SZAU Projekt III

Gajewski Paweł
Kaczmarek Kacper

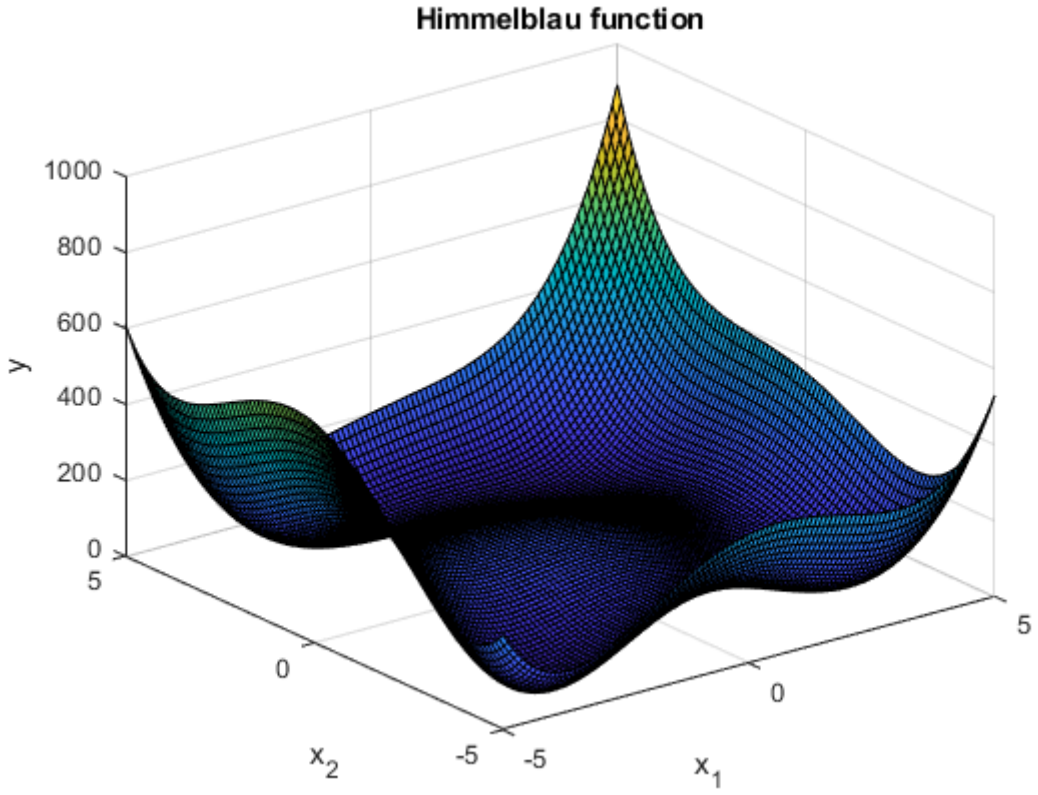
Styczeń 2020

1 Algorytm PSO - Particle Swarm Optimisation (algorytm roju)

1.1 Funkcja Himmelblau

Funkcja jest opisana wzorem:

$$y(x_1, x_2) = (x_1^2 + x_2 - 11)^2 + (x_1 + x_2^2 - 7)^2 \tag{1.1}$$



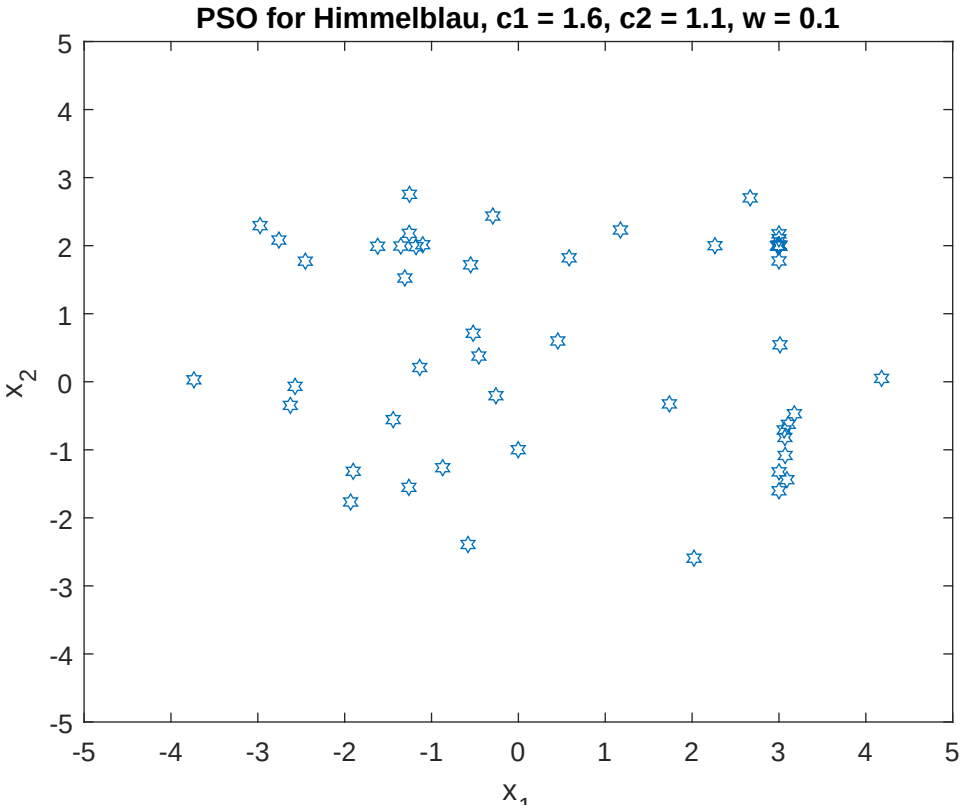
Minimum tej funkcji znajduje się w punkcie $[x_1, x_2] = [3, 2]$.

1.2 Wpływ wartości parametrów na rozwiązanie

W celu doboru najlepszych parametrów zostały przeprowadzone eksperymenty mające na celu wyłonienie najlepszych parametrów. Przeprowadzone eksperymenty prowadzone były po 5 razy dla każdej kombinacji paramterów($c_1 = [0.1, 0.6, 1.1, 1.6]$, $c_2 = [0.1, 0.6, 1.1, 1.6]$, $w = [0.1, 0.4, 0.9]$), a wyniki przedstawione w tabelce. Przedstawiono wyniki jedynie najlepszego wyboru, aby zwiększyć czytelność sprawozdania.

$C_1 = 1.6, C_2 = 1.1, W = 0.1$		
Próba	x_1	x_2
1	3.000000	2.000000
2	3.000000	2.000000
3	3.000000	2.000000
4	3.000000	2.000000
5	3.584428	-1.848127

Paramtery te jako jedyne cztery razy znalazły dobre rozwiązanie oraz znalazły je z odpowiednio dużą dokładnością (do 6 miejsc po przecinku). Algorytm ten dla następujących nastaw znajdujący rozwiązanie zakończył działanie w przedstawiony poniżej sposób



1.3 Proces

Równania procesu przedstawiają się następująco:

$$x_1(k) = -\alpha_1 x_1(k-1) + x_2(k-1) + \beta_1 g_1(u(k-5)) \tag{1.2}$$

$$x_2(k) = -\alpha_2 x_1(k-1) + \beta_2 g_1(u(k-5)) \tag{1.3}$$

$$y(k) = g_2(x_1(k)) \tag{1.4}$$

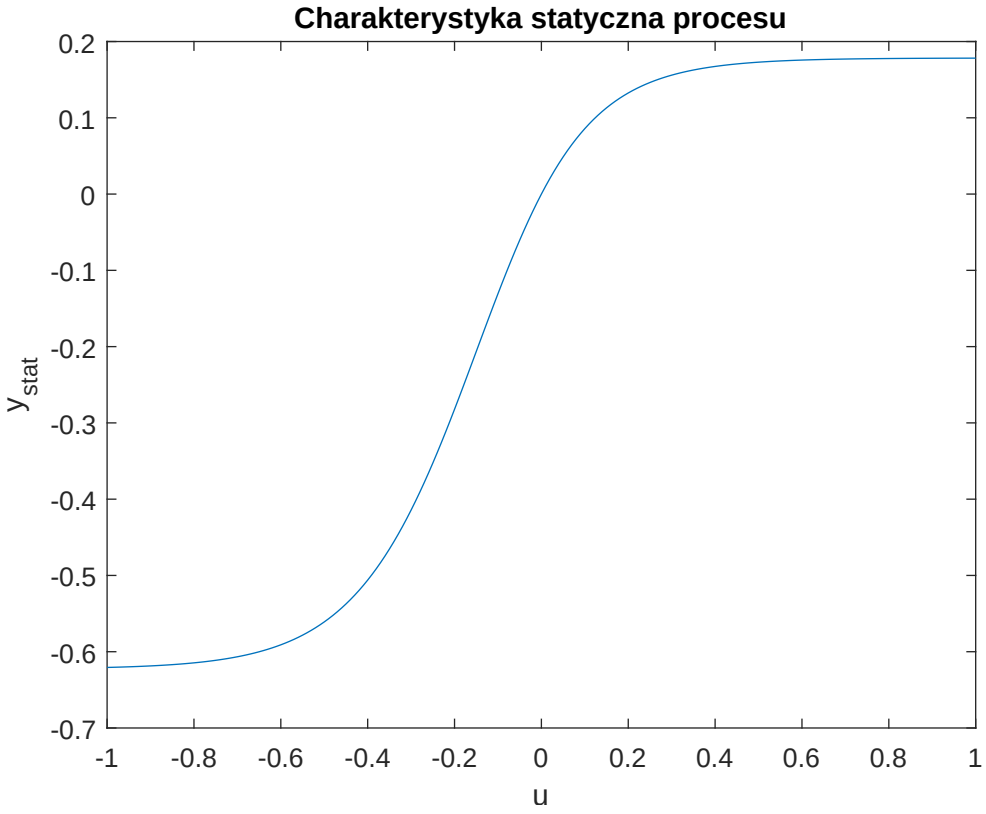
gdzie parametery maja wartości α_1=-1,489028, α_2=0.535261, β_1 = 0.012757, $\beta_2$0=0.01036 i

$$g_1(u(k-5)) = \frac{(e^{7u(k-5)} - 1)}{(e^{7u(k-5)} + 1)} \tag{1.5}$$

$$g_2(x_1(k)) = 0.25(1 - e^{-2.5x_1(k)}) \tag{1.6}$$

Charakterystykę statyczną można wyznaczyć analitycznie. Wtedy otrzymane równanie to:

$$y(u) = g_2(\frac{(\beta_2 + \beta_1)g_1(u)}{1 + \alpha_1 + \alpha_2}) \tag{1.7}$$



2 Regulator PID

2.1 Opis działania

Algorytm regulacji PID został zaimplementowany w swojej klasycznej wersji numerycznej, jednak aby go dostroić używane były nastawy dla PID'a ciągłego, ponieważ mają one przełożenie na rzeczywistość i łatwiej dzięki nim dostroić PID. Nastawy te to:

- K - wzmocnienie
- T_i - stała całkowania
- T_d - stała różniczkowania

Mają powyższe wartości możemy wyliczyć współczynniki dla werji dyskretnej (gdzie T jest stałą próbkowania sygnału)

$$r_2 = \frac{K * Td}{T} \tag{2.1}$$

$$r_1 = K * (\frac{T}{2 * T_i} - \frac{2 * T_d}{T} - 1) \tag{2.2}$$

$$r_0 = K * (\frac{T}{2 * T_i} + \frac{T_d}{T} + 1) \tag{2.3}$$

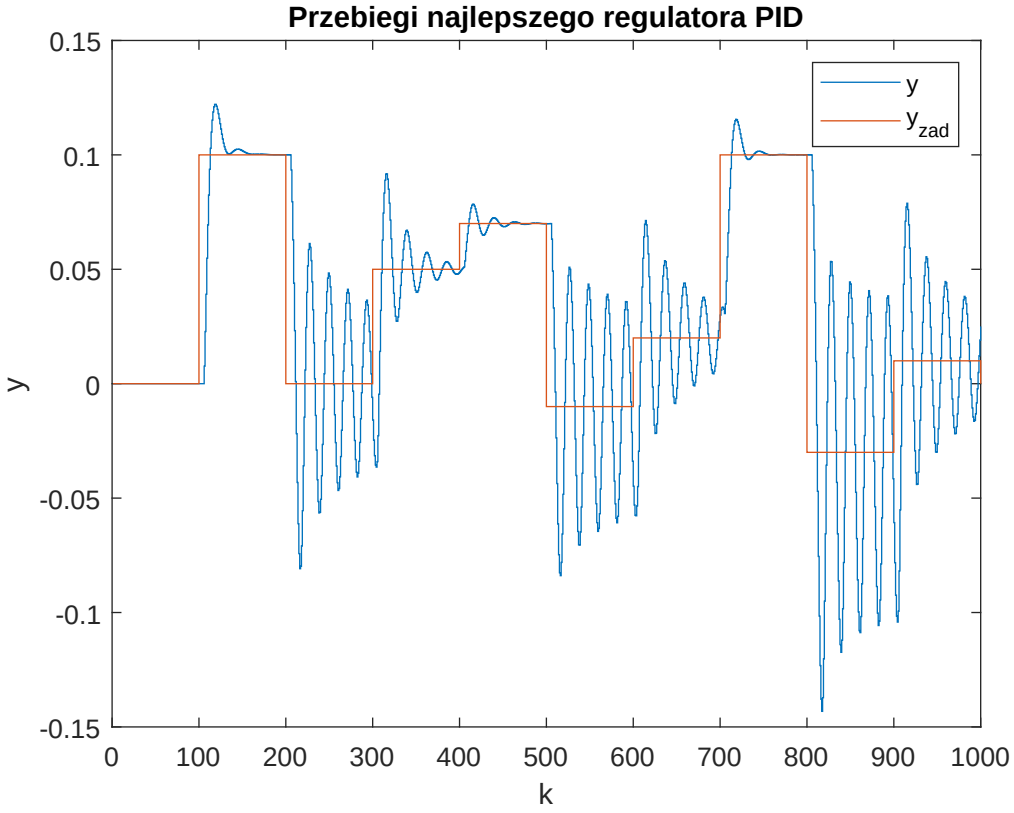
Prawo regulacji wygląda wtedy następująco:

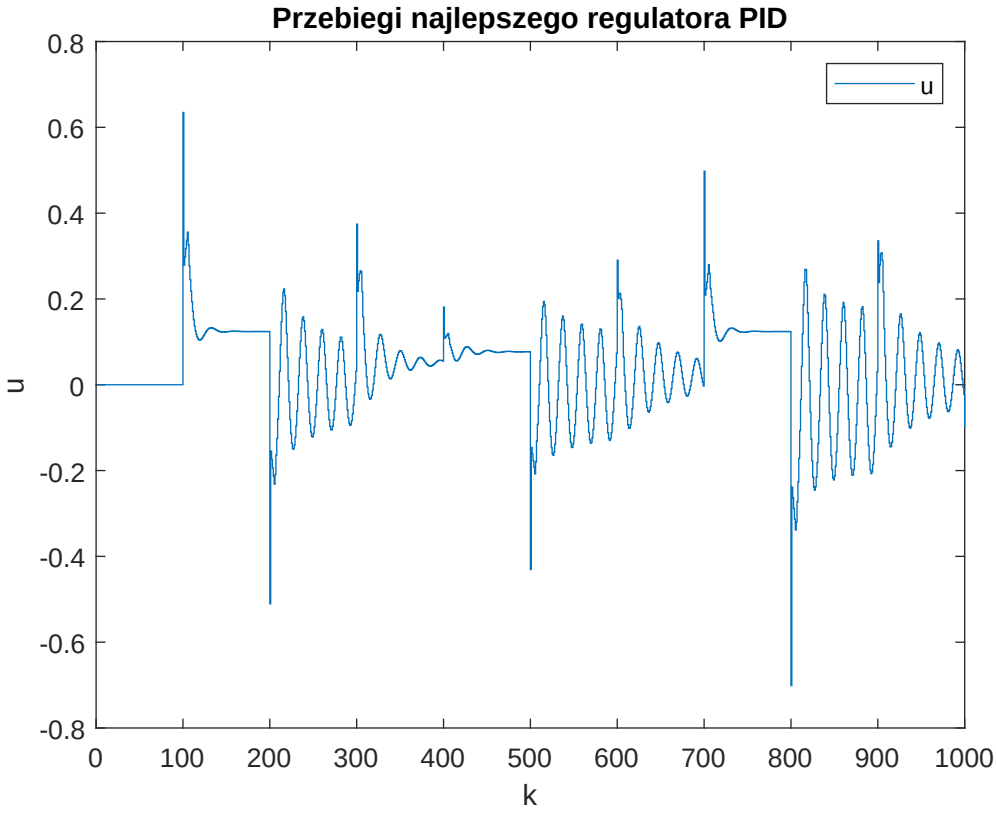
$$u(k) = r_2 * e(k-2) + r_1 * e(k-2) + r_0 * e(k) + u(k-1) \tag{2.4}$$

W ten sposób co iterację wyliczane są kolejen uchyby(oznaczone jako e) i wyznaczane nowe wartości sterowania ze wzoru powyżej.

2.2 Metoda inżynierska

Aby dostroić regulator odpowiednio, próbowano zastosować metodę inżynierską. Pierwszym etapem jest znaleznienie regulatra P, który powoduje drgania niegasnące procesu podczas skoku jednostkowego(tutaj skok o 0.1). Następnie próba dobrania dla opisanego wzmocnienia podzielonego przez 2 reszty parametrów. Eksperyment dał następujące rezultaty: K = 2.5, $T_i = 1.3$, $T_d = 0.15$





2.3 Szukanie nastaw algorytmu za pomocą PSO

Aby znaleźć optymalne nastawy dla algorytmu PID zastosowana została następująca funkcja celu:

$$E = \sum_{k=k_{pocz}}^{k_{konc}} (y^{zad}(k) - y(k))^2 \tag{2.5}$$

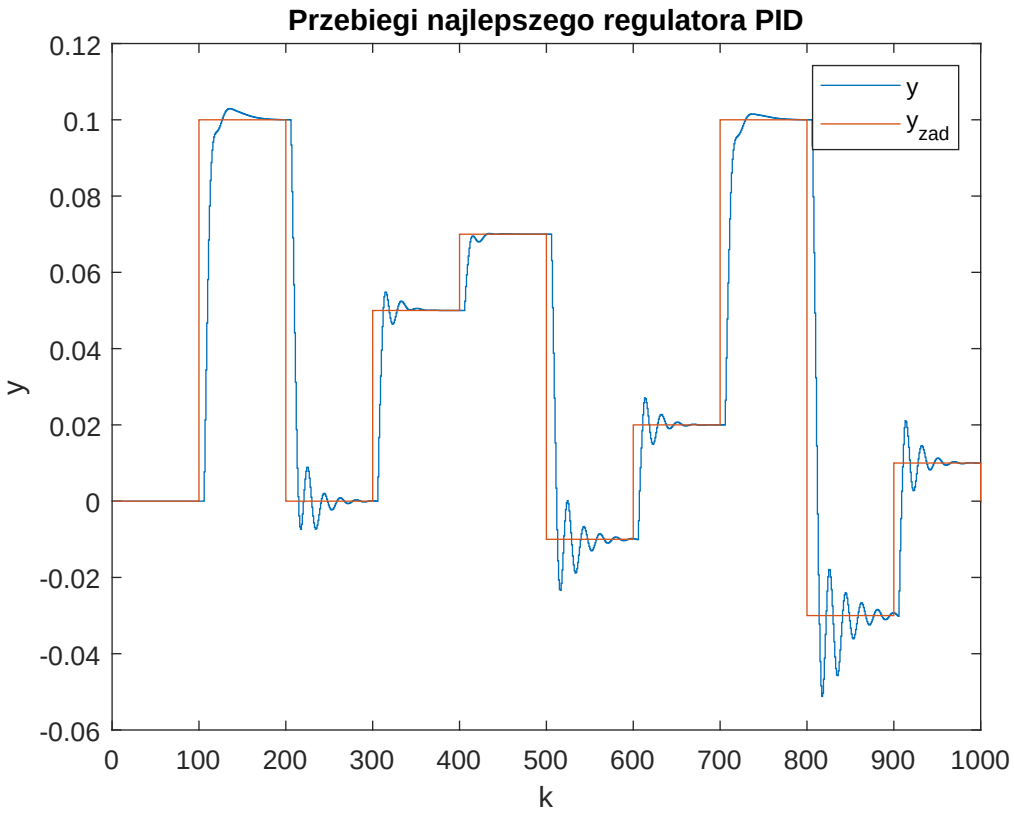
Algorytm stałał się znaleźć rozwiązanie kierując się minimalizacją tej sumy kwadratów błędów. Także w tym przypadku przeprowadzony został szereg eksperymentów mający na celu wyłonienie najlepszych paramterów próbujących rozwiązać ten problem. Najlepsze wyniki zostały przedstawione w tabeli poniżej.

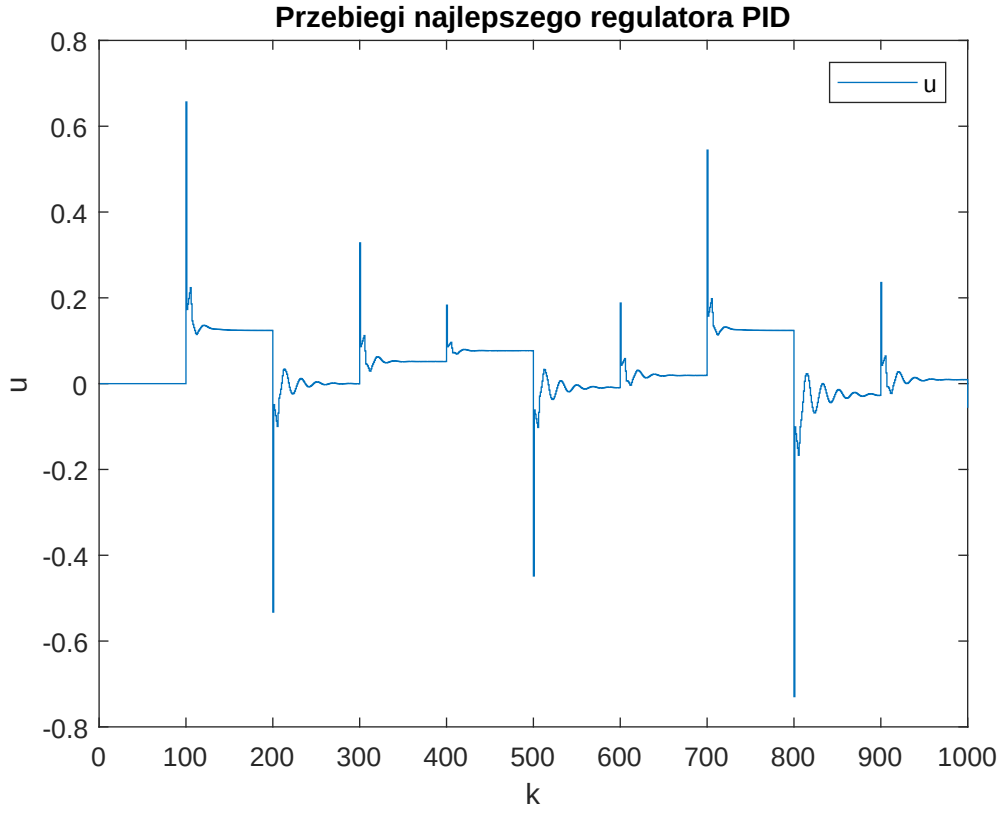
$C_1 = 1.6, C_2 = 1.1, W = 0.1$	
Próba	E
1	0.478424
2	0.478424
3	0.479112
4	0.478424
5	0.478424

Nastawy, które zostały obliczone przez algorytm są to:

1. $K = 1.538641$
2. $T_i = 1.210391$
3. $T_d = 0.322527$

A działanie regulatora przedstawia się następująco:


Przebiegi najlepszego regulatora PID
y
k
y
y_zad
0.12
0.1
0.08
0.06
0.04
0.02
0
-0.02
-0.04
-0.06
0
100
200
300
400
500
600
700
800
900
1000


Przebiegi najlepszego regulatora PID
u
k
u
0.8
0.6
0.4
0.2
0
-0.2
-0.4
-0.6
-0.8
0
100
200
300
400
500
600
700
800
900
1000

2.4 Wnioski

Regulator nie działa zadowalająco. W niektórych momentach wpada w gasnące oscylacje, głównie przy ujemnych wartościach wyjścia. Prawdopodobnie dzieje sie tak, ponieważ równiania opisujące proces nie są liniowe i mimo, że dla niektórych skoków PID działa dobrze, to dla niektórych niestety nie będzie działał zadowalająco.

3 Regulator NPL

Algorytm NPL jest odpowiedzią na wadę algorytmu NO. Algorytm NO jest złożony obliczeniowo, natomiast zastosowanie lineryzcaji pozwala zmniejszyć ilość obliczeń. Linearyzacji poddajemy tutaj model, który zostanie wykorzystany do predykcji, co sprawia, że zadanie optymalizaci jest kwadratowe. Wzór na liniową aproksymację wygląda następująco:

$$y(k) = \sum_{l=1}^{n_B} b_l(k)u(k-l) - \sum_{l=1}^{n_A} a_l(k)y(k-l) \tag{3.1}$$

gdzie $a_l(k)$ oraz $b_l(k)$ są zależnymi od punktu pracy współczynnikami modelu zlinearyzowanego. Linearyzacja potrzebna jest do stworzenia macierzy dynamicznej M, a następnie macierzy K, która wykorzystywana jest do znalezienia optymalnego przyrostu sterowania. Macierz m to kolejne współczynniki odpowiedzi skokowej liczone w podany sposób.

$$s_j(k) = \sum_{i=1}^{\min(j,n_B)} b_i(k) - \sum_{i=1}^{min(j-1,n_A)} a_i(k)s_{j-i}(k) \tag{3.2}$$

$$M(k) = \begin{bmatrix} s_1(k) & 0 & \dots & 0 \\ s_2(k) & s_1(k) & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ s_N(k) & s_{N-1}(k) & \dots & S_{N-N_u+1}(k) \end{bmatrix} \tag{3.3}$$

$$K = (M^T * M + \Lambda) * M^{-1} \tag{3.4}$$

Trajektoria swobodna jest liczona bez linaryzacji, z wykorzystaniem sieci neuronowej bez zmian. Następnie sterowanie wyliczane jest ze wzoru:

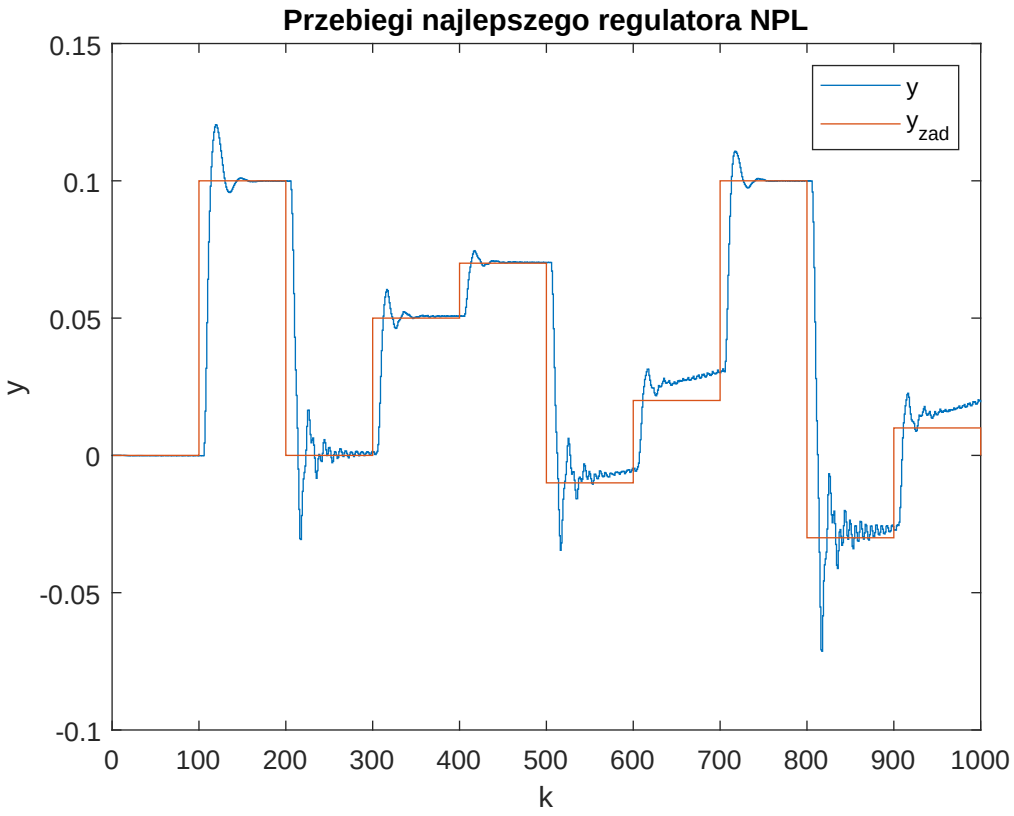
$$\Delta u = K * (y^{zad}(k) - y_0) \tag{3.5}$$

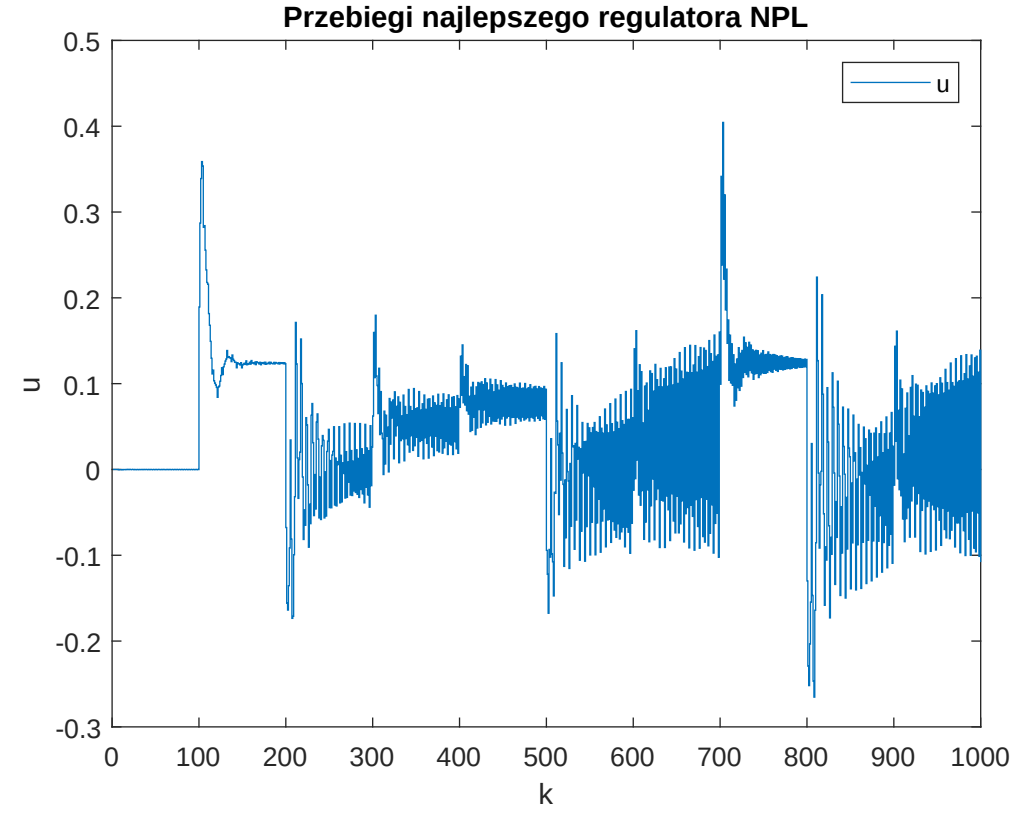
Wartość wyliczana powyżej jest następnie dodawana do sterowania z chwili poprzedniej. W tym algorytmie model jest linearyzowany w każdej iteracji i macierz K jest przeliczana od nowa. Parametrami sterowalnymi w regulatorze NPL są horyzonty: predykcji i sterowania oraz wartość współczynnika kary λ.
Nastawy regulatora NPL dobierane były za pomocą algorytmu PSO stając się zminimalizować funckję opisaną przy algorytmie PID. Wyniki przedstawiają się następująco:

$C_1 = 1, C_2 = 0.7, W = 0.6$	
Próba	E
1	2.89
2	0.492783
3	0.492783
4	0.492783
5	0.492783

Wybrane nastawy są następujące:

1. $N = 10$
2. $N_u = 6$
3. $\lambda = 0.1524098$


Przebiegi najlepszego regulatora NPL
y
y_zad
0.15
0.1
0.05
0
-0.05
-0.1
0
100
200
300
400
500
600
700
800
900
1000
k


Przebiegi najlepszego regulatora NPL
u
0.5
0.4
0.3
0.2
0.1
0
-0.1
-0.2
-0.3
0
100
200
300
400
500
600
700
800
900
1000
k

3.1 Wnioski

Niestety przebiegi nie są zadowalające pomimo wyboru najleszych nastaw przez algorytm PSO. Lambda jest za mała i sprawia że sterowania jest bardzo szybko zmienne w całkiem sporych zakresach, co prawdopodobnie nie byłoby możliwe do zrealizowania na prawdziwym obiekcie. Regulator ten zachowuje się jednak lepiej niżeli regulator PID.

4 Modelowanie neuronowe procesu

4.1 Model neuronowy

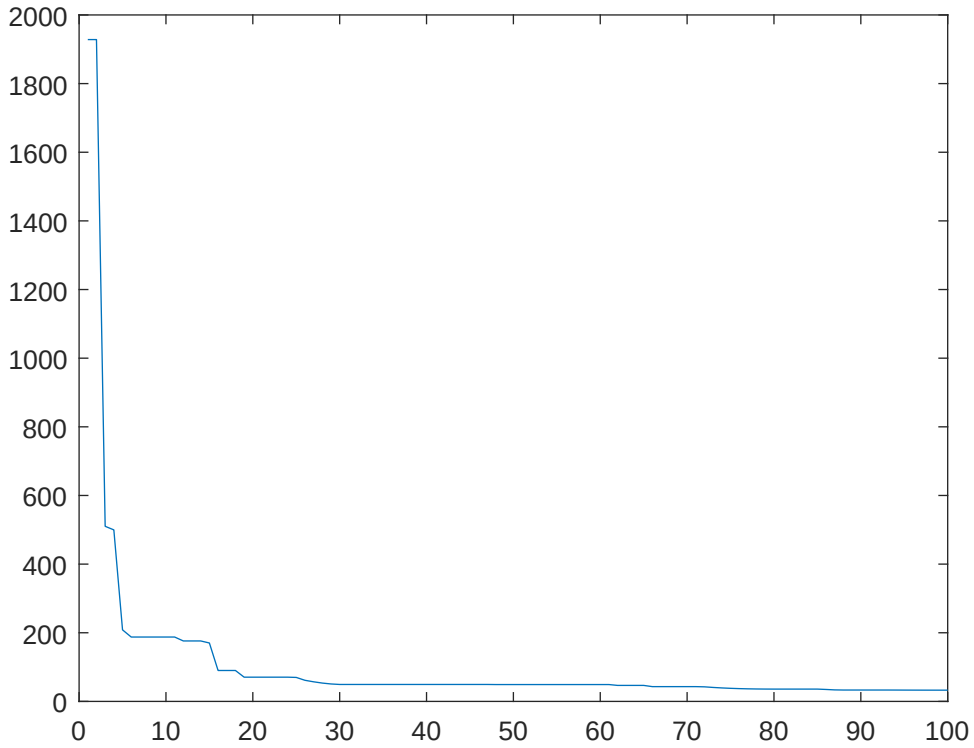
Uczony model będzie drugiego rzędu, który można opisać równaniem

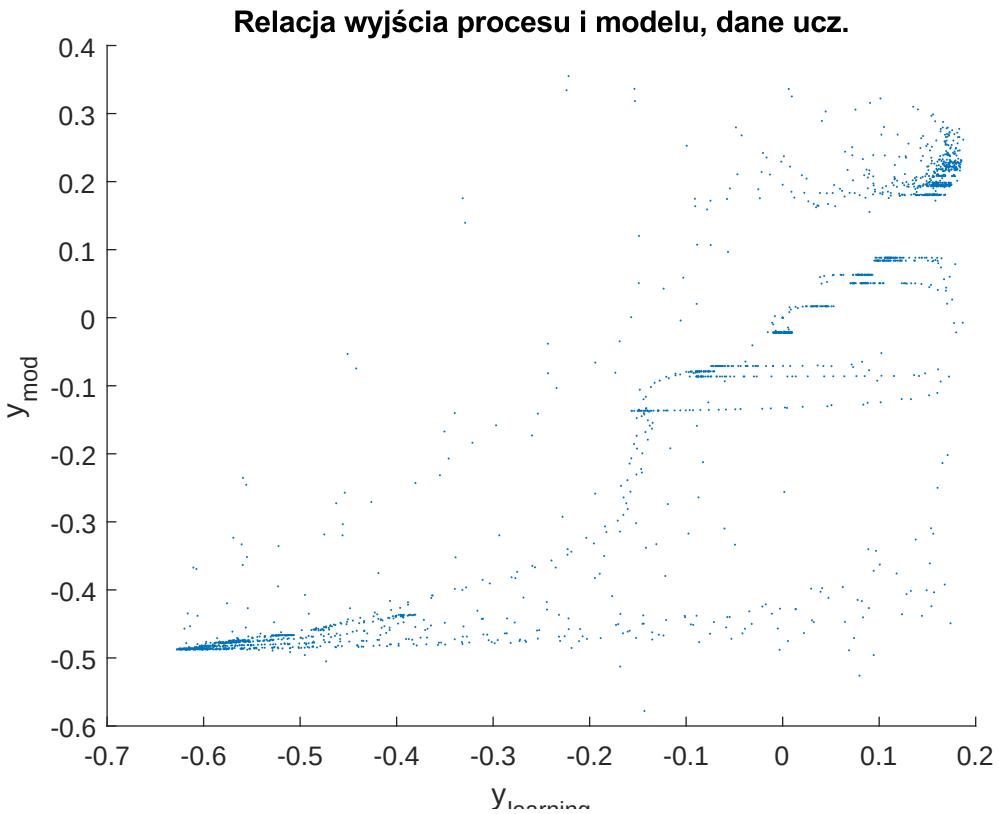
$$y(k) = f(u(k-\tau), u(k-\tau-1), y(k-1), y(k-2)) \tag{4.1}$$

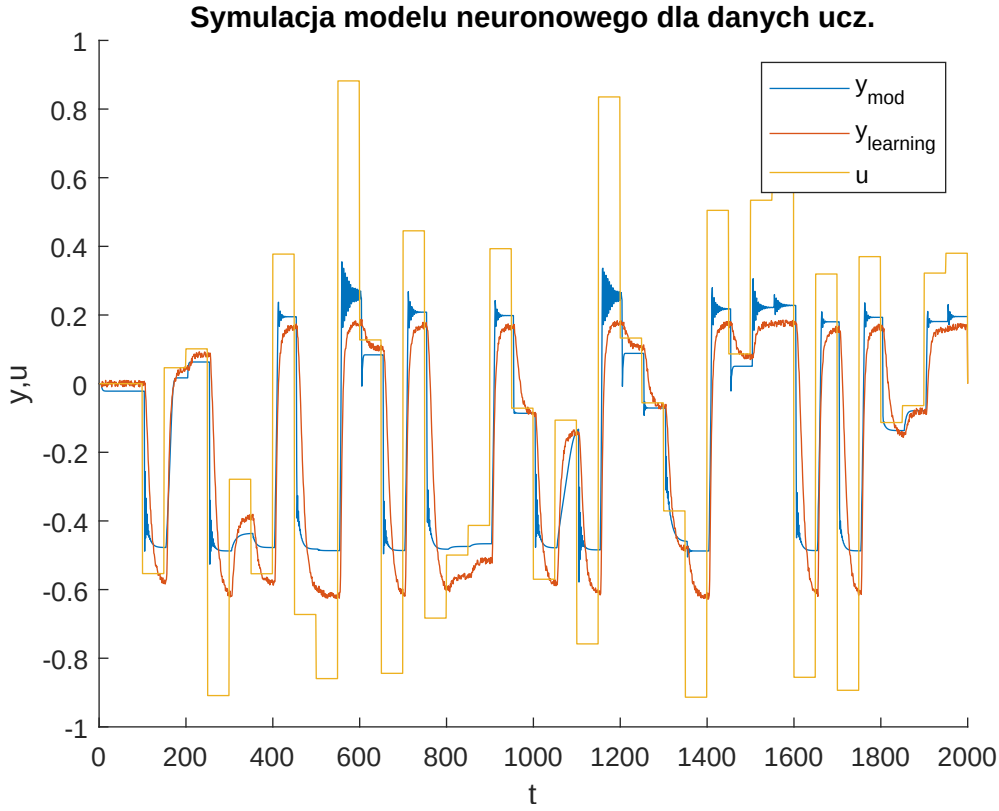
Gdzie $\tau = 5$, tak jak w projekcie nr. 2. Oraz uczenie będzie prowadzone za pomocą algprytmu PSO. Najlepszy model został przedstawiony w tabeli poniżej.

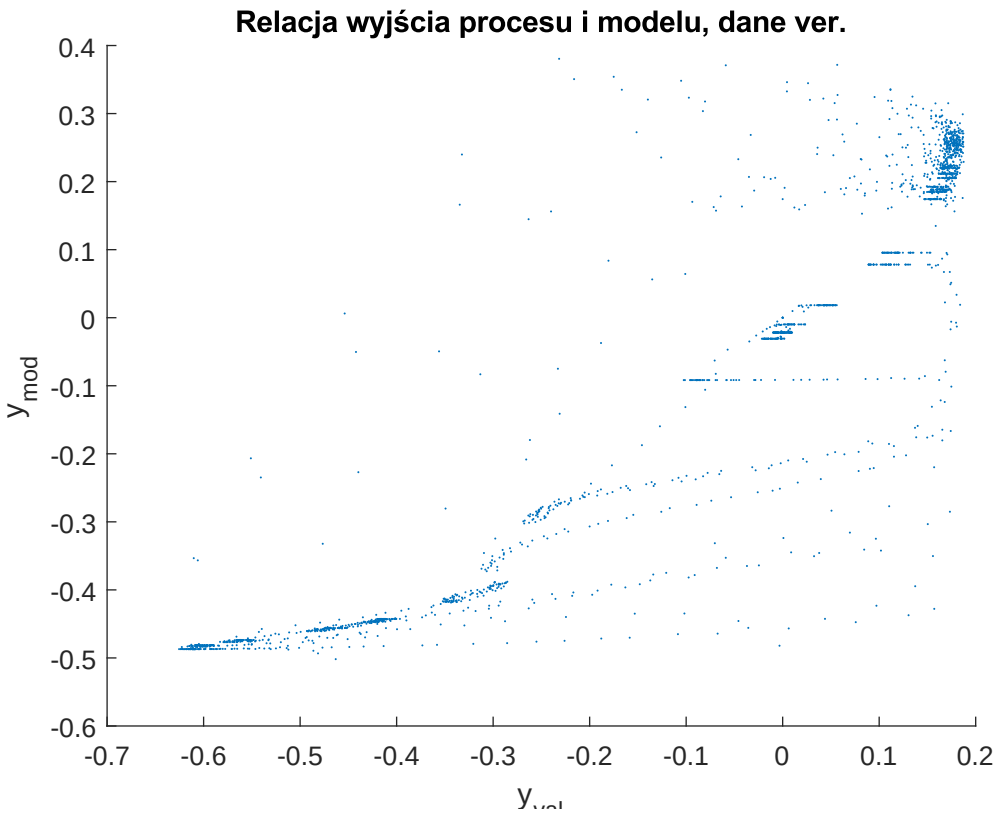
$C_1 = 1.4, C_2 = 0.12, W = 0.9$		
Próba	E_{ucz}	E_{wer}
1	45.7097	94.8137
2	47.2869	84.3509
3	32.7456	48.2534
4	54.3495	78.3467
5	49.4533	55.2232
6	56.9073	87.0342
7	64.5246	67.2341
8	67.6427	82.2375
9	45.8129	62.0775
10	46.7939	225.0479

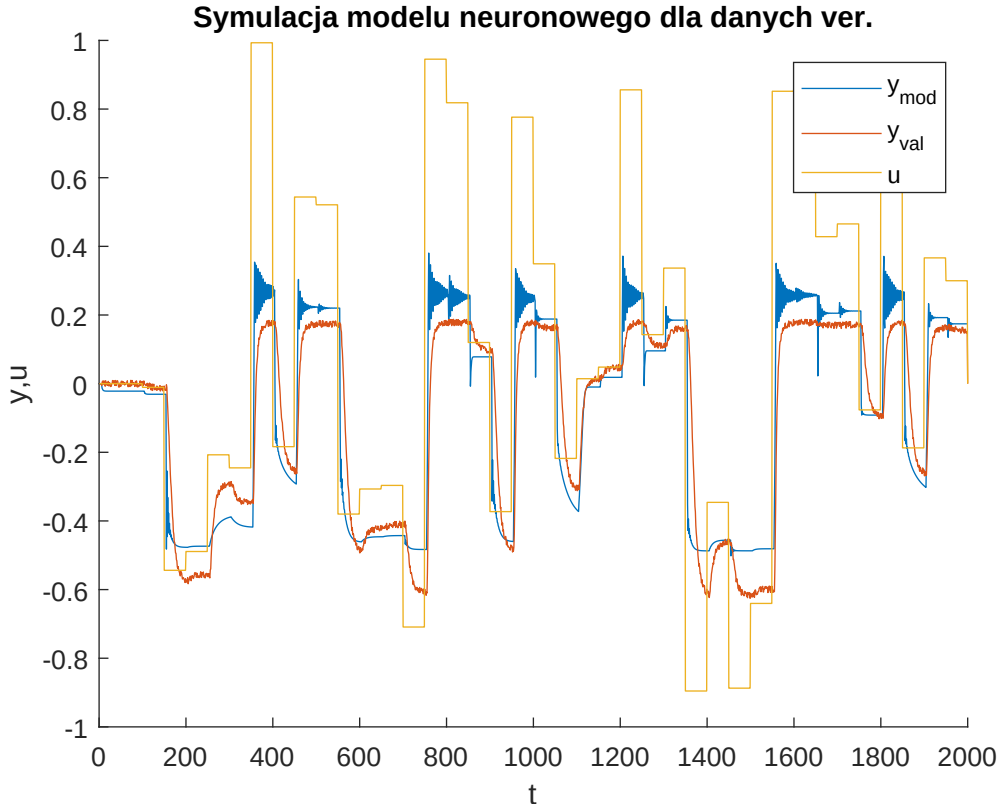
Przebieg nauki oraz działanie regulaotra dla danych uczących oraz weryfikujących przedstawiono poniżej.




Relacja wyjścia procesu i modelu, dane ucz.
y_mod
y_learning
0.4
0.3
0.2
0.1
0
-0.1
-0.2
-0.3
-0.4
-0.5
-0.6
-0.7
-0.6
-0.5
-0.4
-0.3
-0.2
-0.1
0
0.1
0.2


Symulacja modelu neuronowego dla danych ucz.
y_mod
y_learning
u
y,u
t
1
0.8
0.6
0.4
0.2
0
-0.2
-0.4
-0.6
-0.8
-1
0
200
400
600
800
1000
1200
1400
1600
1800
2000


Relacja wyjścia procesu i modelu, dane ver.
y_mod
y_val
0.4
0.3
0.2
0.1
0
-0.1
-0.2
-0.3
-0.4
-0.5
-0.6
-0.7
-0.6
-0.5
-0.4
-0.3
-0.2
-0.1
0
0.1
0.2


Symulacja modelu neuronowego dla danych ver.
y,u
t
y_mod
y_val
u
1
0.8
0.6
0.4
0.2
0
-0.2
-0.4
-0.6
-0.8
-1
0
200
400
600
800
1000
1200
1400
1600
1800
2000

4.2 Wnioski

Widać po błędach dla zbioru werfikującego, że niekonieczene ten regulator byłby lepszy niż zaprezentowany NPL, jednak ma on mniejsze zmiany sterowań. Prawdopodnie model neuronowy byłby lepszy gdyby znaleźć odpowiednie mininum funkcji, która odpowiada za uczenie sieci neuronowej, jednak w tym eksperymencie nie udało się tego zrobić.