Sztuczna inteligencja w automatyce – projekt II, zadanie 2

Symulator procesu oraz przykład użycia znajdują się w pliku w pliku `https://www.ia.pw.edu.pl/~maciek/szau/proces2.zip`. W początkowym punkcie pracy $u = y = x = 0$, sygnał wejściowy może się zmieniać w granicach od $u^{\min} = -1$ do $u^{\max} = 1$.

I. Symulacja procesu

1. Wyznaczyć metodą symulacyjną i zamieścić w sprawozdaniu charakterystykę statyczną procesu (zależność $y(u)$) dla sygnału sterującego z zakresu $u^{\min} \dots u^{\max}$.
2. Przeprowadzić symulację procesu (MATLAB) dla sekwencji losowych zmian skokowych sygnału sterującego z zakresu $u^{\min} \dots u^{\max}$. Należy wygenerować dwa zbiory danych (zbiór danych uczących i weryfikujących), w każdym zbiorze powinno być co najmniej 2000 probek. Dobrać okres zmian sygnału sterującego (np. co 30 kroków). Zamieścić rysunki danych.

II. Modelowanie neuronowe procesu przy użyciu sieci ELM

Wyznaczyć serię modeli neuronowych typu Extreme Learning Machines z jedną warstwą ukrytą zawierającą $K = 5, 10, 15, 20, \dots$ neuronów z funkcją aktywacji tangens hiperboliczny. Przyjąć rząd dynamiki modelu i opóźnienie takie same jak w symulatorze. Na przykład, dla symulatora `proces1_symulator(u(k-3),u(k-4),x(k-1),x(k-2))`, należy przyjąć wejścia sieci $u(k-3)$, $u(k-4)$, $y(k-1)$, $y(k-2)$.

1. Dla każdej sieci:
   a) uczenie powtórzyć co najmniej 5 razy,
   b) wyznaczyć błędy dla zbioru uczącego i weryfikującego w dwóch trybach: bez rekurencji (jeden krok do przodu) i rekurencyjnym.
2. Wybrać najlepszy model dla każdej liczby neuronów ukrytych, charakteryzujący się w trybie rekurencyjnym najmniejszym błędem dla zbioru danych weryfikujących. Wyniki dla najlepszych modeli przedstawić w formie rysunku lub tabeli, tzn. pokazać jak wpływa liczba neuronów ukrytych na błąd dla obu zbiorów danych.
3. Wybrać jeden model, kierując się kryterium małego błędu rekurencyjnego i możliwie małą złożonością.
4. Pokazać na rysunkach wyjście wybranego modelu dla obu zbiorów danych w trybie bez rekurencji i z rekurencją.
5. Pokazać na rysunkach problemy z działaniem rekurencyjnym (jeżeli występują), np. model ze zbyt małą liczbą parametrów lub przewymiarowany.
6. Opisać otrzymane wyniki. Zamieścić programy.

III. Modelowanie neuronowe procesu przy użyciu Neural Network Toolbox (wersja standardowa) <u>lub</u> Deep Learning Toolbox (wersja rozszerzona, dodatkowe 4 punkty)

1. W przypadku Neural Network Toolbox, należy wyznaczyć serię modeli neuronowych z jedną warstwą ukrytą zawierającą $K = 1, 2, 3, \dots$ neuronów z funkcją aktywacji tangens hiperboliczny. Zastosować algorytm uczenia Levenberga-Marquardta. Powtórzyć polecenia 1-6 z punktu II. Dodatkowo, dla wybranej struktury modelu sprawdzić efektywność uczenia algorytmem najszybszego spadku (omówić zbieżność oraz dokładność modelu w obu trybach). **Uwaga:** należy pamiętać o wyłączeniu automatycznego podziału danych oraz skalowania sygnałów wejścia i wyjścia sieci.
2. W przypadku Deep Learning Toolbox, należy wyznaczyć serię modeli neuronowych z jedną warstwą ukrytą zawierającą $K = 1, 2, 3, \dots$ neuronów z funkcją aktywacji tangens hiperboliczny oraz dodatkowymi sprzężeniami. Niech x_1, x_2, x_3 oraz x_4 będą wejściami sieci

(np. dla symulatora `proces1_symulator(u(k-3),u(k-4),x(k-1),x(k-2))`, $x_1 = u(k-3)$, $x_2 = u(k-4)$, $x_3 = y(k-1)$, $x_4 = y(k-2)$). Sygnał wyjściowy sieci ma postać

$$y(k) = \underbrace{w_1^{22}\sin(x_1(k)) + w_2^{22}x_2(k)}_{\text{dodatkowe składniki}} + w_0^2 + \sum_{i=1}^{K} w_i^2 v_i$$

gdzie $v_i(k)$ oznaczają sygnały wyjściowe kolejnych neuronów ukrytych, $i = 1, \ldots, K$. Wyjścia poszczególnych neuronów ukrytych oblicza się jako

$$v_i(x) = \varphi(z_i) = \text{tgh}(z_i)$$

Suma sygnałów wejściowych poszczególnych neuronów ukrytych ma postać

$$z_i(k) = \begin{cases} w_{i,0}^1 + \sum_{j=1}^{4} w_{i,j}^1 x_j(k) + \underbrace{w_1^{11}\cos(x_2(k))}_{\text{dodatkowy składnik}} & \text{gdy } i = 1 \\ w_{i,0}^1 + \sum_{j=1}^{4} w_{i,j}^1 x_j(k) & \text{w pozostałych przypadkach} \end{cases}$$

Zastosować algorytm uczenia LBFGS. Powtórzyć polecenia 1-6 z punktu II. Dodatkowo, dla wybranej struktury modelu sprawdzić efektywność uczenia algorytmem Adam (omówić zbieżność oraz dokładność modelu w obu trybach). **Uwaga:** do uczenia i weryfikacji stosować całe zbiory danych (trajektorie), nie dzielić na podzbiory (ang. mini batch).

IV. Modelowanie liniowe procesu

Metodą najmniejszych kwadratów wyznaczyć model liniowy o argumentach takich samych jak modele neuronowe. Wyznaczyć błędy dla zbioru uczącego i weryfikującego w trybie bez rekurencji i rekurencyjnym. Pokazać na rysunku wyjście modelu dla obu zbiorów danych w obu trybach.

V. Regulacja procesu

1. Zaimplementować algorytm regulacji predykcyjnej z Nieliniową Predykcją i Linearyzacją (NPL) bazujący na dwóch modelach neuronowych, wybranych w zadaniu II i III. Zastosować algorytm w wersji analitycznej (z przycinaniem obliczonego sygnału sterującego do zakresu $u^{\min} \ldots u^{\max}$).
2. Przeprowadzić strojenie algorytmu NPL: dobrać wartości horyzontu sterowania i predykcji oraz możliwie małą wartość współczynnika kary λ. 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 dla różnych ustawień parametrów oraz obu modeli.

Uwagi:

a) W sprawozdaniu opisać dokładnie co i jak zostało zrobione. Nie podawać teorii. Zamieścić programy oraz omówić wyniki.
b) Przesłać sprawozdanie w pliku pdf oraz **spakowane** wszystkie pliki źródłowe (MATLAB) do modułu Sprawozdania na serwerze Studia do dnia 20.1.2026, godz. 23.59. Nie przysyłać innych plików, np. graficznych, doc, tex itp.
c) Maksymalna liczba punktów wynosi 20 (+4 punkty dodatkowe). Za każdy dzień spóźnienia odejmowany jest 1 punkt.
d) Projekt będzie przyjmowany do dnia 29.1.2026, godz. 23.59.