**Sztuczna inteligencja w automatyce – projekt II, zadanie 11**

Regulowany proces (symulowana rzeczywistość) opisany jest następującymi równaniami

$$\begin{aligned} x_1(k) &= -\alpha_1 x_1(k-1) + x_2(k-1) + \beta_1 g_1(u(k-4)) \\ x_2(k) &= -\alpha_2 x_1(k-1) + \beta_2 g_1(u(k-4)) \\ y(k) &= g_2(x_1(k)) \end{aligned}$$

gdzie $u$ – sygnał wejściowy, $y$ – sygnał wyjściowy, $x_1$, $x_2$ – zmienne stanu, $\alpha_1 = -1{,}413505$, $\alpha_2 = 0{,}462120$, $\beta_1 = 0{,}016447, \beta_2 = 0{,}012722$ oraz

$$g_1(u(k-4)) = \frac{(\exp(4{,}25u(k-4)) - 1)}{(\exp(4{,}25u(k-4)) + 1)},\ g_2(x_1(k)) = -0{,}7(1 - \exp(-2{,}5x_1(k)))$$

W nominalnym punkcie pracy $u = y = x_1 = x_2 = 0$, sygnał wejściowy może się zmieniać w granicach od $u^{\min} = -1$ do $u^{\max} = 1$.

**I. Symulacja procesu (0-2 pkt.)**

1. Wyznaczyć (metodą analityczną lub symulacyjną) i zamieścić w sprawozdaniu charakterystykę statyczną procesu (zależność $y(u)$) dla sygnału sterującego z zakresu $u^{\min} \ldots u^{\max}$.
2. Przeprowadzić symulację procesu (MATLAB) dla sekwencji zmian skokowych sygnału sterującego z zakresu $u^{\min} \ldots u^{\max}$. Należy wygenerować dwa zbiory danych (zbiór danych uczących i weryfikujących), w każdym zbiorze powinno być co najmniej 2000 probek. Eksperymentalnie dobrać okres zmian sygnału sterującego (np. co 50 kroków). Zamieścić rysunki danych.

**II. Modelowanie procesu (0-7 pkt.)**

1. Określić opóźnienie $\tau$ procesu.
2. Przeprowadzić uczenie serii modeli neuronowych procesu za pomocą programu `sieci` (program i opis dostępne są w pliku `www.ia.pw.edu.pl/~maciek/szau/sieci.zip`). Modele uczyć w trybie rekurencyjnym (predyktor OE), zastosować algorytm BFGS. Przyjąć dynamikę drugiego rzędu
$$\hat{y}(k) = f(u(k-\tau), u(k-\tau-1), \hat{y}(k-1), \hat{y}(k-2))$$
Rozważyć modele o liczbie neuronów ukrytych od 1 do 10. Dla każdej struktury modelu (tzn. liczby neuronów ukrytych) uczenie powtórzyć co najmniej 5 razy. Uzyskane wyniki (dla modeli o najmniejszym błędzie dla zbioru danych weryfikujących) przedstawić w formie tabeli – pokazać jak wpływa liczba neuronów ukrytych na błąd dla obu zbiorów danych.
3. Dokonać wyboru najlepszego modelu neuronowego (pod względem błędu i liczby parametrów), tzn. określić liczbę neuronów ukrytych. Dla wybranego modelu przedstawić na rysunku zmiany błędów predyktora ARX (bez rekurencji) i OE w kolejnych iteracjach uczących.
4. Wybrany model neuronowy zasymulować w trybie rekurencyjnym (predyktor OE) dla zbioru danych uczących i weryfikujących. Zamieścić przebiegi sygnału wyjściowego procesu i modelu oraz wykresy relacji tych sygnałów (oddzielnie dla zbioru uczącego i weryfikującego).
5. Przeprowadzić uczenie modelu neuronowego w trybie rekurencyjnym, zastosować algorytm najszybszego spadku. Przyjąć dynamikę drugiego rzędu oraz wybraną liczbę neuronów ukrytych, uczenie powtórzyć kilka razy. Wybrać model o najmniejszym błędzie dla zbioru danych weryfikujących. Dla wybranego modelu przedstawić na rysunku zmiany błędów predyktora ARX i OE w kolejnych iteracjach uczących.

6. Przeprowadzić uczenie modelu neuronowego w trybie bez rekurencji (predyktor ARX), zastosować algorytm BFGS. Przyjąć dynamikę drugiego rzędu

$$\hat{y}(k) = f(u(k-\tau), u(k-\tau-1), y(k-1), y(k-2))$$

   oraz wybraną liczbę neuronów ukrytych, uczenie powtórzyć kilka razy. Wybrać model o najmniejszym błędzie dla zbioru danych weryfikujących. Dla wybranego modelu przedstawić na rysunku zmiany błędów predyktora ARX i OE w kolejnych iteracjach uczących.
7. Wybrany model neuronowy uczony w trybie ARX zasymulować w trybie rekurencyjnym (predyktor OE) dla zbioru danych uczących i weryfikujących. Zamieścić przebiegi sygnału wyjściowego procesu i modelu oraz wykresy relacji tych sygnałów (oddzielnie dla zbioru uczącego i weryfikującego).
8. Metodą najmniejszych kwadratów wyznaczyć model liniowy o dynamice drugiego rzędu (jak model neuronowy). Zasymulować model w trybie rekurencyjnym (predyktor OE) dla zbioru danych uczących i weryfikujących. Zamieścić przebiegi sygnału wyjściowego procesu i modelu oraz wykresy relacji tych sygnałów (oddzielnie dla zbioru uczącego i weryfikującego).

**III. Modelowanie procesu za pomocą przyborników programu MATLAB (0-3 pkt.)**

1. Wybrać przybornik programu MATLAB nadający się do zadania modelowania procesu za pomocą sieci neuronowych (np. Deep Learning Toolbox lub dowolny inny przybornik spełniający wymagania). W sprawozdaniu zamieścić zwięzły, kilkuzdaniowy opis wybranego przybornika.
2. Przeprowadzić uczenie kilku modeli neuronowych za pomocą wybranego przybornika (jedynie w trybie ARX). Struktura modeli oraz liczba ich neuronów ukrytych powinna być taka, jak dla modelu o najmniejszym błędzie wyznaczonym w punkcie II projektu. Skorzystać z dwóch dowolnie wybranych dostępnych algorytmów uczenia modeli (np. algorytm Levenberga-Marquardta oraz algorytm gradientów sprzężonych)
3. Wybrany model zasymulować zarówno w trybie ARX, jak i OE dla zbioru danych uczących i weryfikujących. Zamieścić przebiegi sygnału wyjściowego procesu i modelu oraz wykresy relacji tych sygnałów (oddzielnie dla zbioru uczącego i weryfikującego).
4. Przeprowadzić porównanie jakości wybranego modelu wyznaczonego za pomocą przybornika programu MATLAB z wybranym najlepszym modelem wyznaczonym w zadaniu II projektu.

**IV. Regulacja procesu (0-8 pkt.)**

1. Zaimplementować algorytm regulacji predykcyjnej z Nieliniową Predykcją i Linearyzacją (NPL) bazujący na wybranym modelu neuronowym **wyznaczonym w zadaniu II projektu** (uczonym w trybie rekurencyjnym). Możliwa jest wersja analityczna algorytmu (z przycinaniem obliczonego sygnału sterującego do zakresu $u^{\min} \ldots u^{\max}$) lub wersja numeryczna (uwzględniająca ograniczenia wartości sygnału w zadaniu optymalizacji).
2. Przeprowadzić strojenie algorytmu NPL: dobrać wartości horyzontu sterowania i predykcji oraz możliwie małą wartość współczynnika kary $\lambda$. Na podstawie charakterystyki statycznej procesu zaproponować trajektorię zadaną w postaci kilku skoków w szerokim zakresie zmian sygnału wyjściowego. Zamieścić przebiegi sygnału wejściowego i wyjściowego procesu.
3. **W tym samym pliku co algorytm NPL** zaimplementować algorytm regulacji predykcyjnej typu GPC bazujący na wyznaczonym modelu liniowym (fragmenty programu muszą być wspólne). Wersja algorytmu GPC (analityczna lub numeryczna) musi być taka sama jak wersja algorytmu NPL, oba algorytmy muszą mieć te same parametry (horyzonty i współczynnik $\lambda$), przyjąć taką samą trajektorię zadaną. Zamieścić przebiegi sygnału wejściowego i wyjściowego procesu.

**Zadania dodatkowe (punktowane dodatkowo w skali 0-5 pkt.)**

4. Zaimplementować algorytm PID z ograniczniem sygnału sterującego do zakresu $u^{\min} \dots u^{\max}$. Przeprowadzić strojenie algorytmu metodą Zieglera-Nicholsa lub metodą prób i błędów (wskazówka: najwygodniej zadawać parametry regulatora ciągłego i na ich podstawie wyznaczać nastawy regulatora dyskretnego). Przyjąć taką samą trajektorię zadaną jak w algorytmach NPL i GPC. Zamieścić przebiegi sygnału wejściowego i wyjściowego procesu.
5. Zaimplementować algorytm regulacji predykcyjnej z Nieliniową Optymalizacją (NO) w wersji numerycznej (z ograniczeniami). Przyjąć te same parametry (horyzonty i współczynnik $\lambda$) oraz taką samą trajektorię zadaną jak w algorytmach NPL, GPC i PID. Zamieścić przebiegi sygnału wejściowego i wyjściowego procesu.

**Uwagi:**

a) Wszystkie algorytmy regulacji muszą być symulowane w jednym programie (wspólna inicjalizacja, symulacja procesu i niektóre fragmenty kodu).
b) Przesłać sprawozdanie w pliku pdf oraz **spakowane** wszystkie pliki źródłowe (MATLAB) do modułu Sprawozdania na serwerze Studia do dnia 16.1.2024, godz. 23.59. Nie przysyłać innych plików, np. graficznych, doc, tex itp.
c) Maksymalna liczba punktów wynosi 20 (+5 punktów dodatkowych). Za każdy dzień spóźnienia odejmowany jest 1 punkt.
d) Projekt będzie przyjmowany do dnia 25.1.2024, godz. 23.59.