Teoria Sterowania

Sterowanie Inteligentne: Ewolucyjna identyfikacja obiektu sterowania

Zasady zaliczenia pakietu instrukcji "Sterowanie inteligentne"

Podstawą do realizacji ćwiczeń jest wiedza z wykładu (na wykładach znajdziecie Państwo komplet wiedzy niezbędnej do zrozumienia zadań oraz zaplanowania sposobu ich wykonywania) oraz wiedza o sieciach neuronowych i algorytmach ewolucyjnych prezentowana na przedmiotach *Przetwarzanie Sygnałów i Identyfikacja w Sterowaniu Urządzeń Mechatronicznych* oraz *Przetwarzanie Sygnałów i Identyfikacja w Monitorowaniu Urządzeń Mechatronicznych.*

Komplet materiałów z w.w. przedmiotów oraz notatki wykładowe z wykładu o sterowaniu inteligentnym znaleźć można na stronie **http://galaxy.agh.edu.pl/~zdw/students.html**

W instrukcji znajdują się "Zadania na 3.0" - oznaczone kolorem czerwonym, "Zadania na 4.0" oznaczone kolorem pomarańczowym oraz "Zadania na 5.0" oznaczone kolorem zielonym. Wykonanie zadań na daną ocenę oznacza uzyskanie oceny warunkowej (np. zrobienie wszystkich zadań na 3.0 oraz na 4.0 skutkuje uzyskaniem warunkowego "4.0" - o ile na kolejnych zajęciach student zda i obroni komplet podstawowych zadań z instrukcji do 5.0 włącznie).

W przypadku jednokrotnej nieobecności lub jednokrotnego braku uzyskania oceny 3.0 na laboratorium student powinien rozwiązać komplet zadań podstawowych z problematycznej instrukcji oraz jedno wskazane przez prowadzącego "zadanie rozszerzające" (w kolorze niebieskim) - i zdać rezultaty w formie raportu.

Przypadki nieobjęte powyższymi zasadami (nieobecność na wielu zajęciach, Brak uzyskania oceny 3.0 na wielu zajęciach) traktowane będą indywidualnie i wymagać będą kontaktu z prowadzącym w celu ustalenia wymaganego zakresu wiedzy i formy zaliczenia laboratorium.

Pierwsza instrukcja rozpoczynać się będzie od krótkiego kolokwium wejściowego obejmującego materiał z wykładu.

Definicje "indywidualnych zadań"

Przed rozpoczęciem zajęć należy podzielić przez 5 sumę liter swojego imienia i nazwiska. Reszta z tego dzielenia stanowi numer funkcji tworzącej "indywidualne zadanie". Funkcje zawierające indywidualne zadania znajdują się w folderze *Plants*

Ewolucyjna identyfikacja obiektu sterowania

Dany jest pewien obiekt o nieznanej transmitancji. Należy odkryć tą transmitancję poprzez zarejestrowanie odpowiedzi układu na wymuszenie impulsowe a następnie zbudowanie nowego układu o takiej samej odpowiedzi. Pracę rozpoczniemy od wczytania obiektu. W celu poprawnego wykonania ćwiczenia proszę o powstrzymanie się od zaglądania do wnętrza funkcji, oglądania równania identyfikowanego obiektu, wykreślania jego charakterystyk Nyquista czy Bodego, obserwowania położeń biegunów itp. Na to przyjdzie czas w etapie weryfikacyjnym – gdy będziemy chcieli sprawdzić, czy nasz system identyfikacyjny poradził sobie z wykonaniem zadania. Nie zmieniaj również zaznaczonego na żółto argumentu funkcji. Jego zmiana będzie wymagana w drugiej części instrukcji, w problemie adaptacyjnym. Zmienić natomiast należy część niebieską – na numer swojego zadania indywidualnego.

```
clear all
close all
clc
addpath Plants\

% First: Lets test our unknown system using a step function and see what it does:
 T = [0:0.1:100];
 goalResult = ObjectForIdentification_4P_Test(T,1);
 figure;
 plot(ReferenceResponse)
```

Zaobserwowaliśmy reakcję obiektu na wymuszenie skokiem jednostkowym. Ten sygnał odpowiedzi będzie naszą referencją i zarazem jedyną informacją o układzie. Identyfikację rozpoczniemy od bardzo prostego kroku: Wygenerowania kilku losowych transmitancji a następnie porównania ich z naszą referencją. Wykonajmy poniższy kod i zaobserwujmy rezultat:

```
RandomObject = tf(1,[randn(),randn(),randn(), randn()]);
figure;
step(RandomObject,T);
```

Czy udało się zbudować układ stabilny?

Zadanie 1: Porównaj kilkakrotnie sygnał odpowiedzi obiektu zbudowanego losowo z referencją uzyskaną z układu wczytanego z **indywidualnego zadania**. W ilu przypadkach udało się uzyskać sygnał podobny do referencyjnego? W ilu uzyskany układ nie był stabilny? Od czego zależy prawdopodobieństwo wygenerowania układu stabilnego? Czy będzie się zmieniało wraz z rzędem generowanego losowo układu?

Warto zauważyć, że do wstępnego oszacowania, czy symulowany obiekt jest stabilny możemy sprawdzić wartość ostatniego punktu zarejestrowanego sygnału – dla wszystkich wygenerowanych obiektów, a następnie zliczyć obiekty nie spełniające warunku:

```
StartingPoint = 10*randn(4,1000);

 for k = 1:1000
     L = 1;
     M = StartingPoint(:,k);
     testSys = tf(L,M');
     testResult(:,k) = step(testSys,T);
 end
```



```
StableOnes = find(abs(testResult(end,:))<10);
StableNumber = length(StableOnes)
```

Porównajmy teraz pozostałe sygnały i zastanówmy się, czy udało nam się trafić wystarczająco blisko. Porównajmy na przykład pierwszy ze "stabilnych obiektów losowych" z naszym obiektem referencyjnym:

```
figure;
plot(goalResult,'k'); hold on
plot(testResult(:,StableOnes(1)),'r');
```

Porównywanie sygnałów "na oko" nie jest oczywiście rozsądnym podejściem do dokładnej identyfikacji obiektu. Aby podejść do tematu metodycznie i być w stanie porównać ze sobą wiele podobnych obiektów, potrzebować będziemy miary jakości. Na tym etapie zastosować możemy prosty wskaźnik RMS różnicy sygnałów referencyjnego i wygenerowanego. Korzystając z tej miary, uszeregujmy wygenerowane obiekty poczynając od najlepszego (najbardziej podobnego do referencji):

```
Residua = testResult-goalResult;
[Values,Indices] = sort(rms(Residua));

figure;
plot(goalResult,'k'); hold on
plot(testResult(:,StableOnes(1)),'r');
plot(testResult(:,Indices(1)),'b');
```

Zauważmy, że czynności, które właśnie wykonaliśmy, to po prostu ocena przystosowania wewnątrz pewnej populacji.

Dodajmy więc do tego kodu definicję populacji początkowej, prostą selekcję rankingową (n-najlepszych), oraz mutację w niewielkim stopniu zmieniającą parametry poszczególnych obiektów. Teraz wystarczy jedynie zamknąć odpowiedni fragment naszego kodu wewnątrz pętli odpowiedzialnej za kolejne generacje – i przygotowaliśmy podstawowy algorytm genetyczny do działania:


```
PopulationSize = 100;
MaxSteps = 100;
InitialStep = 1;
n = 30;

StartingPoint = 10*randn(4,PopulationSize);
Population = StartingPoint;

for iter = 1:MaxSteps
    clc; iter
    S(iter) = InitialStep;
    % Lets build phenotypes:
        for k = 1:PopulationSize
            L = 1;
            M = Population(:,k);
            testSys = tf(L,M');
            testResult(:,k) = step(testSys,T);
        end
        Residua = testResult-goalResult;
        % Lest sort the solutions and build a ranking:
        [Values,Indices] = sort(rms(Residua));
        BestResult(iter) = Values(1);
        % And now lets build a new population
        NewPopulation(:,1) = Population(:,Indices(1));
           for pop = 2:PopulationSize
               NewPopulation(:,pop) = Population(:,randi(n));
               NewPopulation(:,pop) = NewPopulation(:,pop) + ...
               S(iter)*randn(size(NewPopulation(:,pop)));
           end
        Population = NewPopulation;
        clear Values Indices StableOnes Residua
end
% Lets see the convergence curve:
       figure;   plot(BestResult);
% And lets see the final result:
       finalSys = tf(1,Population(:,1)');
       finalResult = step(finalSys,T);
       figure;
       plot(goalResult,'k'); hold on
       plot(finalResult,'r');
```

Zadanie 2: Przygotuj algorytm genetyczny o stałym kroku mutacji. Przetestuj statystycznie jego działanie ze względu na wartość zastosowanego kroku mutacji.

Zadanie 3: Po zakończeniu optymalizacji przyszedł czas na jakościowe porównanie obu układów (referencyjnego oraz zidentyfikowanego). Porównaj ze sobą uzyskane transmitancje oraz mapy zer i biegunów i oceń poprawność działania systemu.

Podpowiedź: układ identyfikowany możesz otrzymać używając tego kodu:

```
[~, goalSys] = ObjectForIdentification_4P_Test(T,1);
```

Zadanie 4: Wyposaż swój AG w zmienny krok mutacji i skonfiguruj go tak, aby uzyskać powtarzalne wyniki. Zastosuj 10 000 sprawdzeń funkcji celu (tj. np. 100 iteracji x 100 osobników)

Zadanie 5: Czy poziom trudności zadania zależy od stopnia nadmiarowości genotypu w stosunku do rzeczywistej złożoności układu? Przetestuj ponownie działanie AG ale tym razem zastosuj 5 współczynników równania charakterystycznego w genotypie. Porównaj odpowiedzi układu, transmitancje oraz mapy zer i biegunów uzyskane za pomocą tak skonfigurowanego algorytmu. Czy wynik staje się lepszy po zastosowaniu dużej nadmiarowości konfiguracji (np. 10x większa populacja)?

Identyfikacja obiektu w zmiennym środowisku

Do tej pory pracowaliśmy na obiektach statycznych. Teraz spróbujemy podążać z naszą optymalizacją za obiektem zmiennym w czasie. Na nasz algorytm genetyczny nałożymy nowe ograniczenie: mamy do dyspozycji co najwyżej 50 "sprawdzeń funkcji celu", po których obiekt ulega niewielkiej zmianie. Zauważmy, że oznacza to w praktyce konieczność "zmieszczenia się" w tych 50 testach z całą populacją. Ograniczmy więc wielkość populacji do 50 i modyfikujmy obiekt po każdej generacji. Modyfikacja obiektu będzie polegała na zmianie wartości iteratora przy obiekcie, czyli zmodyfikowaniu kodu w następujący sposób:

```
goalResult = ObjectForIdentification_4P_Test(T,iter);
```

Zadanie 6: Wykonaj identyfikację adaptacyjną obiektu zdefiniowanego w ramach swojego ***indywidualnego zadania.*** Zmodyfikuj AG tak, by w jak największym stopniu podążał za obiektem. Nowym kryterium oceny AG stanie się średnia wartość przystosowania najlepszego osobnika przez wszystkie generacje algorytmu, czyli:

```
mean(BestResult)
```

Przetestuj algorytmy testowane w ramach zadania 2 (kilka metod o stałym kroku, jedna zoptymalizowana metoda o zmiennym kroku) ze względu na tak zdefiniowane kryterium. Wykonaj modyfikację algorytmu polegającą na tym, że krok mutacji nie jest stały w obrębie generacji ale dobierany losowo w mutacji danego osobnika jako wartość "duża" (pozwalająca na szybką konwergencję) lub "mała" – pozwalająca na skuteczną eksploatację.

*** Zadanie 7**: Oceń zasadność zastosowanego podejścia z praktycznego punktu widzenia. Jaki jest koszt obliczeniowy zastosowanego rozwiązania w porównaniu do rozwiązań znanych z klasycznej Teorii Sterowania? Wykonaj identyfikację układu za pomocą metod klasycznych. W jakich sytuacjach stosowanie rozwiązań inteligentnych w sytuacjach praktycznych jest uzasadnione?

*** Zadanie 8**: W pierwszej części instrukcji zastosowaliśmy bardzo uproszczony sposób generowania populacji oraz późniejszego jej optymalizowania. Zaproponuj modyfikacje do Algorytmu Genetycznego uwzględniające następujące kwestie:

- Generowanie populacji początkowej w sposób gwarantujący stabilność oraz operatory mutacji napisane tak, by uniemożliwić destabilizację układu.
- Uwzględnienie faktu, że sam fakt położenia biegunów wpływa na ich znaczenie w finalnej odpowiedzi – oraz optymalizowanie położenia biegunów i zer indywidualnie bądź parami

*** Zadanie 9:** Aktualnie wykorzystywaliśmy metrykę bazującą wartości RMS residuum pomiędzy sygnałem referencyjnym i sygnałem obiektu testowego w trakcie identyfikacji. Przetestuj inne metryki jakości, uwzględniając:

- Metrykę bazującą na informacjach pozyskiwanej z funkcji step (czas wzrostu, czas ustalania, przeregulowanie itp.)
- Metrykę bazującą na średnim błędzie pomiędzy sygnałem referencyjnym i sygnałem obiektu testowego

*** Zadanie 10:** Wykonaj zadanie adaptacyjnej identyfikacji obiektu z zastosowaniem algorytmu memetycznego: Każdy osobnik w AG przed oceną przystosowania powinien zostać poddany lokalnej optymalizacji poprzez wykonanie kilku kroków algorytmu gradientowego. Krok algorytmu gradientowego powinien być taki sam, jak aktualnie wykorzystywany w AG krok mutacji. Spróbuj skonfigurować tak zaprojektowane rozwiązanie w celu maksymalizacji powtarzalności i skuteczności

Podpowiedź:
Zastanów się nad tym jak wpływa na różnorodność populacji zastosowanie dodatkowego kroku optymalizacji lokalnej. Zaobserwuj różnorodność populacji i zadbaj o jej podtrzymanie.

*** Zadanie 11:** Wykonaj zadanie adaptacyjnej identyfikacji obiektu z zastosowaniem algorytmu genetycznego, w którym krok mutacji jest uwzględniony jako kolejny gen: krok nie powinien więc być predefiniowany lub zmieniany w zależności od numeru generacji, ale generowany losowo, przekazywany potomstwu i mutowany. Czy takie rozwiązanie pozwala na skuteczniejszą adaptację do zmieniającego się obiektu?

*** Zadanie 12:** Wykonaj zadanie adaptacyjnej identyfikacji obiektu z zastosowaniem algorytmu genetycznego, w którym rozmiar wektora parametrów (liczba współczynników transmitancji) jest uwzględniona jako kolejny gen. AG powinien inicjalizować populację osobników o różnej wielkkości (od 2 do 8 współczynników) a następnie uwzględnić jako jeden z operatorów mutacji możliwość zmniejszenia lub zwiększenia ilości współczynników. W przypadku zmniejszenia losowy współczynnik powinien być usuwany, w przypadku zwiększenia losowy współczynnik powinien być dodawany. Czy takie rozwiązanie pozwala na automatyczne odnalezienie poprawnego rzędu transmitancji? Co zrobić, aby pozwoliło?

