tansig','tansig','tansig','tansig'},'traingdx')
```

Wynik testu przeprowadzonego dla modelu szeregowo-równoległego przedstawia poniższy rysunek:



Rys.5 Testowanie modelu szeregowo-równoległego nieliniowego systemu dynamicznego, dla sieci czterowarstwowej

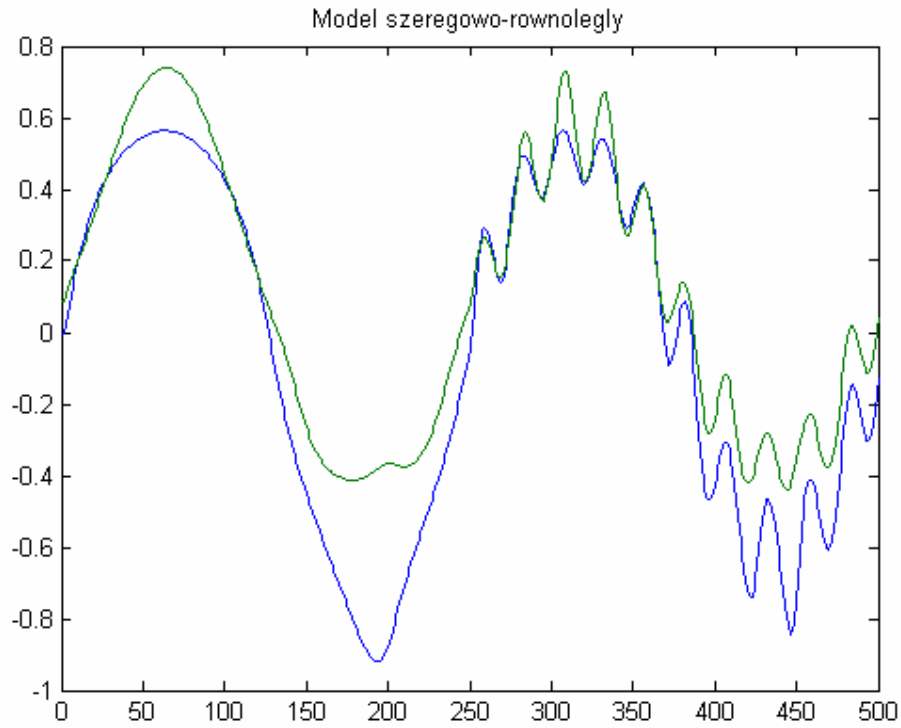
--- sygnał uczący  
--- odpowiedź sieci

- zmiana liczby neuronów w sieci

Do badania wpływu zmiany liczby neuronów w poszczególnych warstwach sieci na proces uczenia sieci wykorzystaliśmy następujący rodzaj sieci:

```
siec=newff([-1 1;-1 1;-1 1;-1 1],[5 10 15 1],{'tansig','tansig','tansig','tansig'},'traingdx');
```

Wynik testu przeprowadzonego dla modelu szeregowo-równoległego przedstawia poniższy rysunek:



Rys.6 Testowanie modelu szeregowo-równoległego nieliniowego systemu dynamicznego, dla sieci czterowarstwowej(zmieniona liczba neuronów)

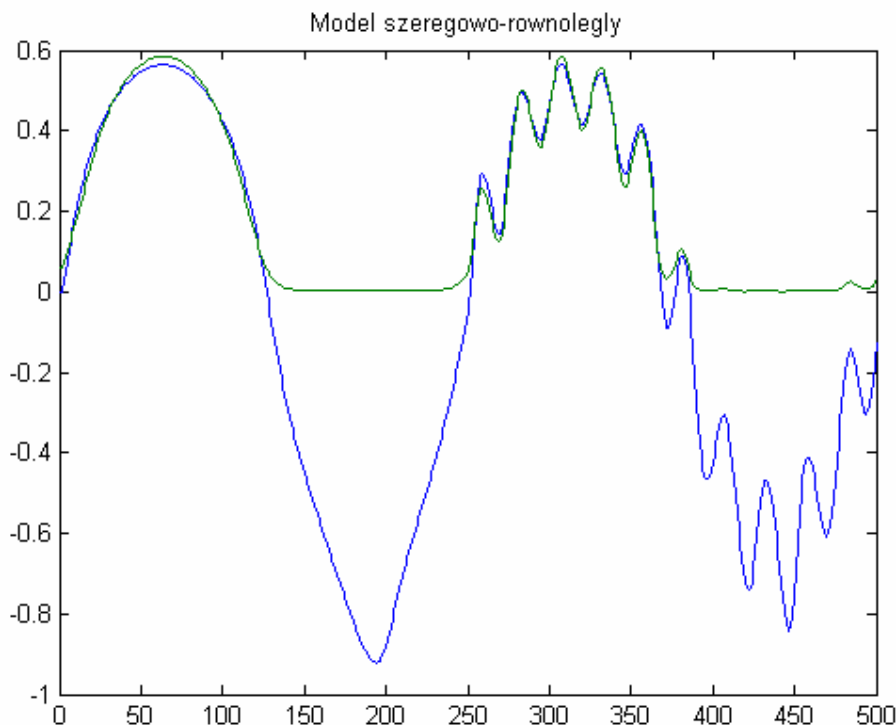
----- sygnał uczący  
----- odpowiedź sieci

- zmiana funkcji aktywacji neuronów

Do badania wpływu zmiany funkcji aktywacji dla poszczególnych warstw sieci na proces uczenia sieci wykorzystaliśmy następujący rodzaj sieci:

```
siec=newff([-1 1;-1 1;-1 1;-1 1;-1 1],[5 1],{'logsig','logsig'},'traingdx');
```

Wynik testu przeprowadzonego dla modelu szeregowo-równoległego przedstawia poniższy rysunek:



Rys.7 Testowanie modelu szeregowo-równoległego nieliniowego systemu dynamicznego, dla sieci dwuwarstwowej(zmienione funkcje aktywacji )

--- sygnał uczący  
 --- odpowiedź sieci

### **Wnioski:**

Celem ćwiczenia było poznanie struktur sieci neuronowych umożliwiających identyfikację systemów dynamicznych.

W podpunkcie *a* i *b* przeprowadziliśmy proces uczenia modelu szeregowo-równoległego oraz modelu równoległego nieliniowego systemu dynamicznego. Jak wynika z porównania rysunku 2 i 4 (przedstawiających wynik testowania sieci) model szeregowo-równoległy okazał się modelem dokładniejszym.

W kolejnym podpunkcie nadaliśmy zmianę architektury sieci na wpływ jej uczenia i dokładności. Okazało się, że zmiana liczby warstw sieci nie wpłynęła korzystnie na dokładność odpowiedzi (*Rysunek 5*), przy czym należy zauważyć, że zmiana ilości neuronów w poszczególnych warstwach znacznie poprawiła dokładność odpowiedzi badanej sieci (*Rysunek 6*). Jak wynika z *Rysunku 7*, bardzo ważnym czynnikiem wpływającym na poprawność działania sieci jest zastosowanie odpowiedniej funkcji aktywacji. Choć ilość warstw i liczba neuronów nie uległa zmianie w porównaniu do sieci wzorcowej, to zastosowanie funkcji sigmoidalnej (przyjmuje wartości 0...1) znacząco pogorszyło wyniki sieci w porównaniu z tangensoidalną funkcją aktywacji (która przyjmuje wartości -1...1). Z podpunktu *c* wynika jednoznacznie, że odpowiedni dobór architektury ma znaczący wpływ na poprawne działanie sieci. Sieć, która bardzo dobrze sprawdza się dla jednego problemu, w innym może okazać się bezużyteczna.