POLITECHNIKA WARSZAWSKA

WYDZIAŁ ELEKTRONIKI I TECHNIK INFORMACYJNYCH
INSTYTUT AUTOMATYKI I INFORMATYKI STOSOWANEJ

Krzysztof Ślusarczyk

Zbadanie możliwości implementacji algorytmu predykcyjnego w przemysłowym regulatorze LB – 600

Praca Magisterska

Promotor:

Dr inż. Zygmunt Komor

Ocena .....

.....
Podpis Przewodniczącego
Komisji Egzaminu Dyplomowego

Krzysztof Ślusarczyk

Specjalność podstawowa: Komputerowe Systemy Sterowania

Data urodzenia: 29 lipca 1981

Data rozpoczęcia studiów: 1 października 2000

Życiorys

Urodziłem się 29 lipca 1981 roku w Nisku (woj. podkarpackie). W 1988 roku rozpocząłem naukę w Szkole Podstawowej nr 11 w Stalowej Woli. W okresie ośmiu lat nauki uczestniczyłem w olimpiadach: historycznej, fizycznej i matematycznej.

Kolejnym krokiem mojej edukacji było rozpoczęcie w 1996 roku nauki w Liceum Ogólnokształcącym im. Komisji Edukacji Narodowej w Stalowej Woli na kierunku matematyczno – fizycznym. Szkołę ukończyłem z wyróżnieniem.

Po zdaniu egzaminów maturalnych dostałem się na studia dzienne I stopnia, na Wydział Elektroniki i Technik Informacyjnych Politechniki Warszawskiej na drugą grupę specjalności. Po dwóch latach studiów wybrałem specjalność Komputerowe Systemy Sterowania w Instytucie Automatyki i Informatyki Stosowanej gdzie napisałem pracę dyplomową magisterską.

Interesuję się kulturą Stanów Zjednoczonych, czego wynikiem były dwie podróże do Nowego Jorku w 2002 i 2003 roku.

……

Podpis studenta

EGZAMIN DYPLOMOWY

Złożył egzamin dyplomowy w dniu: …… 2006 roku

z wynikiem: ……

Ogólny wynik studiów: ……..

Dodatkowe wnioski i uwagi Komisji: ……

……..…

…….

Streszczenie

W pracy poruszony jest problem implementacji algorytm predykcyjnego GPC w przemysłowym regulatorze LB – 600. Rozważana jest analityczna postać prawa regulacji, ograniczenia nałożone na sygnały sterujące oraz wpływ zakłóceń niemierzalnych. Regulowany obiekt został zamodelowany w postaci jednej inercji z opóźnieniem. Rozważany jest model dwuwymiarowy.

W pracy jest opisywana ilość zmiennych wykorzystanych przez regulator. Istotną część stanowi próba modyfikacji algorytmu tak, aby ograniczyć ilość zmiennych, przy jednoczesnym zachowaniu dobrej jakości regulacji i akceptowalnym stopniu komplikacji algorytmu. Przy analizie wykorzystania zmiennych uwzględniono cykl obliczeń regulatora oraz wpływ okresu próbkowania obiektu. Przytoczone wyniki symulacji pokazują, że wprowadzone ulepszenia pozwalają znacząco ograniczyć ilość zmiennych, nie pogarszając jakości regulacji i nie komplikując w dużym stopniu algorytmu. Analiza wykorzystanych zmiennych jest jedną z ważniejszych części pracy. Inna ważna jej część to algorytm regulacji napisany w języku C., uwzględniający budowę strukturalną regulatora. Wybór języka programowania był podyktowany koniecznością współpracy z informatykiem dokonującym umieszczenia kodu algorytmu w fizycznym urządzeniu.

***

Abstract

The Thesis brings up the issue of implementation a GPC predictive algorithm into industrial LB – 600 controller. Analytical version of algorithm was considered together with limitation for steering signal and influence of non-measured disturbance.

Important part of the thesis is evolving over modifications of existing algorithm in order to save memory space used for variables but at the same time takes into consideration complexity of algorithm. Presented results of performed simulations seems to confirms fact that made modifications limits the number of variables and do not complicate algorithm too much.

Spis treści:

Rozdział 1

Wprowadzenie

W ostatnich latach zauważa się rosnące zainteresowanie nowoczesnymi technikami regulacji. Rozwój mocy obliczeniowych komputerów jak również powszechna ich dostępność sprawiły, że coraz większa ilość producentów urządzeń regulacji automatycznej zastanawia się nad uwzględnieniem w swojej ofercie jednej z tzw. *zaawansowanych technik regulacji*.

Przed wielu laty również w Polsce, w firmie Mera – Pnefal były podejmowane przygotowania odpowiedniej oferty. Starania te zakończyły się fiaskiem z kilku dobrze znanych powodów, wśród których można wskazać niedostateczne w tamtym czasie zainteresowanie przemysłu, jak i brakiem wnikliwych badań algorytmów pod kątem wymogów konkretnych implementacji.

Większość wdrożeń zaawansowanych układów regulacji predykcyjnej dotyczy obiektów, które posiadają gotowe, pracujące struktury regulacji klasycznej oparte na regulacji jednopętlowej typu PID. Celem wdrażania i implementowania regulacji zaawansowanej jest *wyraźna poprawa jakości regulacji* w warunkach pracy zbliżonych do nominalnych, dla trudniejszych obiektów (np. z dużym opóźnieniem), a szczególnie w przypadkach występowania silnych interakcji w obiekcie wielowymiarowym oraz występowania ograniczeń sterowania. Sytuacje te są najczęściej zbyt wymagające dla klasycznych struktur regulacji np. PID. Przy tym wdrażane są najczęściej układy regulacji zaawansowanej oparte na modelach liniowych (opis obiektu w nominalnym punkcie pracy). Wykorzystanie odpowiedniego modelu obiektu ma o tyle wielkie znaczenie, że regulacja predykcyjna może przynieść spodziewaną poprawę jakości regulacji jedynie w przypadku dobrze zidentyfikowanego obiektu.

W wielu pracach można znaleźć informacje na temat rezultatów uzyskanych w wyniku wdrożenia nowoczesnych technik regulacji predykcyjnej [Jar03, Nun01, BL93, BGD05], mało jest jednak prac, które prezentują sam sposób implementacji oraz trudności, jakie temu towarzyszą.

Rozważania niniejszej pracy dotyczą właśnie przedstawionej sytuacji. Pokazano wymagania jakie musi spełniać regulator przemysłowy (LB – 600 firmy LAB-EL), aby można było skutecznie zaimplementować w urządzeniu jedną z zaawansowanych technik regulacji predykcyjnej. Praca powinna służyć przekonaniu konstruktorów do podjęcia wysiłku implementacji algorytmu.

Przedstawione w pracy wzory, opisy budowy urządzenia oraz cytaty pochodzą z opracowań, niezbędne dla czytelności materiału.

Cała praca podzielona jest na siedem rozdziałów. Krótki ich opis został zamieszczony poniżej.

W rozdz. 2 opisano sposób modelowania obiektu. Dokładny opis obiektu, w przypadku regulacji predykcyjnej z przesuwanym horyzontem, ma kluczowe znaczenie dla osiągnięcia dobrych rezultatów regulacji. Pokazane zostało podejście inżyniera – konstruktora, prezentujące

obiekt w postaci dyskretnego równania różnicowego, opisującego obiekt w postaci wejście – wyjście (transmitancja dyskretna).

Rozdz. 3 daje wprowadzenie w teorię regulacji predykcyjnej. W pierwszej części opisana została ogólna zasada regulacji predykcyjnej z przesuwanym horyzontem. W kolejnej części zostały opisane najczęściej stosowane algorytmy regulacji predykcyjnej. W kolejnej części zaprezentowano algorytm DMC w wersji analitycznej oraz analityczne prawo regulacji dla algorytmu GPC. Pominięto jednak wyprowadzenie wzorów na prawa regulacji. Takie wyprowadzenie wykracza poza zakres tej pracy. Zainteresowanych można odesłać do prac [Plam06] lub [Mar02], gdzie znajduje się szczegółowe wyprowadzenie wzorów na prawo regulacji dla algorytmu GPC jak i DMC.

Początek rozdz. 4 traktuje o budowie regulatora LB – 600 firmy LAB – EL. Krótko opisano moduły z jakich składa się urządzenie. Kolejna część rozdziału to spojrzenie na regulator pod kątem dostępnej do wykorzystania pamięci i szybkości obliczeń. W ostatniej części rozdziału przedstawiona została jedna z funkcji urządzenia: eksperyment identyfikacji obiektu.

W rozdz. 5 dokonano analizy algorytmu predykcyjnego pod względem zajętości pamięci. Na podstawie wstępnych wyników analizy zaproponowano algorytm, który może posłużyć do implementacji w regulatorze. W dalszej kolejności starano się ulepszyć zaproponowany algorytm, tak aby zredukować ilość zmiennych, ale zbytnio nie komplikować istniejącego algorytmu. Starano się wziąć pod uwagę stopień komplikacji nowej propozycji. Ostatnia część rozdziału to opis wpływu czasu próbkowania obiektu na implementowany algorytm.

Kolejny rozdział przedstawia zasoby krytyczne regulatora LB – 600, jakimi są pojemność pamięci oraz cykl obliczeń procesora.

Rozdział 7 zawiera opis implementacji algorytmu w programie Matlab oraz jego najważniejsze części składowe

Wyniki symulacji dla przykładowego obiektu przedstawiono w rozdz. 6. Do symulacji użyto modelu dwuwymiarowego, w postaci jednej inercji i opóźnienia dla każdej z par wejście – wyjście.

Autor pracy pragnie podziękować panu dr inż. Piotrowi Marusakowi z Instytutu Automatyki i Informatyki Stosowanej Politechniki Warszawskiej za okazaną pomoc przy implementacji algorytmu oraz cenne wskazówki pozwalające na szczegółowe przedstawienie zagadnienie zajętości pamięci na zmienne.

1.1 Cel pracy

W niniejszej pracy rozważany jest obiekt dwuwymiarowy (dwa wejścia i dwa wyjścia) w postaci jednej inercji i opóźnienia. Wszelkie analizy dotyczące praw regulacji i analizy zajętości można jednak łatwo przenieść na modele bardziej skomplikowane, jak również na modele o większej wymiarowości. Praca skupia się na modelu obiektu zapisanym w postaci równań transmitancyjnych. Z tego powodu w naturalny sposób został wybrany algorytm GPC.

Praca pokazuje ilość zmiennych używanych przez algorytm oraz próbuje odpowiedzieć na pytanie: „W jaki sposób można ograniczyć ilość tych zmiennych?” Oczywiście ograniczenie ilości zmiennych w algorytmie pociąga za sobą stopień jego komplikacji, co może zniechęcić konstruktorów regulatora do implementacji algorytmu.

Praca bazuje na znajomości budowy regulatora. W ten sposób możliwe było wykonanie jednego krok naprzód: nie ograniczanie się jedynie do analizy samego algorytmu, ale zastanowienie się nad sposobem działania urządzenia tak, aby implementacja algorytmu stała się możliwa.

Rozdział 2

Sposób modelowania obiektu

2.1 Wybór modelu obiektu

W pracy założono, że w badaniach ograniczymy się do obiektów o dwóch wyjściach regulowanych ($n_y = 2$) oraz dwóch sterowaniach ($n_u = 2$). Uogólnienie na większą wymiarowość nie wpływa w żaden sposób na schemat postępowania. Założono również, że transmitancja dla każdej pary wejście – wyjście jest opisana w postaci jednej inercji oraz opóźnienia:

$$H(s) = \frac{K_{ob} \cdot e^{-T_0 s}}{1 + T_1 \cdot s}$$

Założenie to można uogólnić na bardziej skomplikowane obiekty (np. dwuinercyjne), por rozdz. 5

Ważniejszy, z punktu widzenia wyboru algorytmu, jest jednak sposób opisu obiektu. Każdy z algorytmów uwzględnia inne właściwości stosowanego opisu. I tak algorytmy predykcyjne z przesuwanym horyzontem korzystają najczęściej z opisu obiektu w postaci :

- odpowiedzi skokowej,
- opisu transmitancyjnego,
- opisu w postaci równań stanu.

Oczywiście z opisu transmitancyjnego możemy uzyskać odpowiedź skokową obiektu (podobnie z wartości odpowiedzi skokowej możemy uzyskać opis transmitancyjny). Do modeli opisanych w powyższy sposób, stosuje się odmienne algorytmy:

- algorytm DMC (*ang. Dynamic Matrix Control*) do modeli opisanych odpowiedzią skokową lub impulsową,

- algorytm GPC (*ang. Generized Predictive Control*) do modeli w postaci dyskretnych równań różnicowych (opis transmitancyjny),

- algorytm MPCS (*ang. Model Predictive Control with State equations)* do modeli w postaci równań stanu.

Ze względu na faktu, że rozważany jest ten sam obiekt, sposób w jaki będziemy go opisywać nie ma większego znaczenia, jeśli chodzi o jakość regulacji. Ma to natomiast duże znaczenie ze względu na zrozumienie przez osoby odpowiedzialne za wdrażanie i implementację algorytmu.

W przypadku regulatora LB – 600 osobą odpowiedzialną za implementację algorytmu będzie konstruktor – automatyk. Dlatego w pracy zdecydowano się na opis transmitancyjny. Powodem dla którego tak postąpiono jest fakt, że dla inżynierów opis transmitancyjny jest zwięzły i zrozumiały. Pociąga jednakże za sobą konieczność przeprowadzenia identyfikacji obiektu. Ten krok można jednak ominąć, bo urządzenie jest wyposażone w funkcję identyfikacji obiektu.

2.2 Wybór algorytmu predykcyjnego

Opierając się na opisie transmitancyjnym obiektu zdecydowano się uwzględnić w dalszych rozważaniach algorytm typu GPC. Wykorzystuje on model w postaci dyskretnego równania różnicowego opisującego relację wejście – wyjście (transmitancji dyskretnej). Jego zaletą jest ”oszczędność” reprezentacji (w stosunku do algorytmu DMC, bazującego na odpowiedzi skokowej).

W stosunku do opisu w postaci odpowiedzi skokowej (na której bazuje regulator DMC) opis w postaci transmitancji pozwala na objęcie rozważaniami szerszej klasy obiektów, włącznie z obiektami niestabilnymi (z biegunami transmitancji dyskretnej o module większym od jedności). Dla zapewnienia poprawnej pracy regulatora ważny jest wówczas właściwy wybór *horyzontów sterowania* i *predykcji* (muszą one być odpowiednio długie tak, aby wytłumić wpływ niestabilnych biegunów obiektu w czasie nie przekraczającym długości *horyzontu predykcji*).

Łatwo można pokazać, że przy dostatecznie wysokim rzędzie (równym ilości współczynników w odpowiedzi skokowej) sygnału wejściowego transmitancja dyskretna może bezbłędnie reprezentować dowolną odpowiedź skokową. W takim przypadku wartości współczynników w transmitancji dyskretnej są równe różnicy kolejnych współczynników z modelu odpowiedzi skokowej.

Model w postaci równania różnicowego pozwala na odwzorowanie dowolnej odpowiedzi skokowej w najgorszym wypadku przy użyciu takiej samej, jak w modelu odpowiedzi skokowej liczby parametrów. W znakomitej jednak większości liczba parametrów modelu w postaci transmitancji dyskretnej może być mniejsza niż liczba współczynników odpowiedzi skokowej, co pozwala na inne ujęcie modelu, w formie bardziej zwięzłej i oszczędnej z punktu widzenia zasobów pamięci (patrz rozdział 5).

Rozdział 3

Algorytm regulacji predykcyjnej z przesuwanym horyzontem

3.1 Zasada działania algorytmów predykcyjnych z przesuwanym horyzontem

Algorytmy predykcyjne z przesuwanym horyzontem charakteryzują się tym, że podczas wyznaczania sterowania są w nich brane pod uwagę nie tylko informacje z bieżącej chwili, ale także przewidywane wartości wyjścia w przyszłości, na wiele chwil do przodu (na tzw. *horyzoncie predykcji*). Predykcja w chwilach przyszłych dokonywana jest na podstawie dostępnych informacji o obiekcie, o występujących ograniczeniach, o przewidywanych (przyszłych) zakłóceniach i wartościach zadanych oraz innych informacji mogących poprawić jakość prognozy. Innymi słowy możliwe jest użycie całej dostępnej wiedzy (o obiekcie i układzie regulacji), podczas opracowania regulatora, a algorytmy predykcyjne z przesuwanym horyzontem, dzięki ich strukturze pozwalają uczynić to możliwie efektywnie. Przyszłe sterowania są przez regulator wyznaczane tak, aby przewidywane zachowanie układu regulacji spełniało założone kryteria.

Ogólna zasada działania regulacji predykcyjnej z przesuwanym horyzontem jest następująca:
w każdej iteracji algorytmu, czyli w każdej kolejnej chwili kT_p (gdzie T_p oznacza okres próbkowania , czyli okres powtarzania interwencji regulatora, $k = 0,\ldots$) , dysponując:

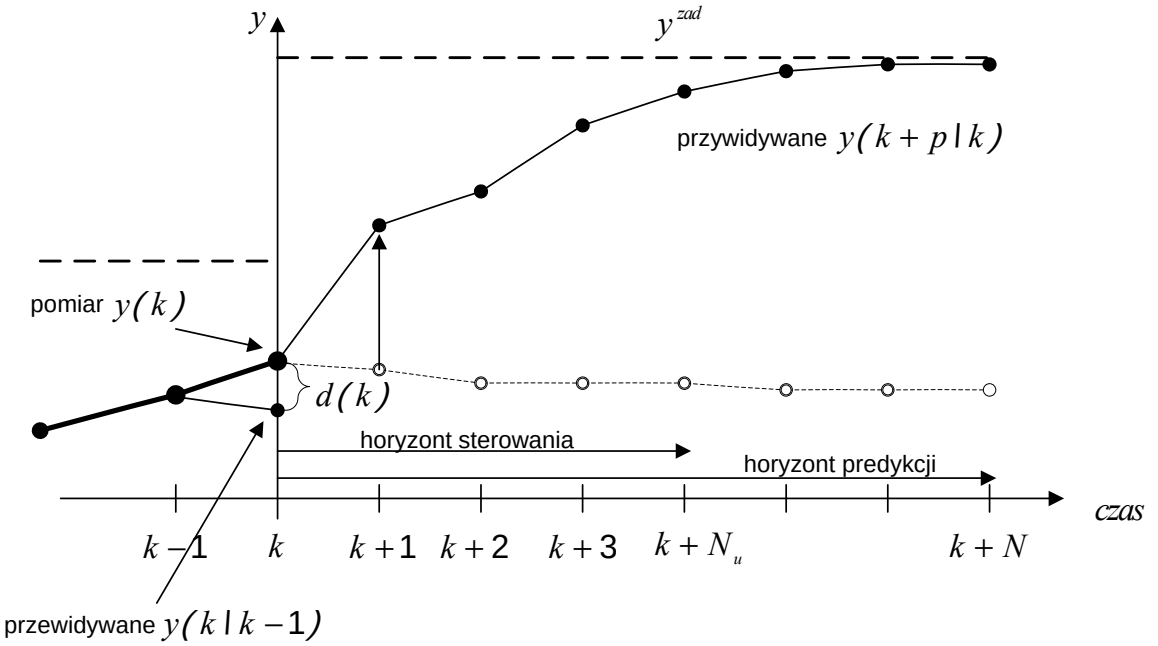
- dynamicznym modelem obiektu
- pomiarami zmiennych wyjściowych układu w chwili bieżącej i poprzednich
- poprzednimi wartościami sterowania
- znaną (lub założoną) trajektorią wartości zadanych wyjść regulowanych obiektu w chwili bieżącej i chwilach przyszłych

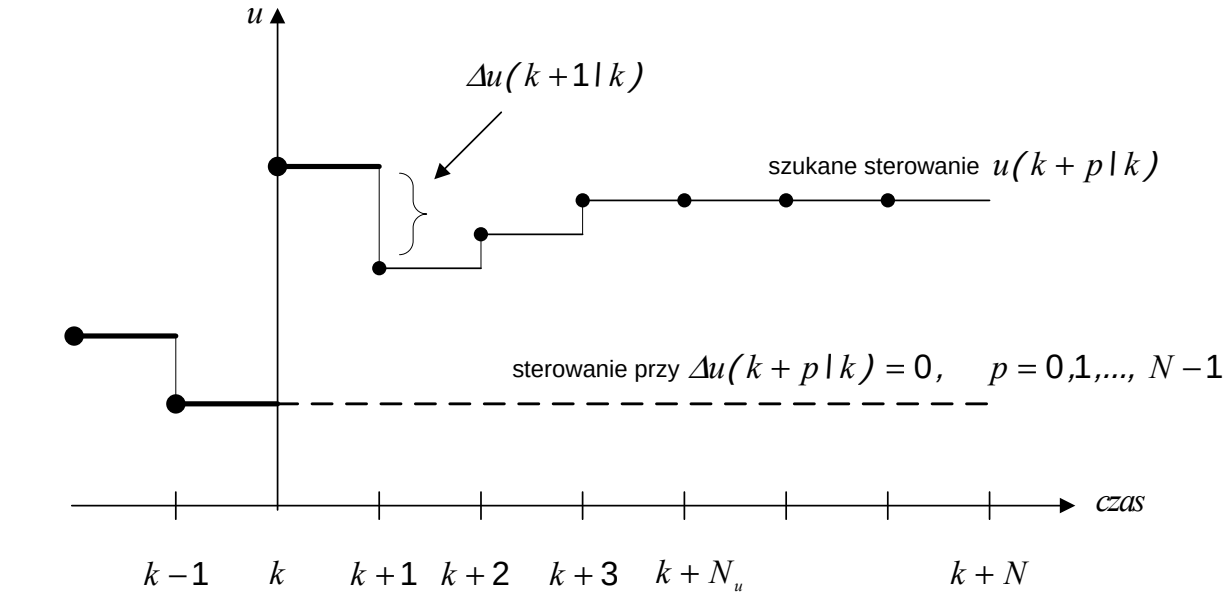
wyznaczamy wartość sterowań w bieżącej chwili, jak i wartość sterowania w chwilach przyszłych (do N_u - horyzontu sterowania). Sterowanie wyznaczane jest tak, aby zminimalizować różnice między wartościami regulowanych wyjść obiektu, przewidywanymi w chwili k na chwilę $k + p$, a wartościami zadanymi dla tych wyjść (też na chwilę $k + p$), na horyzoncie predykcji N ($p = 1,2,\ldots,N$). Minimalizacja różnic rozumiana jest w sensie określonego kryterium jakości regulacji. Do sterowania wyznaczany jest jedynie pierwszy element tak wyznaczonego optymalnego ciągu wartości sterowań. W kolejnej chwili $(k+1)T_p$

następuje nowy pomiar wyjścia obiektu i cała procedura jest powtarzana, z horyzontem predykcji o niezmienionej długości N. Stosuje się więc zasadę przesuwanego horyzontu, zwaną też zasadą sterowania repetycyjego [Fin74]. Cała procedura wyznaczania bieżącej wartości sterowania powtarzana jest w każdej kolejnej iteracji algorytmu (w każdej z kolejnych chwil kT_p).

W algorytmach predykcyjnych wykorzystujących modele typu wejście – wyjście najczęściej używanym wskaźnikiem jakości jest:

$$J(k) = \left\| y^{zad}(k) - y^0(k) - M\Delta U(k) \right\|^2_{\underline{\Psi}} + \left\| \Delta U(k) \right\|^2_{\underline{\Lambda}}$$





Rys. 3.1: Zasada regulacji predykcyjnej

Algorytmy predykcyjne, dzięki ich cechom, warto stosować w przypadkach, gdy:

- obiekt zawiera duże opóźnienia,
- sterowania są ograniczone,
- istnieją silne interakcje w obiekcie wielowymiarowym (algorytmy predykcyjne mają silne właściwości odsprzęgające),
- można przewidzieć przyszłe zmiany wartości zadanej (np. regulacja programowa),
- można przewidzieć zmiany ograniczeń,
- można przewidzieć wpływ zakłóceń.

Algorytmy predykcyjne z kwadratowym wskaźnikiem jakości występują w dwóch odmianach: numerycznej i analitycznej. Rozwiązanie numeryczne bazuje na rozwiązaniu zadania optymalizacji kwadratowej z liniowymi ograniczeniami, w każdym kroku algorytmu. Wymaga większych zasobów pamięci ze względu na optymalizację wartości sterowania oraz większych nakładów obliczeń niż algorytm w wersji analitycznej. Właśnie ze względu na prostotę i często dobre wyniki symulacji (optymalne sterowanie) w mniej skomplikowanych urządzeniach przemysłowych implementowane są algorytmy analityczne [Mar02]. Poniższy opis algorytmów predykcyjnych dotyczy właśnie takiego opisu algorytmu. Szerszy opis wersji numerycznej można znaleźć w [Tat02].

3.2 Przegląd algorytmów predykcyjnych

W rozdziale przedstawiono dwie najczęściej spotykane odmiany algorytmów predykcyjnych z przesuwanym horyzontem: algorytm GPC i algorytm DMC.

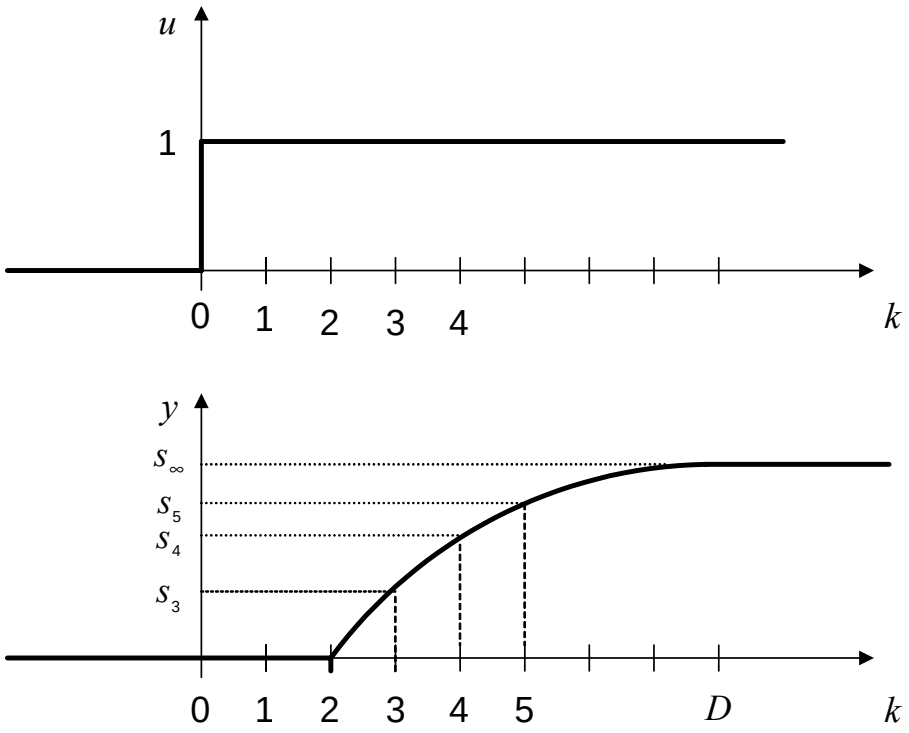
Przedstawione wzory oraz szczegółowy opis ich wyprowadzenia można znaleźć w [Tat02], jednak ze względu na fakt analizy zapotrzebowania na zasoby (w przypadku algorytmu GPC) zdecydowano się na powtórzenie istotnych fragmentów.

Niniejszy rozdział pokazuje jak można implementować algorytmu DMC w wersji analitycznej. Wzory, opisujące prawo regulacji uzupełniono przykładem obliczeń dla założonego obiektu.

3.2.1 Algorytm DMC w wersji analitycznej

Regulator DMC (ang. *Dynamic Matrix Control*) był jednym z pierwszych algorytmów regulacji predykcyjnej. Jego początki sięgają lat 70 – tych ubiegłego stulecia. Wtedy też po raz pierwszy użyto go dla potrzeb przemysłu chemicznego. Od tamtego czasu algorytm przeszedł wiele udoskonaleń, początkując też szybki rozwój algorytmów predykcyjnych z przesuwanym horyzontem. W algorytmie DMC dynamika obiektu regulacji modelowana jest dyskretnymi odpowiedziami skokowymi wiążącymi wyjściami obiektu z jego wejściami. Jest to bardzo prosty i wygodny sposób modelowania dynamiki obiektu. Model obiektu można otrzymać na drodze prostego eksperymentu polegającego na dokonaniu wymuszenia skokowego na wejściu obiektu. Prostota modelowania przyczyniła się do popularyzacji algorytmu DMC; nie bez znaczenia pozostaje też fakt, że algorytm ten jest najstarszym i przez to jednym z najlepiej znanych algorytmów regulacji predykcyjnej. Taka forma modelowania obiektu pozwala na implementację algorytmu osobom nieznającym technik identyfikacji obiektu.

Przykładowa odpowiedź skokowa obiektu przedstawia się następująco:



Rys. 3.1: Przykład odpowiedzi wyjścia obiektu y na skok jednostkowy wejścia u

W oparciu o model w postaci odpowiedzi skokowej można dokonywać predykcji wyjścia obiektu na kolejne chwile w przyszłości. Wyprowadzenie wzoru na analityczną postać prawa regulacji można znaleźć w [Tat02]. Poniżej zdecydowano się jedynie na przedstawienie poszczególnych równań prawa regulacji. Czytelnik może posłużyć się nimi przy implementacji algorytmu, bez konieczności zgłębiania żmudnego wyprowadzania wzorów.

Poniższe wzory są prawdziwe przy następujących założeniach:

- założono, że nie jest znany model zakłócenia i przyjęto, że zakłócenie nie będzie się zmieniać na horyzoncie predykcji N
- przyjęto uproszczenie macierzy wag: $\underline{\Psi} = I$ oraz $\underline{\Lambda} = \lambda I$ (por rozdz. 3.2.2)

Zaprezentowane wzory dotyczą obiektu wielowymiarowego. Z reguły przejście z obiektu wielowymiarowego na jednowymiarowy jest prostsze niż w kierunku odwrotnym – z obiektu jednowymiarowego do wielowymiarowego. Dlatego próba implementacji algorytmu dla obiektu o jednym wejściu i jednym wyjściu nie powinna nastręczać problemów. Na zakończenie rozdziału przepisano te wzory dla obiektu o dwóch wejściach i dwóch wyjściach regulowanych.

Algorytm DMC bazuje na modelu obiektu w postaci odpowiedzi skokowych:

$$S_m = \begin{bmatrix} s_m^{11} & s_m^{12} & s_m^{13} & \cdots & s_m^{1n_u} \\ s_m^{21} & s_m^{22} & s_m^{23} & \cdots & s_m^{2n_u} \\ s_m^{31} & s_m^{32} & s_m^{23} & \cdots & s_m^{3n_u} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ s_m^{n_y 1} & s_m^{n_y 2} & s_m^{n_y 3} & \cdots & s_m^{n_y n_u} \end{bmatrix}, \quad m = 1,2,\ldots D \tag{3.1}$$

λ jest dostrajanym parametrem.
N_1 to pierwszy niezerowy współczynnik odpowiedzi skokowej
N - horyzont predykcji
N_u - horyzont sterowania

Element s^{ij} podmacierzy S jest odpowiedzią i-tego wyjścia na skok jednostkowy na j-tym wejściu, przy pozostałych wejściach ustalonych. W przypadku jednowymiarowym macierz S jest jednoelementową wartością skalarną, podczas gdy w przypadku wielowymiarowym zawiera poszczególne współczynniki będące wartościami skalarnymi.
Wprowadźmy dodatkową macierz:

$$M = \begin{bmatrix} S_{N_1} & 0 & 0 & 0 & \cdots & 0 \\ S_{N_1+1} & S_{N_1} & 0 & 0 & \cdots & 0 \\ S_{N_1+2} & S_{N_1+1} & S_{N_1} & 0 & \cdots & 0 \\ S_{N_1+3} & S_{N_1+2} & S_{N_1+1} & s_{N_1} & \cdots & 0 \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ S_{N_u+N_1-1} & S_{N_u+N_1-2} & S_{N_u+N_1-3} & s_{N_u+N_1-4} & \cdots & S_{N_1} \\ S_{N_u+N_1} & S_{N_u+N_1-1} & S_{N_u+N_1-2} & S_{N_u+N_1-3} & \cdots & S_{N_1+1} \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ S_N & S_{N-1} & S_{N-2} & S_{N-3} & \cdots & S_{N-N_u+1} \end{bmatrix} \tag{3.2}$$

Macierz M jest nazywana *macierzą dynamiczną*. Zawiera ona podmacierze odpowiedzi skokowych. W przypadku jednowymiarowym macierz M zawiera poszczególne współczynniki będące wartościami skalarnymi, podczas gdy w przypadku wielowymiarowym zawiera podmacierze odpowiedzi skokowych.

Przed przystąpieniem do właściwej regulacji, należy obliczyć macierz pomocniczą K. Oblicza się jeden raz, w tzw. *„trybie off – line”* :

$$K = \left[M^T M + \lambda I\right]^{-1} M^T \tag{3.3},$$

gdzie M jest macierzą dynamiczną obliczoną według wzoru (3.2).

Po obliczeniu macierzy K można przystąpić do właściwych obliczeń.

W każdym kroku algorytmu wyznaczana jest macierz zmian sterowania.

$$\Delta\hat{U}_k = K \left\| Y_k^{zad} - Y_k - M^P \Delta U_k^P \right\| \quad \text{, gdzie}$$

$$Y_k^{zad} = \begin{bmatrix} y_{k+N_1/k}^{(1)zad} \\ \vdots \\ y_{k+N_1/k}^{(n)zad} \\ \\ \vdots \\ \\ y_{k+N/k}^{(1)zad} \\ \vdots \\ y_{k+N/k}^{(n)zad} \end{bmatrix}, \qquad Y_k = \begin{bmatrix} y_k^{(1)} \\ \vdots \\ y_k^{(n)} \\ \\ \vdots \\ \\ y_k^{(1)} \\ \vdots \\ y_k^{(n)} \end{bmatrix}, \qquad \Delta U_k^P = \begin{bmatrix} \Delta u_{k-1} \\ \dots \\ \Delta u_{k-(D-1)} \end{bmatrix} = \begin{bmatrix} \Delta u_{k-1}^{(1)} \\ \vdots \\ \Delta u_{k-1}^{(n)} \\ \\ \vdots \\ \\ \Delta u_{k-(D-1)}^{(1)} \\ \vdots \\ \Delta u_{k-(D-1)}^{(n)} \end{bmatrix},$$

oraz

$$M^P = \begin{bmatrix} S_{1+N_1} - S_1 & S_{2+N_1} - S_2 & \cdots & S_{D-1+N_1} - S_{D-1} \\ S_{2+N_1} - S_1 & S_{3+N_1} - S_2 & \cdots & S_{D+N_1} - S_{D-1} \\ \vdots & \vdots & \ddots & \vdots \\ S_{N+1} - S_1 & S_{N+2} - S_2 & \cdots & S_{N+D-1} - S_{D-1} \end{bmatrix},$$

Macierz M^P wyznacza predykcje wyjść w zależności jedynie od przeszłych (ang. *Past*) przyrostów sterowań.

Y_k^{zad} - macierz wartości zadanych dla poszczególnych wyjść, $y_{k+N_1/k}^{(1)zad}$ - wartość zadana dla wyjścia numer 1 przewidywana w chwili k na chwilę $k + N_1$

Y_k - macierz wyjść regulowanych zmierzonych w chwili k, $y_k^{(1)}$ - wyjście numer 1 zmierzone w chwili k

ΔU_k^P - macierz zmian wartości sterowania w chwili k, $\Delta u_{k-1}^{(1)}$ - zmiana wartości pierwszego sterowania w chwili $k-1$ ($\Delta u_{k-1}^{(1)} = \Delta u_{k-1}^{(1)} - \Delta u_{k-2}^{(1)}$)

D - horyzont dynamiki:

Dla rzeczywistych obiektów stabilnych (bez całkowania) stan wyjścia po wymuszeniu skokowym ustala się, $lim_{k\to\infty} s_k^{ij} = s_\infty^{ij}$. Daje to duże uproszczenie algorytmu, bo wystarczy znać D współczynników odpowiedzi skokowej, tzn. liczbę kroków dyskretyzacji, po której wartość odpowiedzi skokowej można uznać za ustaloną, równą wzmocnieniu statycznemu obiektu $k_m = s_\infty^{ij}$.
D jest horyzontem dynamiki wspólnym dla wszystkich odpowiedzi, tzn $s_m^{ij} = const$, dla $m \geq D$.

Prawo regulacji zapisane w powyższy sposób zobrazowano na przykładzie obiektu dwuwymiarowego.

Przykład 3.1

Rozważmy obiekt o dwóch wejściach i dwóch wyjściach, o wartościach odpowiedzi skokowych równych s_1^{ij}(czyli pierwszy element odpowiedzi i- tego wyjścia na j- te wejście). Załóżmy, że parametry układu regulacji są następujące: $N = 6$, $N_u = 3$, $N_1 = 2$ (w obiekcie występują opóźnienia), $D = 6$. Dodatkowo wartości zadane w chwili k dla poszczególnych wyjść wynoszą: $y_k^{(1)zad}$ oraz $y_k^{(2)zad}$.

$$M = \begin{bmatrix} S_2 & 0 & 0 \\ S_3 & S_2 & 0 \\ S_4 & S_3 & S_2 \\ S_5 & S_4 & S_3 \\ S_6 & S_5 & S_4 \end{bmatrix}, \qquad M^P = \begin{bmatrix} S_3 - S_1 & S_4 - S_2 & \cdots & S_7 - S_5 \\ S_4 - S_1 & S_5 - S_2 & \cdots & S_8 - S_5 \\ \vdots & \vdots & \ddots & \vdots \\ S_7 - S_1 & S_8 - S_2 & \cdots & S_{11} - S_5 \end{bmatrix}$$

Elementy podmacierzy S_m wynoszą odpowiednio :

$$S_m = \begin{bmatrix} s_m^{11} & s_m^{12} \\ s_m^{21} & s_m^{22} \end{bmatrix}, \qquad \text{gdzie } m = 1,2,\ldots 6$$

W chwili k wartości macierzy: wartości zadanych, wyjść układu regulacji i zmian sterowania będą wynosić odpowiednio:

$$Y_k^{zad} = \begin{bmatrix} y_{k+2}^{(1)zad} \\ y_{k+2}^{(2)zad} \\ y_{k+3}^{(1)zad} \\ y_{k+3}^{(2)zad} \\ y_{k+4}^{(1)zad} \\ y_{k+4}^{(2)zad} \\ y_{k+5}^{(1)zad} \\ y_{k+5}^{(2)zad} \\ y_{k+6}^{(1)zad} \\ y_{k+6}^{(2)zad} \end{bmatrix}, \quad Y_k = \begin{bmatrix} y_k^{(1)} \\ y_k^{(2)} \\ y_k^{(1)} \\ y_k^{(2)} \\ y_k^{(1)} \\ y_k^{(2)} \\ y_k^{(1)} \\ y_k^{(2)} \\ y_k^{(1)} \\ y_k^{(2)} \end{bmatrix}, \quad \Delta U_k^P = \begin{bmatrix} \Delta u_{k-1}^{(1)} \\ \Delta u_{k-1}^{(2)} \\ \Delta u_{k-2}^{(1)} \\ \Delta u_{k-2}^{(2)} \\ \Delta u_{k-3}^{(1)} \\ \Delta u_{k-3}^{(2)} \\ \Delta u_{k-4}^{(1)} \\ \Delta u_{k-4}^{(2)} \\ \Delta u_{k-5}^{(1)} \\ \Delta u_{k-5}^{(2)} \\ \Delta u_{k-6}^{(1)} \\ \Delta u_{k-6}^{(2)} \\ \Delta u_{k-7}^{(1)} \\ \Delta u_{k-7}^{(2)} \end{bmatrix}$$

W przypadku algorytmu DMC należy zapamiętywać przeszłe wartości sterowania (a dokładnie ich zmiany). Istotne jest, aby w następnej chwili użyć wartości sterowania obliczonej w chwili obecnej. Można do tego użyć tzw. *rejestru przesuwnego* (por 7.1.2).

3.2.2 Algorytm GPC w wersji analitycznej

Podobnie jak poprzednim rozdziale, ograniczono się do przedstawienia równań opisujących analityczne prawo regulacji. Uwzględniono również wpływ zakłóceń niemierzalnych. Ze względu na duży stopień komplikacji zapisu przyjęto założenia zmierzające do maksymalnego jego uproszczenia, które nie wpływają w żaden sposób na ograniczoność stosowania.

Algorytm GPC powstał w końcu lat osiemdziesiątych i historycznie zaliczany jest do algorytmów drugiej generacji [Tat02]. Dzięki stosunkowo prostej i zarazem dającej duże możliwości postaci modelu wewnętrznego, algorytm ten zdobył dużą popularność. Pierwsze prace teoretyczne podające dowody stabilności wyprowadzono dla algorytmów wywodzących się od algorytmu GPC. Najbardziej znaczące to algorytmy CRHPC (ang. *Constrained Receding Horizon Predictive Control*) oraz SIORHC (ang. *Stabilizing Input Output Receding Horizon*

Control), które oparte były na pomyśle dodatkowych ograniczeń równościowych sygnału wyjściowego [Plam06]. Przytoczone wzory pochodzą z [Tat02].

W algorytmie GPC wykorzystujemy model obiektu w postaci dyskretnego równania różnicowego. W ogólnym przypadku obiektu wielowymiarowego o n_u wejściach i n_y wyjściach jest to model postaci :

$$A(z^{-1})Y_k^m = B(z^{-1})U_{k-1} + C(z^{-1})\frac{v_k}{\Delta} \tag{3.4}$$

gdzie A, B, C są macierzami wielomianowymi:

$$A(z^{-1}) = 1 + A_1 z^{-1} + A_2 z^{-2} + \ldots + A_{n_A} z^{-n_A}$$
$$B(z^{-1}) = B_0 + B_1 z^{-1} + B_2 z^{-2} + \ldots + B_{n_B} z^{-n_B}$$
$$C(z^{-1}) = 1 + C_1 z^{-1} + C_2 z^{-2} + \ldots + C_{n_C} z^{-n_C}$$

z^{-1} oznacza operator jednostkowego opóźnienia. W dalszej części pracy przyjęto, że $C(z^{-1}) = 1$.

Jeśli w obiekcie (ciągłym) występuje opóźnienie, to odpowiadających mu τ współczynników wielomianu B będzie równych zero. Aby nie komplikować zapisu, przyjęto również, że stopień każdego z wielomianów B jest równy n_B. Oznacza to, że rząd każdego z wielomianów jest uzupełniany do maksymalnego rzędu sygnału sterującego (w takim przypadku uzupełnione współczynniki wielomianu B będą zerami).

W końcowej części rozdziału znajduje się przykład ilustrujący opisany powyżej sposób postępowania.

Prawo regulacji algorytmu analitycznego GPC ulega znacznemu uproszczeniu o ile poczyniono założenia:

- nie jest znany model zakłócenia, ale zakłócenie jest stałe na horyzoncie predykcji *N,* tzn.

  $$d_{k+1|k}^m = d_{k+2|k}^m = \ldots = d_{k+N|k}^m = d_k^m$$

  Zakłócenia przyjmują stałą wartość, równą wartości wyznaczonej w chwili k (model zakłóceń typu DMC).

- uproszczeniu ulegają macierze wag: $\underline{\Psi} = I$ oraz $\underline{\Lambda} = \lambda I$

Macierz Ψ_p jest macierzą wag umożliwiającą różnicowanie wpływu poszczególnych składowych wektora wyjść względem siebie w chwili $k+p$, zwykle jest to macierz diagonalna. Jeśli macierze Ψ_p są różne dla różnych wartości p, to następuje też różnicowanie wpływu prognozowanych w poszczególnych chwilach wartości uchybów regulacji $Y^{zad}_{k+p|k} - Y_{k+p|k}$. Jeśli różnicowanie takie nie jest potrzebne, to dostajemy najprostszy przypadek $\Psi_p = I$, gdzie I oznacza macierz jednostkową o wymiarze $n_y \times n_y$. Rolą macierzy $\Lambda_p \geq 0$ jest różnicowanie wzajemnego wpływu poszczególnych składowych wektora przyrostów sterowania, ale ważniejsze jest określenie wagi wpływu zmienności sterowania wobec składników odpowiadających prognozowanym uchybom regulacji. W najprostszym przypadku (braku różnicowania wpływu wektorów przyrostu sterowania w zależności od horyzontu predykcji, jak i poszczególnych składowych wektora przyrostów sterowań względem siebie w tych samych chwilach) dostajemy $\Lambda_p = \lambda I$, gdzie I jest macierzą jednostkową o wymiarze $n_u \times n_u$. Skalar λ określa wagę tłumienia zmienności sterowań w stosunku do redukcji uchybów regulacji.

Analityczne prawo regulacji dla algorytmu GPC można zapisać w postaci:

$$\Delta\hat{U}_k = K \cdot \left[Y_k^{zad} - Y_k^0 \right] \text{ , gdzie} \tag{3.2}$$

$$Y_k^{zad} = \begin{bmatrix} y^{(1)zad}_{k+N_1/k} \\ \vdots \\ y^{(n)zad}_{k+N_1/k} \\ \\ \vdots \\ \\ y^{(1)zad}_{k+N/k} \\ \vdots \\ y^{(n)zad}_{k+N/k} \end{bmatrix}, \qquad Y_k^0 = \begin{bmatrix} y_k^{(0)1} \\ \vdots \\ y_k^{(0)n_y} \\ \\ \vdots \\ \\ y_k^{(0)1} \\ \vdots \\ y_k^{(0)n_y} \end{bmatrix}, \qquad K = \left[M^T M + \lambda I \right]^{-1} M^T$$

Y_k^{zad} jest wektorem wartości zadanych wyjść regulowanych. Często zakłada się, że wartości zadane są stałe na horyzoncie predykcji (jeśli nie znamy zmian ich wartości)

Y_k^0 to tzw. *trajektoria swobodna prognozy wyjść regulowanych.* Zależy ona jedynie od poprzednich wartości sterowania

Podobnie jak w przypadku algorytmu DMC, macierz K jest obliczana raz w trybie *„off – line”* .

Formuły definiujące algorytm GPC można wyprowadzić zakładając model zakłóceń typu DMC dla każdego z wyjść obiektu. Uzyskujemy w ten sposób elementy odpowiedzi skokowych i (rekurencyjnie) składowych swobodnych trajektorii wyjść. Rozważać będziemy macierz diagonalną $A(z^{-1})$

$$A = \begin{bmatrix} A_1 & 0 & 0 \\ 0 & \ddots & 0 \\ 0 & 0 & A_{max\_n} \end{bmatrix} \quad , \qquad \text{gdzie } A_1 = \begin{bmatrix} a_1^1 & \dots & a_1^{n_y} \end{bmatrix}$$

$max\_n = n_A$ - maksymalny rząd sygnału wyjściowego

a_j^i - współczynniki wielomianu składowego A_j macierzy A

oraz macierz $B(z^{-1})$

$$B = \begin{bmatrix} B_0^1 & \dots & B_{max\_m}^1 \\ \dots & \ddots & \dots \\ B_0^n & \dots & B_{max\_m}^n \end{bmatrix} \quad , \qquad \text{gdzie } B_0^1 = \begin{bmatrix} b_0^{1,1} & \dots & b_0^{1,n_u} \end{bmatrix}$$

$max\_m = n_B$ - maksymalny rząd sygnału wejściowego sterującego

definiujące model odpowiedzi m - tego wyjścia na n_u sterowań.

Wartość wyjścia w chwili k (obliczona) będzie równa:

$$y_k^m = -\sum_{i=1}^{n_A} a_i^m y_{k-i}^m + \sum_{j=1}^{n_u}\sum_{i=0}^{n_B} b_i^{m,j} u_{k-1-i}^j \; , \qquad m = 1, \dots, n_y$$

Z zależności powyższej dostajemy wzór na elementy odpowiedzi skokowej, dla każdej pary wejście (j - te) – wyjście (m - te)

$$s_k^{m,j} = \sum_{i=1}^{min\{k-1, n_A\}} a_i^m s_{k-i}^{m,j} + \sum_{i=0}^{min\{k-1, n_B\}} b_i^{m,j} \; , \qquad m = 1, \dots, n_y$$
$$j = 1, \dots n_u$$

Znając elementy wielowymiarowej odpowiedzi skokowej możemy sformułować postać macierzy dynamicznej określającej *trajektorie wymuszone* wyjść (zależne od przyrostów sterowania w chwilach bieżących i przyszłych) na horyzoncie predykcji :

$$M = \begin{bmatrix} S_{N_1} & S_{N_1-1} & \dots & S_1 & \dots & 0 \\ S_{N_1+1} & S_{N_1} & \dots & 0 & \dots & 0 \\ \dots & \dots & \dots & \dots & \ddots & \\ S_N & S_{N-1} & \dots & S_{N-N_1} & \dots & S_{N-N_u+1} \end{bmatrix}$$

Dla wyznaczenia elementów trajektorii swobodnej przyjmujemy na horyzoncie predykcji sterowanie stałe, równe wartości wyznaczonej w chwili poprzedniej :

$$u^m_{k+p|k} = u^m_{k-1} \quad , \quad p = 0,1,\dots,N-1 \qquad m = 1,\dots,n_y$$

Przy takich sterowaniach wyznaczamy $y^{(0)\,m}_{k+p|k}$ kolejno, dla $p = 1,\dots,N$, ze wzoru

$$y^{(0)\,m}_{k+p|k} = y^m_{k+p|k} + d^m_k$$

gdzie wartości $y^m_{k+p|k}$ liczymy z modelu 3.1 . Prognozowane na podstawie modelu 3.1 wyjścia swobodne są uzupełnione estymatą zakłócenia d^m_k w chwili k, stałą na horyzoncie i równą wartości obliczonej w chwili bieżącej k (zgodnie z przyjętym modelem zakłócenia typu DMC)

$$d^m_k = y^m_k - y^m_{k|k-1} = y^m_k - \left[-\sum_{i=1}^{n_A} a^m_i y^m_{k-i} + \sum_{i=1}^{n_u}\sum_{i=0}^{n_B} b^{m,j}_i u^j_{k-1-i} \right]$$
$$m = 1,\dots,n_y$$
$$p = 1,\dots,N$$

Przedstawiony sposób postępowania prowadzi do następującego wzoru na elementy trajektorii swobodnej wyjść przewidywanych na horyzoncie predykcji:

$$y^{(0)\,m}_{k+p\backslash k} = -\sum_{i=1}^{\min\{n_A,p-1\}} a^m_i y^{(0)\,m}_{k+p-i|k} - \sum_{i=\min\{n_A,p-1\}+1}^{n_A} a^m_i y^m_{k+p-i} + \\ -\sum_{j=1}^{n_u}\left[\sum_{i=0}^{\min\{n_B,p\}} b^{m,j}_i u^j_{k-1} + \sum_{i=\min\{n_B,p\}+1}^{n_B} b^{m,j}_i u^j_{k-1+p-i} \right] + d_m(k) \quad ,$$

gdzie

$$d^m_k = y^m_k - \left[-\sum_{i=1}^{n_A} a^m_i y^m_{k-i} + \sum_{i=1}^{n_u}\sum_{i=0}^{n_B} b^{m,j}_i u^j_{k-1-i} \right]$$
$$m = 1,\dots,n_y$$
$$p = 1,\dots,N$$

W ten sposób udało się uzyskać wzór na odpowiedź swobodną, a tym samym ustalono wartości czynników z równania 3.2

Przedstawiona powyżej sytuacja nie uwzględnia ograniczeń sygnałów sterujących. Jak pokazuje praktyka przemysłowa, ograniczenia wielkości sygnałów sterujących są często bardzo istotne, gdyż wygenerowane przez regulatory liniowe sterowania są nierealizowalne fizycznie – o zbyt dużych amplitudach czy też szybkościach narastania. Ignorowanie w pracy regulatora faktu aktywności ograniczeń sygnałów sterujących może prowadzić do dramatycznego pogorszenia jakości regulacji [Tat02].
Ograniczenia na wartość sygnałów sterujących w analitycznym regulatorze GPC można łatwo wprowadzić: wystarczy macierz zmian sterowania odpowiednio zmodyfikować:

ograniczenia na przyrost sterowań :

$$\begin{aligned} &if \quad \Delta u^m < \Delta u^m_{\min} \quad then \\ &\qquad \Delta u^m = \Delta u^m_{\min} \qquad \text{oraz} \\ &end \end{aligned}$$

$$\begin{aligned} &if \quad \Delta u^m > \Delta u^m_{\max} \quad then \\ &\qquad \Delta u^m = \Delta u^m_{\max} \\ &end \end{aligned}$$

ograniczenia na wartość sterowań:

$$\begin{aligned} &if \quad u^m_{k-1} + \Delta u^m < \Delta u^m_{\min} \quad then \\ &\qquad \Delta u^m = u^m_{\min} - u^m_{k-1} \qquad \text{oraz} \\ &end \end{aligned}$$

$$\begin{aligned} &if \quad u^m_{k-1} + \Delta u^m < \Delta u^m_{\max} \quad then \\ &\qquad \Delta u^m = u^m_{\max} - u^m_{k-1} \qquad , \\ &end \end{aligned}$$

gdzie:

Δu^m - wektor zmian sterowania

$\Delta u^m_{\min}$ - najmniejsze dopuszczalne zmiany sygnału sterującego

$\Delta u^m_{\max}$ - największe dopuszczalne zmiany sygnału sterującego

u^m_{k-1} - wartość sygnału sterującego z poprzedniej chwili

Podsumowanie:

Reasumując, schemat implementacji algorytmu może wyglądać następująco:

1. Przed uruchomieniem regulacji obliczyć macierz $K = \left[M^T M + \lambda I\right]^{-1} M^T$, gdzie

$$M = \begin{bmatrix} S_{N_1} & S_{N_1-1} & \cdots & S_1 & \cdots & 0 \\ S_{N_1+1} & S_{N_1} & \cdots & 0 & \cdots & 0 \\ \cdots & \cdots & \cdots & \cdots & \ddots & \\ S_N & S_{N-1} & \cdots & S_{N-N_1} & \cdots & S_{N-N_u+1} \end{bmatrix}$$

Elementy odpowiedzi skokowej obliczyć ze wzoru:

$$s_k^{m,j} = \sum_{i=1}^{min\{k-1,n_A\}} a_i^m s_{k-i}^{m,j} + \sum_{i=0}^{min\{k-1,n_B\}} b_i^{m,j} \quad , \qquad m = 1,\ldots,n_y \quad j = 1,\ldots n_u$$

2. W każdym kroku algorytmu wyznaczyć wartość trajektorii swobodnej :

$$y_{k+p\backslash k}^{(0)m} = -\sum_{i=1}^{\min\{n_A,p-1\}} a_i^m y_{k+p-i|k}^{(0)m} - \sum_{i=\min\{n_A,p-1\}+1}^{n_A} a_i^m y_{k+p-i}^m + -\sum_{j=1}^{n_u}\left[\sum_{i=0}^{\min\{n_B,p\}} b_i^{m,j} u_{k-1}^j + \sum_{i=\min\{n_B,p\}+1}^{n_B} b_i^{m,j} u_{k-1+p-i}^j\right] + d_m(k) \quad ,$$

gdzie

$$d_k^m = y_k^m - \left[-\sum_{i=1}^{n_A} a_i^m y_{k-i}^m + \sum_{i=1}^{n_u}\sum_{i=0}^{n_B} b_i^{m,j} u_{k-1-i}^j\right]$$

$$m = 1,\ldots,n_y$$
$$p = 1,\ldots,N$$

3. Obliczoną w poprzednim kroku wartość trajektorii swobodnej podstawić do wzoru :

$$\Delta\hat{U}_k = K \cdot \left[Y_k^{zad} - Y_k^0\right]$$

Przykład 3.2

Dla zilustrowania wielowymiarowego algorytmu GPC w wersji analitycznej posłużono się dwuwymiarowym modelem liniowym o podanej transmitancji :

$$H(s) = \begin{bmatrix} \dfrac{K_{11} \cdot e^{-2s}}{1 + T_{11} \cdot s} & \dfrac{K_{12}}{1 + T_{12} \cdot s} \\ \dfrac{K_{21}}{1 + T_{21} \cdot s} & \dfrac{K_{22}}{1 + T_{22} \cdot s} \end{bmatrix}$$

Przykład jest szczegółowo omawiany w [Tat02] , uzupełniono go natomiast o przypadek występowania opóźnienia w obiekcie.

Wybrano następujące parametry regulatora :

- horyzont predykcji : $N = 5$
- horyzont sterowania : $N_u = 4$
- ponadto założono, że $\Psi = I$ oraz $\Lambda = \lambda \cdot I$

W obiekcie występuje opóźnienie pomiędzy pierwszym sygnałem sterującym a wyjściem regulowanym. Z tego faktu wynika równość $N_1 = 1$.
Po dyskretyzacji i sprowadzeniu elementów każdego wiersza macierzy (obiektu) do wspólnego mianownika otrzymano model w postaci:

$$A(z^{-1}) Y_k^m = B(z^{-1}) U_{k-1} \quad , \quad \text{gdzie}$$

$$A(z^{-1}) = \begin{bmatrix} 1 + a_1^1 z^{-1} + a_2^1 z^{-2} & 0 \\ 0 & 1 + a_1^2 z^{-1} + a_2^2 z^{-2} \end{bmatrix}$$

Transmitancja opisująca każdą parę wejście – wyjście jest w postaci jednej inercji, więc rząd każdego z wielomianów składowych wielomianu A będzie taki sam i równy $n_A = 2$.

Po uzupełnieniu rzędu każdego z wielomianów do maksymalnego rzędu sygnału sterującego ($n_B = 3$) otrzymano :

$$B(z^{-1}) = \begin{bmatrix} 0 + 0 \cdot z^{-1} + b_2^{1,1} z^{-2} + b_3^{1,1} z^{-3} & b_0^{1,2} + b_1^{1,2} z^{-1} + 0 \cdot z^{-2} + 0 \cdot z^{-3} \\ b_0^{2,1} + b_1^{2,1} z^{-1} + 0 \cdot z^{-2} + 0 \cdot z^{-3} & b_0^{2,2} + b_1^{2,2} z^{-1} + 0 \cdot z^{-2} + 0 \cdot z^{-3} \end{bmatrix}$$

Elementy odpowiedzi skokowej, dla każdej pary wejście (*j* - te) – wyjście (*m* - te) wyrażają się następująco :

$$s_k^{m,j} = \sum_{i=1}^{min\{k-1,n_A\}} a_i^m s_{k-i}^{m,j} + \sum_{i=0}^{min\{k-1,n_B\}} b_i^{m,j} \quad , m = 1,\ldots,n_y$$

Stąd macierz M :

$$M = \begin{bmatrix} S_3 & S_2 & S_1 \\ S_4 & S_3 & S_2 \\ S_5 & S_4 & S_3 \end{bmatrix}$$

Mając macierz M można obliczyć macierz $K = \left[M^T M + \lambda I\right]^{-1} M^T$.

Obliczenie macierzy dynamicznej oraz macierzy K kończy etap przygotowań w fazie *„off – line”*. Obliczana w każdym kroku trajektoria swobodna będzie wynosić :

$$Y_k^0 = \left[y_{k+3}^{(0)1} \quad y_{k+3}^{(0)2} \quad y_{k+4}^{(0)1} \quad y_{k+4}^{(0)2} \quad y_{k+5}^{(0)1} \quad y_{k+5}^{(0)2}\right]^T.$$

Jej elementy opisuje wzór:

$$y_{k+p\backslash k}^{(0)m} = \sum_{i=1}^{\min\{n_A,p-1\}} a_i^m y_{k+p-i|k}^{(0)m} - \sum_{i=\min\{n_A,p-1\}+1}^{n_A} a_i^m y_{k+p-i}^m + \\ - \sum_{j=1}^{n_u}\left[\sum_{i=0}^{\min\{n_B,p\}} b_i^{m,j} u_{k-1}^j + \sum_{i=\min\{n_B,p\}}^{n_B} b_i^{m,j} u_{k-1+p-i}^j\right] + d_k^m$$

gdzie zakłócenie jest obliczane w sposób następujący :

$$d_k^m = y_k^m - \left[-\sum_{i=1}^{n_A} a_i^m y_{k-i}^m + \sum_{i=1}^{n_u}\sum_{i=0}^{n_B} b_i^{m,j} u_{k-1-i}^j\right],$$

$$m = 1,\ldots,n_y$$
$$p = 1,\ldots,N$$

Jeśli nie znamy wartości zadanej dla poszczególnych wyjść w chwili k, to zakładamy, że są one równe wartości na chwilę bieżącą, czyli :

$$Y_k^{zad} = \left[y_{k+3}^{(1)\,zad} \quad y_{k+3}^{(2)\,zad} \quad y_{k+4}^{(1)\,zad} \quad y_{k+4}^{(2)\,zad} \quad y_{k+5}^{(1)\,zad} \quad y_{k+5}^{(2)\,zad} \right]^T$$

W ten sposób, wykorzystując obliczone macierze można wyznaczyć macierz zmian wartości sterowania:

$$\Delta \hat{U}_k = K \cdot \left[Y_k^{zad} - Y_k^0 \right]$$

W chwili bieżącej k wykorzystujemy jedynie pierwsze n_u wartości tak obliczonej macierzy.

Jak już wspomniano obliczenia wykonywane w trybie *„off – line”* ograniczają się do obliczenia macierzy K, według wzoru 3.3. W warunkach implementacji algorytmu w rzeczywistym urządzeniu (np. regulatorze), obliczeń tych dokonuje się w na maszynie (komputerze) wyposażonej w szybkie i dokładne programy obliczeniowe, a następnie przesyła do urządzenia. Taki sposób postępowania wydaję się zasadny, bowiem regulator nie musi wykonywać skomplikowanych obliczeń, które są wykorzystywane w zasadzie tylko do obliczenia pomocniczej macierzy K, natomiast ich pożytek w przypadku innych funkcji realizowanych przez regulator jest znikome. Zaoszczędza się w ten sposób zasoby urządzenia, głównie jeśli chodzi o pamięć. Macierz K jest specyficzna dla regulowanego obiektu, więc przesłanie takie może nastąpić przez osobę odpowiedzialną za wdrożenie instalacji. Istnieje również możliwość zapisania takiej macierzy w trakcie produkcji regulatora o ile użytkownik prześle producentowi urządzenia dane umożliwiające identyfikację obiektu.
W kolejnym rozdziale skupiono się na budowie wykorzystywanego regulatora jak również na sposobie przesyłu danych pomiędzy komputerem a urządzeniem.

Rozdział 4

Opis regulatora LB – 600

Poprzedni rozdział zawierał opis algorytmów oraz wzory reprezentujące prawo regulacji. Aby dokonać analizy niezbędnych do implementacji algorytmu predykcyjnego zasobów urządzenia należy poznać jego budowę, zarówno fizyczną jak i funkcjonalną. Niniejszy rozdział zawiera podstawowe informacje o budowie docelowego urządzenia implementacji (regulatora), mające pomóc w analizie zasobów wykorzystywanych przez algorytm GPC oraz uzyskania odpowiedzi na pytanie, kiedy taka implementacja w regulatorze będzie możliwa. Zaprezentowane poniżej fragmenty opisu urządzenia pochodzą z oficjalnej strony firmy LAB – EL: www.label.com.pl.

4.1 Budowa fizyczna urządzenia

Uniwersalny regulator mikroprocesorowy LB-600 został skonstruowany z uwzględnieniem wszystkich wymagań stawianych podobnym urządzeniom stosowanym w automatyce przemysłowej. Łatwość montażu umożliwia konfigurację przyrządu poprzez odpowiednią kombinację pakietów wejściowo-wyjściowych oraz załadowanie z zewnątrz oprogramowania (bez konieczności rozbierania przyrządu w celu wymiany pamięci EPROM) odpowiedniego do realizowanych funkcji. Regulator LB-600 posiada klasyczne i specjalizowane algorytmy oraz funkcje sterowania (np. PID, PID fuzzy, samostrojenie). Ważną cechą, z punktu widzenia celu pracy, jest zastosowanie reprogramowalnej, nieulotnej pamięci typu FLASH MEMORY (jako pamięci danych, programu i użytkownika) oraz niezależnego procesora do obsługi wyświetlacza.[1]

[1] Więcej na temat budowy i funkcji specjalizowanych urządzenia w [KSŁ99]

Rys. 4.1 Regulator LB – 600

Aparat ma konstrukcję modułową :

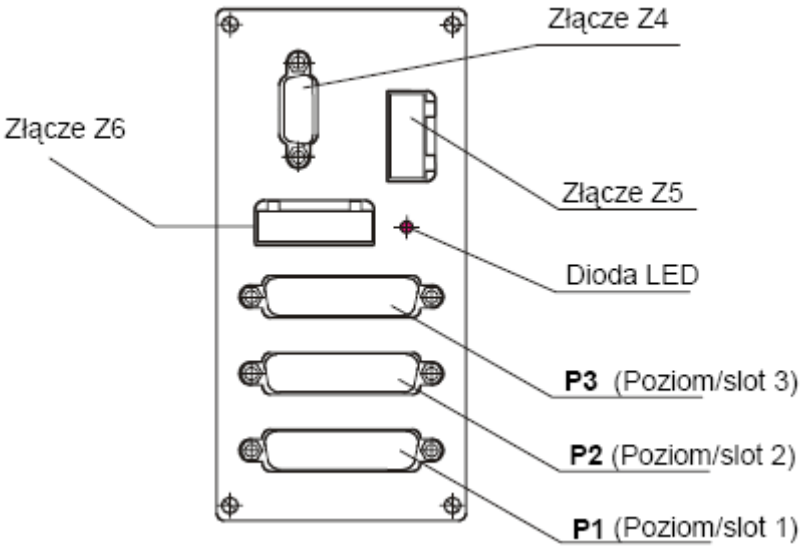
- MODUŁ OPERATORSKI wyposażony w panel przedni, 12 stykowe gniazdo do połączenia z pakietem procesora oraz 7 stykowe złącze grzebieniowe do podłączenia foliowej klawiatury. Moduł operatorski obsługiwany jest przez niezależny procesor AT90S2313, co zwalnia procesor główny od obsługi wyświetlania i klawiatury

- MODUŁ PROCESORA zawierający jednoukładowy specjalizowany mikrokontroler AT89S8252 firmy ATMEL oraz układy pamięciowe MB84256 (RAM 32Kb0 i AT29C010 (FLASH 128 kB). Moduł procesora wykonany w postaci jednej płytki drukowanej wyposażonej w złącza do połączenia z modułem operatorskim, modułem zasilacza i sterownika transmisji szeregowej, pakietami wejść i wyjść analogowych oraz pakietem wejść / wyjść dyskretnych. W module procesora następuje przetwarzanie sygnałów analogowych na cyfrowe, zapamiętywanie zmiennych procesowych, struktur sterowania, parametrów sterowania i regulacji, itp.

- PAKIET WEJŚĆ ANALOGOWYCH może w zależności od konfiguracji obsłużyć do 10 wejść analogowych (unipolarnych ze wspólnym punktem masy sygnałowej), a możliwość umieszczenia do trzech takich pakietów w przyrządzie daje obsługę pomiarową do 30 wejść.. Pakiet wykonany jest w postaci płytki drukowanej wyposażonej w złącza do połączenia z pakietem procesora oraz złącze do połączenia przewodów z przetworników pomiarowych.

- PAKIET WYJŚĆ ANALOGOWYCH, jeden dla obu wyjść, służy do uzyskania dwu w pełni niezależnych, odseparowanych galwanicznie sygnałów wyjściowych napięciowych lub/i prądowych regulatora. Pakiet wykonany jest w postaci płytki drukowanej wyposażonej w złącze do połączenia z pakietem procesora oraz złącze do połączenia przewodów wyprowadzających sygnały wyjściowe analogowe prądowe.

- PAKIET WEJŚĆ / WYJŚĆ DYSKRETNYCH jest jeden dla 8 wejść i 6 wyjść.
  Pakiet jest wyposażony w złącze do połączenia z pakietem procesora oraz złącze do połączenia kablowego z nadajnikami i odbiornikami sygnałów dyskretnych.



Rys. 4.2 Rozmieszczenie przyłączy

► Złącze Z4 – transmisja szeregowa RS 485, RS232 i 4 wejścia sygnałów cyfrowej pętli prądowej S300, jest to złącze typu CANNON 15R3 (15 pinów w trzech rzędach)

► Złącze Z5 – zasilanie regulatora – dla wykonania regulatora na napięcie zasilające 24Vdc, złącze ma inny typ,
► Złącze Z6 – złącze do zasilania przetworników pomiarowych napięcie 24Vdc, prąd obciążenia do 500mA (zabezpieczenie przeciążeniowe),
► P3 – złącze pakietu wejściowo-wyjściowego umiejscowionego w pozycji górnej – poziom/slot 3,
► P2 – złącze pakietu wejściowo-wyjściowego umiejscowionego w pozycji środkowej – poziom/slot 2,
► P1 – złącze pakietu wejściowo-wyjściowego umiejscowionego w pozycji dolnej – poziom/slot 1,
► Dioda LED – sygnalizuje obecność napięcia zasilającego

Poprzez odpowiednią kombinację pakietów wejściowo – wyjściowych (w poszczególnych "slotach" regulatora) możliwe jest przekształcenie regulatora w przyrząd pomiarowy, sterownik realizujący funkcje czasowe, itp.

4.2 Struktura funkcjonalna regulatora

Regulator stanowi zbiór swobodnie programowalnych bloków funkcjonalnych zwanych dalej *funktorami*. Każdy z funktorów jest elementem posiadającym wiele wejść oraz tylko jedno wyjście. Wartości wejść jak i wyjść są podawane w skali znormalizowanej: 0…1. Niektóre funktory mogą generować binarne sygnały alarmowe **AL** i **AH.** Funktory mogą realizować różne funkcje w zależności od potrzeby wynikającej ze specyfiki sterowanego procesu. Stanowią one swego rodzaju tablicę krosową, gdzie funktory można łączyć i w ten sposób określać sekwencję operacji. Identyfikacja lokalizacji funktora polega na podaniu jego "adresu", czyli numeru warstwy oraz numeru toru (kanału).

Poszczególne warstwy związane są z obsługą pewnych funkcji regulatora; i tak:

- warstwa 0 – parametry generalne (hasła, adresy, zegary, alarmy czasowe, itp.)
- warstwa 1 – obsługa wejść analogowych
- warstwa 2 – obsługa wejść binarnych
- RSB (Rejestr Stanów Binarnych) – między innymi generowanie alarmów
- warstwy 3, 4, 5, 6 – obsługa funkcji arytmetycznych, logicznych i czasowych wielu zmiennych wejściowych
- warstwa 7 – obsługa właściwej funkcji regulatora (tryby i algorytmy regulacji)
- warstwa 8 – dodatkowy zbiór funktorów arytmetycznych dla sygnałów analogowych
- warstwa 9 – obsługa wyjść analogowych
- warstwa A – obsługa wyjściowych funktorów binarnych
- warstwa b – obsługa skalowania wejść i wyjść analogowych

Oznaczenie **31** określa funktor warstwy 3 z kanału (toru) 1 (*Rys. 4.31*)

Rys. 4.3 Przykładowe oznaczenie funktora

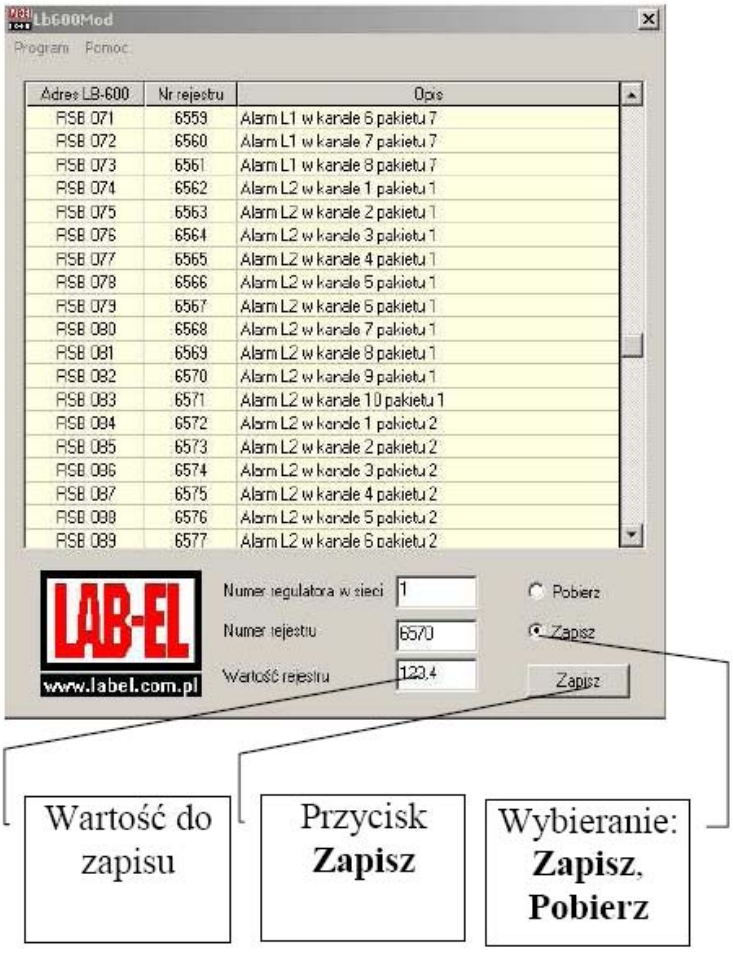
Każdy parametr funktora można określić przez podanie numeru warstwy i kanału, z którego pochodzi dany funktor, oraz numeru tej zmiennej, np. współczynnik K_1 z funktora **31** ma adres **3114**, a współczynnik K_1 funktora **32** ma adres **3214**. Każdy adres ma przyporządkowany numer rejestru w pamięci regulatora. Spis wszystkich parametrów poszczególnych funktorów oraz odpowiadające im wartości adresów i rejestrów można odczytać z dokumentacji technicznej regulatora LB – 600.
Dla przykładu adres współczynnika K_1 z warstwy 1 wynosi **3114** a jego numer rejestru **2913**.

Przy używaniu zmiennych zapisanymi w regulatorze wygodniej jest posługiwać się adresem zmiennej, a nie jej numerem rejestru. Umożliwia to natychmiastowe rozpoznanie do jakiej warstwy i kanału należy dana zmienna. Jest to o tyle istotne, że jak już wspomniano każda warstwa realizuje inne funkcje. Również w przypadku zaprogramowana regulatora następuje odwołanie się do adresów (a nie bezpośrednio do rejestrów).

4.3 Przegląd oprogramowania użytkowego

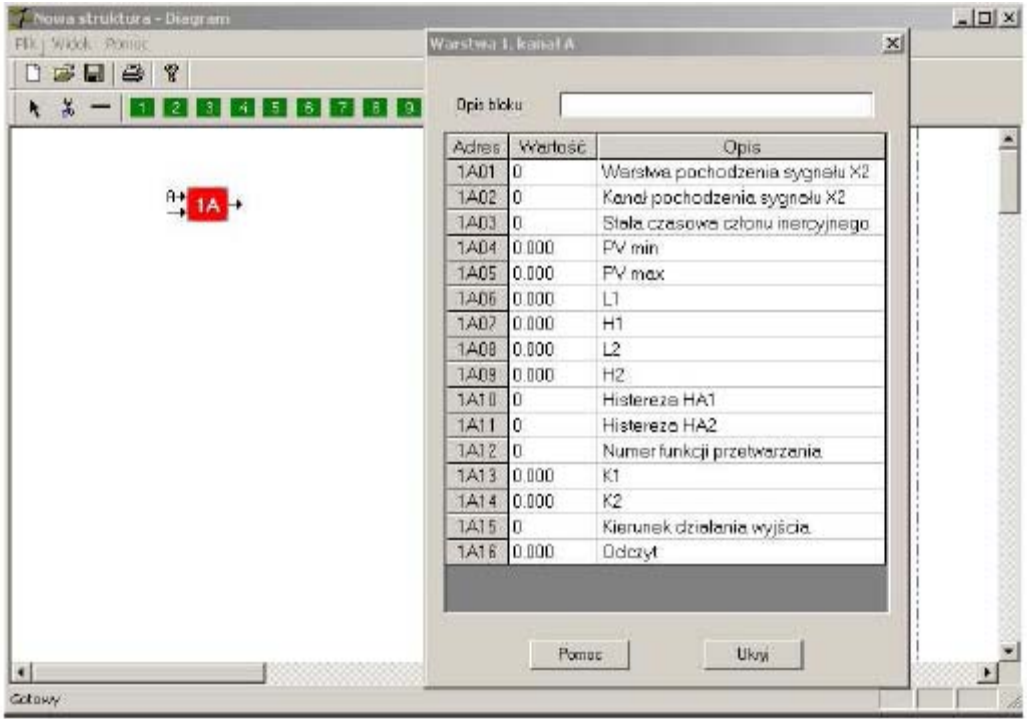
W poprzednich rozdziałach padł zwrot: „zaprogramowanie regulatora". Polega ono na wpisaniu odpowiednich wartości parametrów do adresów regulatora, zgodnie z funkcjami jakie ma on spełniać. Można tego dokonać ręcznie lub przez komputer podłączony do urządzenia. W tym drugim przypadku potrzebne jest specjalizowane oprogramowanie.
Oprogramowanie napisane przez firmę LAB – EL umożliwia współpracę regulatorów LB-600 z systemami komputerowymi. Pozwala ono m.in. na przesyłanie danych do/z regulatora, zaprogramowanie/pobranie struktury regulacji. Najważniejsze programy wykorzystywane do zapisu (i odczytu) zmiennych to:

- *LB600mod* - program umożliwiający zapis i odczyt pojedynczych parametrów z/do rejestrów regulatora. Mogą to być zmienne procesowe jak i zmienne konfiguracyjne



Rys. 4.4 Okno robocze programu Lb600Mod

- *Diagram* - program umożliwiający tworzenie struktur w środowisku graficznym tzn. struktury tworzone są poprzez pobieranie z biblioteki funktorów odpowiednich „bloczków funkcjonalnych", łączenie ich pomiędzy sobą oraz parametryzowanie poprzez zapis odpowiednich wartości w tablicach konfiguracyjnych poszczególnych bloków.
  Korzystając z niego można zapisać i odczytać sposób połączenia funktorów oraz wartości zmiennych regulacji.



Rys. 4.5 Okno pola roboczego tworzonej struktury wraz z tablicą dotyczącą funktora 1A

Kolejny program spełnia już nieco inną funkcję :

- *Regulator* - program łączy w sobie funkcje:

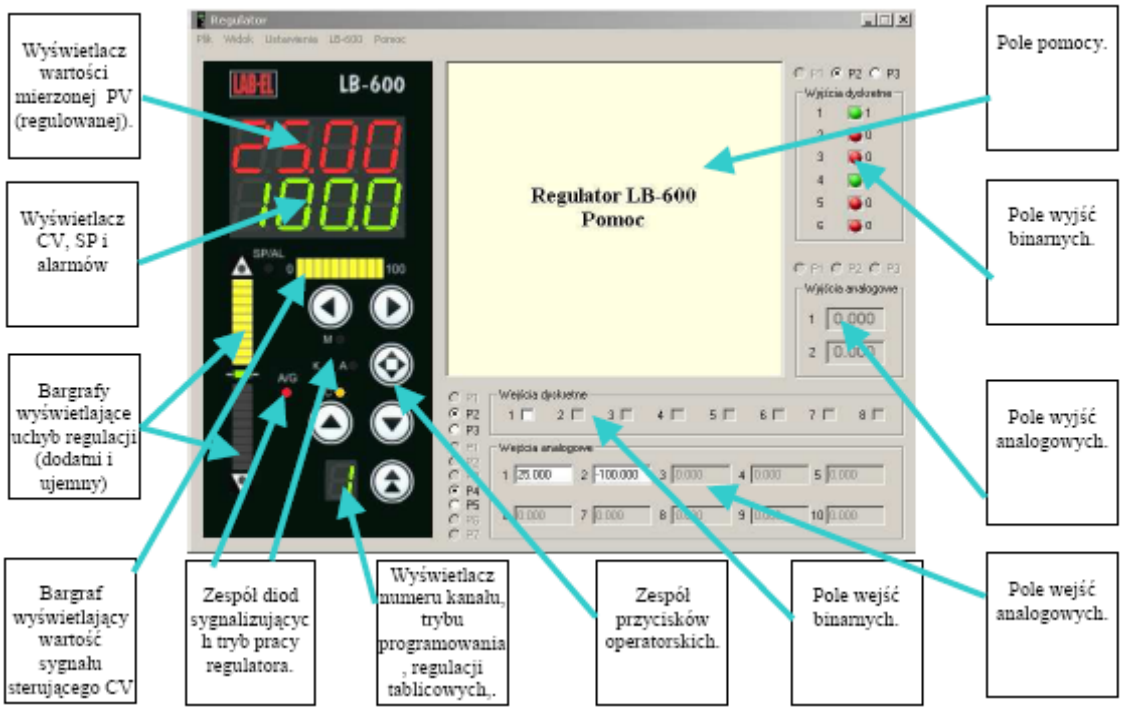
  → symulatora – zaprogramowany symulator realizuje rzeczywiste algorytmy i generuje sygnały, które można obserwować na wyjściach

  → konfiguratora – sprawdzoną i skorygowaną strukturę sterowania wpisuje się do regulatora poprzez przesłanie jej łączem transmisyjnym.

Ma on o tyle istotne znaczenie, że realizuje on rzeczywiste algorytmy zaimplementowane w urządzeniu fizycznym. Pozwala na :

→ programowanie struktury regulacji i/lub sterowania,
→ symulację parametrów wejściowych analogowych i/lub binarnych,
→ odczyt symulacyjny sygnałów wyjściowych analogowych i/lub binarnych,
→ przesyłanie całej struktury do zaadresowanego i podłączonego regulatora,
→ zapis całej struktury w postaci zbiorów: binarnego i tekstowego,
→ graficzne tworzenie struktury z bloków funkcjonalnych,
→ wydruk struktury w postaci „formatki" projektowej.

Programowanie struktury przez użytkownika wymaga znajomości funkcji operatorskich, które są identyczne jak w rzeczywistym regulatorze, z tą różnicą, że w regulatorze używa się do niektórych funkcji dwóch przycisków, natomiast w programie *Regulator* używa się klawisza SHIFT (klawiatura komputerowa) oraz myszki, którą „klika się" na rysunku odpowiedniego przycisku.



Rys. 4.6 Okno programu Regulator z wczytaną przykładową strukturą

Regulator LB-600 został wyposażony w kilka rodzajów interfejsów umożliwiających jego pracę w systemach komputerowych. Najważniejszym z nich to interfejs RS – 232. Umożliwia on zapis do pamięci Flash programu obsługi wyświetlacza i klawiatury oraz głównego programu systemowego (regulator wykorzystuje dwa niezależne procesory: jeden do obsługi operatorskiej, drugi do obsługi funkcjonalnej).

Dokładny opis uruchomienia i skonfigurowania opisywanych programów można znaleźć w [SzkA05].

4.4 Identyfikacja obiektu

Regulator został wyposażony w funkcję identyfikacji obiektu. Identyfikacja obiektu w regulatorze LB-600 jest nierozerwalnie związana z funkcją samostrojenia (automatyczny dobór nastaw PID w punkcie pracy), dlatego poznanie zasady działania samostrojenia ma kluczowe znaczenie, jeśli chcemy otrzymać poprawne parametry identyfikowanego obiektu.

Nastawy PID obliczane są automatycznie wg parametrów eksperymentu identyfikacyjnego. Operacja samostrojenia wykorzystuje jeden z trzech algorytmów :

- metodę Cohena-Coona,
- syntezy nastaw według jednego z trzech kryteriów jakości regulacji: 5% przeregulowania, 20% przeregulowania, minimum całki z kwadratu uchybu regulacji,
- metody Zieglera-Nicholsa,

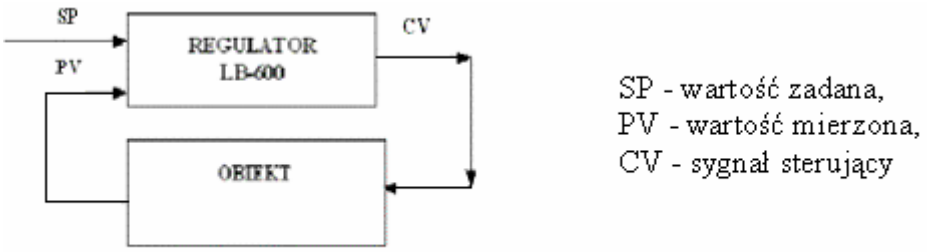
Dobór nastaw regulatora zależy od:

- własności statycznych i dynamicznych układu regulacji,
- wymagań stawianych przed układem regulacji,
- wybranego typu regulatora.

Na podstawie parametrów identyfikacyjnych oraz wyboru odpowiedniego kryterium jakościowego użytkownik współdecyduje o nastawach PID, jakie zostaną wprowadzone do struktury regulacyjnej.
Samostrojenie przeznaczone jest przede wszystkim dla obiektów statycznych, dla których spełniony jest warunek:

$$0{,}1 \le \frac{Opóźnienie\ obiektu}{Zastępcza\ stala\ czasowa} \le 0{,}6$$

Stosunek opóźnienia obiektu do jego zastępczej stałej czasowej mieści się w granicach od 0,1 do 0,6. Samostrojenia nie należy stosować dla obiektów o stałych czasowych mniejszych od kilkunastu sekund. Samostrojenie przeprowadza się w układzie zamkniętym tak jak na rys. 4.7



Rys 4.7 Schemat blokowy podstawowego układu regulacji

Eksperyment samostrojenia polega na wprowadzeniu do sygnału sterującego (CV) okresowego zaburzenia o określonej amplitudzie. W trakcie eksperymentu sygnał sterujący będzie generowany symetrycznie wokół punktu pracy. Na podstawie zmian w sygnale regulowanym (PV), wywołanych zaburzeniem, algorytm samostrojenia ustala nowe nastawy.

Warunkiem przeprowadzenia eksperymentu samostrojenia jest doprowadzenie obiektu do stanu równowagi wokół punktu pracy, tzn. różnica PV - SP musi być mniejsza od ustalonej wartości. Doprowadzenie do stanu równowagi może być dokonane automatycznie (przy pomocy dotychczasowych nastaw) lub ręcznie.

Samostrojenie składa się z trzech etapów:

- doprowadzenie do stanu równowagi,
- pomiaru poziomu zakłóceń,
- właściwego eksperymentu.

Czas trwania pierwszego etapu zależy od szybkości osiągnięcia stanu równowagi. Czas trwania drugiego etapu jest stały i wynosi 2 minuty. Czas trwania trzeciego etapu, czyli właściwego eksperymentu zależy od dynamiki obiektu i nie przekracza wartości $6 \cdot T_Z$, gdzie: T_Z - zastępcza stała czasowa obiektu.

W przypadku stwierdzenia przez procedurę niemożliwości dobrania nastaw, algorytm się wyłącza i sygnalizuje odpowiednim kodem błędu. Eksperyment samostrojenia można wyłączyć w dowolnym momencie. W przypadku pomyślnego zakończenia eksperymentu na wyświetlaczu można podejrzeć nowe nastawy.

Jeśli eksperyment zakończy się niepowodzeniem, pojawi się napis *Err*, a zamiast wartości nowych nastaw będą wyświetlone numery błędów.

Po udanym eksperymencie identyfikacyjnym uzyskuje się nie nastawy regulatora, lecz parametry identyfikacyjne regulowanego obiektu (procesu):

- T - zastępcza stała czasowa obiektu
- k_0 - wzmocnienie obiektu
- T_0 - czas opóźnienia obiektu

W ten sposób obiekt opisany równaniem transmitancji w postaci dwóch stałych czasowych zostaje opisany jedną stałą czasową wynikającą z przeprowadzonego eksperymentu samostrojenia

Otrzymane w eksperymencie parametry identyfikacyjne obiektu regulacji są istotne z uwagi na możliwość opisu dynamiki obiektu i realizacji innych, specjalistycznych algorytmów adaptacyjnych.

Rozdział 5

Analiza potrzebnych zasobów

W poprzednim podrozdziale wskazano na główny problem implementacji zaawansowanych algorytmów, jakim jest brak zasobów, głównie pamięci. Innym ograniczeniem jest stopień komplikacji obliczeń, i co za tym idzie zajętość i czas obliczeń procesora.

W niniejszym rozdziale przedstawiona zostanie analiza zasobów dla algorytmu GPC. Analizowana będzie wersja analityczna algorytmu.

Realizacja algorytmu regulatora predykcyjnego na komputerze wymaga trzech rodzajów pamięci danych, ich szczegółowy opis przedstawiono poniżej :

- Pamięć na stałe struktury danych (*SSD*) – pamięć w której przechowywane są parametry modelu (współczynniki, wartości opóźnień, ilości wejść, wyjść, rzędy) i, np. macierze dynamiki. Cechą charakterystyczną pamięci *SSD* to niezmienność wartości.

- Pamięć na historie sygnałów wejściowych i wyjściowych algorytmu (*I/O D*) – pamięć, w której przechowywane są historyczne wartości wejść i wyjść. Zawartość zmieniana jest w każdym kroku (nowa próbka zastępuje starą).

- Pamięć na obliczenia (stos) – pamięć wykorzystywana do obliczeń takich jak wyznaczanie trajektorii swobodnej. Zawartość pamięć nie musi być przechowywana z kroku na krok.

Podobnie jak w przypadku zapotrzebowania na pamięć, tak też w przypadku nakładu obliczeń wszystko silnie zależy od parametrów algorytmu (opóźnień, horyzontów predykcji, horyzontów sterowań, rzędów).

Poniżej przeprowadzono analizę zajętości pamięci dla obiektu dwuwymiarowego (dwa wejścia sterujące i dwa wyjścia regulowane). Założono, że model obiektu jest w postaci jednej inercji dla każdej pary wejście – wyjście. Dodatkowo została przeprowadzona analiza dla modelu dwuinercyjnego (dla każdego z torów sterowania).

Liczbę zmiennych można obliczyć dwojako:

1. upraszczając algorytm, zwiększając tym samym liczbę użytych zmiennych
2. komplikując algorytm, usuwając zbędne ilości zmiennych

Podejście według takich kryteriów jest wygodne do analizy jeśli zauważymy, że zasobami krytycznymi w regulatorze jest ilość pamięci i szybkość obliczeń. Można zmniejszyć liczbę zmiennych kosztem komplikacji algorytmu i zwiększenia czasu obliczeń. W przypadku kiedy ilość pamięci jest niewystarczająca na implementację algorytmu, konstruktor może spróbować

skomplikować algorytm, usuwając zbędne zmienne i w ten sposób ocenić czy osiągnął zadowalający rezultat. Jeśli natomiast czas obliczeń pojedynczego kroku algorytmu jest za długi, można próbować uprościć algorytm kosztem zwiększenia liczby zmiennych. Jeśli w żadnym z opisanych przypadków nie uda się spełnić ograniczeń na oba zasoby, zasadnym wydaje się przeanalizowanie budowy regulatora pod kątem usunięcia pewnej funkcjonalności kosztem zwiększenia wymaganej pamięci i zmniejszenia szybkości obliczeń pojedynczego kroku algorytmu (por rozdz. 6). Taki sposób podejścia do problemu implikuje konieczność współpracy konstruktora (serwisanta) z użytkownikiem w celu osiągnięcia znacznych oszczędności w ilości wykorzystanych zmiennych.

5.1 Wstępne obliczenie ilości zmiennych

Na początek przeprowadzono analizę dla przypadku uproszczonego algorytmu o zwiększonej liczbie zmiennych. Opisany w ten sposób algorytm staje się łatwy do napisania w komputerze, w wybranym języku programowania. Wówczas ilość użytych zmiennych nie odgrywa tak istotnej roli, jak w przypadku regulatorów (gdzie ilość pamięci jest dużo mniejsza). Poniżej zaprezentowano wyliczenie ilości potrzebnych zmiennych na poszczególne wartości parametrów algorytmu oraz na parametry obiektu (w nawiasach podano, do jakiego rodzaju pamięci należą zmienne).

a) parametry regulatora (*SSD*)

$N, N_1, N_u, \lambda, T_p$ → 5 zmiennych

b) parametry obiektu (*SSD*)

n_u, n_y, n_A, n_B → 4

wartości wzmocnienia obiektu → $n_u \cdot n_y$

opóźnienia → $n_u \cdot n_y$

stałe czasowe → $n_u \cdot n_y$

ograniczenia sygnałów sterujących :

- przyrost sygnałów → $2 \cdot n_u$
- wartości sygnałów → $2 \cdot n_u$

c) wielomiany A i B (*SSD*)

Wielomiany te wykorzystywane są przy obliczaniu odpowiedzi skokowej obiektu. Są one powiązane z sygnałem wyjścia i sygnałem sterowania w następujący sposób:

$$A(z^{-1})Y_k = B(z^{-1})U_{k-1} \tag{5.1}$$

Z powyższego równania wynika, że stopień wielomianu B dla każdego wejścia może być różny (założono, że model obiektu dla każdej pary wejście – wyjście jest taki sam –

stąd stopień wielomianu A będzie taki sam). W celu uproszczenia obliczeń w dalszych rozważaniach zakładam, że stopnie wielomianu $B(n_B)$ są takie same. Powoduje to konieczność uzupełnienia (w przypadku występowania różnych wartości opóźnień dla różnych wejść) współczynników wielomianu b, zgodnie ze wzorem:

$$B^{ij}(z^{-1}) = b_0^{ij} + b_1^{ij} z^{-1} + b_2^{ij} z^{-2} + \ldots + b_{n_B}^{ij} z^{-n_B} \tag{5.2}$$

Dla macierzy opóźnień dla każdej pary wyjście – wejście :

$$delay(s) = \begin{bmatrix} e^{-2s} & e^{-0s} \\ e^{-1s} & e^{-6s} \end{bmatrix} \tag{5.3}$$

wielomian B będzie równy:

(5.4)

$$\begin{aligned}
B^{11}(z^{-1}) &= 0 + 0 \cdot z^{-1} + b_2^{11} \cdot z^{-2} + b_3^{11} \cdot z^{-3} \\
B^{12}(z^{-1}) &= b_0^{12} + b_1^{12} \cdot z^{-1} \\
B^{21}(z^{-1}) &= 0 + b_1^{21} \cdot z^{-1} + b_2^{21} \cdot z^{-2} \\
B^{22}(z^{-1}) &= 0 + 0 \cdot z^{-1} + 0 \cdot z^{-2} + 0 \cdot z^{-3} + 0 \cdot z^{-4} + 0 \cdot z^{-5} + \\
&\quad + b_6^{22} \cdot z^{-6} + b_7^{22} \cdot z^{-7}
\end{aligned} ,$$

gdzie B^{ij} jest wielomianem związanym z $i-tym$ wyjściem i $j-tym$ wejściem.

Uzupełnienie wartości współczynników wielomianu B polega na tym, aby doprowadzić każdy stopień z wielomianów B_{ij} to tej samej wartości :

$$\begin{aligned}
B^{11}(z^{-1}) &= 0 + 0 \cdot z^{-1} + b_2 \cdot z^{-2} + b_3 \cdot z^{-3} + 0 \cdot z^{-4} + 0 \cdot z^{-5} \\
&\quad + 0 \cdot z^{-6} + 0 \cdot z^{-7} \\
B^{12}(z^{-1}) &= b_0 + b_1 \cdot z^{-1} + 0 \cdot z^{-2} + 0 \cdot z^{-3} + 0 \cdot z^{-4} + 0 \cdot z^{-5} \\
&\quad + 0 \cdot z^{-6} + 0 \cdot z^{-7} \\
B^{21}(z^{-1}) &= 0 + b_1 \cdot z^{-1} + b_2 \cdot z^{-2} + 0 \cdot z^{-3} + 0 \cdot z^{-4} + 0 \cdot z^{-5} \\
&\quad + 0 \cdot z^{-6} + 0 \cdot z^{-7} \\
B^{22}(z^{-1}) &= 0 + 0 \cdot z^{-1} + 0 \cdot z^{-2} + 0 \cdot z^{-3} + 0 \cdot z^{-4} + 0 \cdot z^{-5} \\
&\quad + b_6 \cdot z^{-6} + b_7 \cdot z^{-7}
\end{aligned}$$

Jest to sztuczny zabieg, który nie zmienia wartości współczynników a jedynie wyrównuje ich ilość. Można zauważyć, że w stosunku do równań z 5.4 ilość zmiennych zarezerwowana dla współczynników wielomianu $B^{i\,j}$ rośnie.

W rozdziale 6.2 zostały opisane metody na zredukowanie ilości wykorzystywanych zmiennych

---

Policzymy teraz ile potrzeba zmiennych na poszczególne wielomiany.

- analiza wielomianu $A(z^{-1}) = 1 + a_1 z^{-1} + a_2 z^{-2} + \ldots + a_{n_A} z^{-n_A}$

  W przypadku jednej inercji w każdym torze, ilość zmiennych a jest równa:

  $n_A = 2$
  ilość zmiennych → $n_A \cdot n_y = 2 * 2 = 4$

Dla dwóch inercji w każdym torze ilość zmiennych rośnie do:

$$n_A = 4$$

ilość zmiennych $= n_A \cdot n_y = 4 * 2 = 8$

- analiza wielomianu $B(z^{-1}) = b_0 + b_1 z^{-1} + b_2 z^{-2} + \ldots + b_{n_B} z^{-n_B}$

  W przypadku jednej inercji w każdym torze, ilość zmiennych jest równa:

  $n_B = max\{delay\} + 1$
  ilość zmiennych → $(n_B + 1) \cdot n_u \cdot n_y$,

  gdzie *delay* jest macierzą opóźnień dla poszczególnych par wejście – wyjście.

Dla dwóch inercji w każdym torze ilość zmiennych rośnie do:

$$n_B = max\{opóźnienie\} + 3$$

ilość zmiennych $= (n_B + 1) \cdot n_u \cdot n_y$

Wartości współczynników wielomianów A i B należą do stałych struktur danych (SSD). Nie ulegają one zmianie przez cały czas trwania regulacji.

d) Ilość zmiennych y, u

Zmienne zarezerwowane dla wartości sygnałów sterowania oraz wyjścia układu regulacji.

u: aktualna wartość sterowania i $n_B + 1$ do tyłu * n_u, czyli
$(n_B + 2)$ * n_u

y: aktualna wartość wyjścia i n_A do tyłu * n_y, czyli
$(n_A + 1)$ * n_y

W obliczeniach uwzględniono aktualną wartość sterowania oraz aktualne wyjście układu regulacji. Wartości te należy zapamiętywać w bieżącej chwili k algorytmu, ponieważ będzie on odwoływał się do nich w następnym kroku $k+1$ (aż do chwili $k + n_B + 1$ dla sterowania oraz chwili $k + n_A$ dla wyjścia).

e) Ocena miejsca w pamięci dla wektora $Y^{(0)m}_{k+p|k}$ **- trajektorii swobodnej,** $Y^{(n)\,zad}_k$ **oraz zakłócenia niemierzalnego** d^m_k

Trajektoria swobodna określa jak będzie zachowywać się wyjście układu regulacji, przy nie zmienionej wartości sygnału sterującego.

wymiar wektora → $\{n_y \cdot (N - N_1 + 1)\}$

n_y - liczba wyjść
N - horyzont predykcji
N_1 - pierwszy niezerowy składnik odpowiedzi skokowej
$N_1 = min(opóźnienie) + 1$

wymiar wektora Y^0 → $\{n_y \cdot (N - N_1 + 1)\}$

Wartość zakłócenia niemierzalnego jest obliczana w każdej iteracji algorytmu. Jest ona dodawana do obliczonej trajektorii swobodnej. Wyznacza się ją jako różnice wartości pomiędzy wartością zmierzoną wyjścia układu a wartością wyjścia układu obliczoną dla danej chwili (aktualnej).

wymiar wektora d → $d_m = \{n_y\}$

f) Ocena miejsca w pamięci dla wektora zmian sterowania $\Delta\hat{U}(k)$ oraz macierzy K

Jak już wspomniano macierz K jest efektem finalnym działania regulatora w trybie *„off – line”* (tzn. przed właściwym uruchomieniem procesu regulacji). Wymiar macierzy można określić śledząc równanie służące do jej obliczenia:

$$K = \left(M^T M + \lambda I\right)^{-1} \cdot M^T$$

Po przeprowadzeniu opisanych działań okazuje się, że wymiar macierzy K jest równy:

$$\text{wymiar macierzy } K = \{wymiar\ M_y \quad x \quad wymiar\ M_x\} = \{N_u \cdot n_u \quad x \quad (N - N_1 + 1) \cdot n_y\} \tag{5.4}$$

W każdym kroku algorytmu, na podstawie znajomości macierzy K, możemy obliczyć wartość zmian sygnału sterującego. Opisuje to równanie:

$$\Delta\hat{U} = K \cdot \left[Y_k^{zad} - Y_k^0 \right]$$

Do sterowania w bieżącej chwili wykorzystywane jest jedynie pierwsze n_U wartości tak obliczonego sterowania

$$\text{wymiar wektora } \Delta\hat{U}(k) = \{wymiar\ K_x\}$$

g) Ocena miejsca w pamięci dla macierzy dynamicznej M oraz macierzy S

Najważniejszą macierzą, obliczaną w trybie *„off – line”* jest macierz K. Można powiedzieć, że służy ona do odpowiedniego „wyskalowania” uchybu regulacji i w ten sposób do otrzymania sterowania w bieżącej chwili (dokładniej rzec ujmując po wyskalowaniu i przemnożeniu przez uchyb otrzymujemy macierz zmian wartości sterowania). W algorytmie nie wykorzystuje się bezpośrednio macierzy M ani macierzy S. Dlatego można ich nie uwzględniać w analizie zasobów regulatora. Dużo lepszym rozwiązaniem jest obliczenie macierzy K na komputerze i przesłanie do regulatora. Niemniej jednak warto zdawać sobie sprawę ile zyskuje się wolnych zasobów pamięci decydując się na obliczenia *„off – line”* przesłanie macierzy K do regulatora.

$$\text{wymiar macierzy } M = \{(N - N_1 + 1 \quad x \quad N_u) \quad x \quad wymiar(S)\} = \{(N - N_1 + 1) \cdot n_y \quad \times \quad N_u \cdot n_u\}$$

$$M = \underbrace{\begin{bmatrix} S_{N_1} & S_{N_1-1} & \dots & S_1 & \dots & 0 \\ S_{N_1+1} & S_{N_1} & \dots & S_2 & \dots & 0 \\ \dots & & & & & \\ S_N & S_{N-1} & \dots & S_{N-N_1} & \dots & S_{N-N_u+1} \end{bmatrix}}_{M_y} \Bigg\} M_x$$

Rys. 5.1 Wymiar macierzy M

Macierz S, zwana macierzą dynamiczną, zawiera odpowiednie wartości odpowiedzi skokowych dla każdej pary wejście – wyjście.

wymiar podmacierzy $S = \{n_y \quad x \quad n_u\}$

$$S = \underbrace{\begin{bmatrix} \dots & \dots \\ \dots & \dots \end{bmatrix}}_{S_y} \Big\} S_x$$

Rys. 5.2 Wymiar macierzy S

W ten sposób wymiar macierzy M można przedstawić jako złożenie odpowiedniej ilości macierzy S (por. rozdz. 3).

$$M = \underbrace{\begin{bmatrix} \underbrace{\begin{bmatrix} \dots & \dots \\ \dots & \dots \end{bmatrix}}_{S_y} \Big\} S_x & \dots\dots & 0 \\ \vdots & \ddots & \\ & \ddots & \\ \underbrace{\begin{bmatrix} \dots & \dots \\ \dots & \dots \end{bmatrix}}_{S_y} \Big\} S_x & \dots\dots & \underbrace{\begin{bmatrix} \dots & \dots \\ \dots & \dots \end{bmatrix}}_{S_y} \Big\} S_x \end{bmatrix}}_{M_y} \Bigg\} M_x$$

Rys. 5.3 Wymiar macierzy M z uwzględnieniem wymiaru macierzy S

Zbadanie ilości potrzebnych zmiennych dla konkretnego przypadku.

Przykład 5.1

Jako przykład posłuży obiekt o transmitancji danej wzorem:

$$H(s) = \begin{bmatrix} \frac{1 \cdot e^{-2s}}{1+0.7 \cdot s} & \frac{5 \cdot e^{-4s}}{1+0.3 \cdot s} \\ \frac{1 \cdot e^{-13s}}{1+0.5 \cdot s} & \frac{2 \cdot e^{-15s}}{1+0.4 \cdot s} \end{bmatrix}$$

Parametry obiektu wynoszą:

- $n_A = 2$
- $n_B = 16$

Wybrano następujące parametry regulatora:

- $N = 17$
- $N_u = 15$
- $N_1 = 3$
- $\lambda = 0.07$

Ilość użytych zmiennych przedstawia się następująco :

a) Parametry regulatora (*SSD typu integer*) :

Na parametry regulatora potrzeba 3 wartości zmiennych.

b) Parametry obiektu (*SSD typu integer oraz float*) :

Na parametry obiektu potrzeba 24 wartości zmiennych.

c) wielomiany A i B (*SSD typu float*) :

zmienne a → 2*2=4
zmienne b → 17*2*2=68

d) zmienne wejścia i wyjścia (*I/O typu float*) :

poprzednie wartości sterowania u → 18*2=36
wartości wyjścia y → 3*2 = 6

e) trajektoria swobodna i zakłócenia (*Stos, zmienne typu float*) :

y^0	→	2*15=30
d	→	2
Y^{zad}	→	30

f) macierz K i macierz zmian sterowań (*SSD, Stos typu float*) :

macierz K	→	30*30=900
macierz ΔU	→	2

Aby zmniejszyć ilość miejsca w pamięci zużywaną na macierz ΔU można obliczać tylko pierwsze n_u zmian sterowań, zmniejszając w ten sposób ilość użytych zmiennych

---

g) macierz dynamiczna M i macierz S (nie sumująca się do zasobów pamięci regulatora)

macierz S	→	2*2=4
macierz M	→	(15*2)*(15*2)=900

Wnioski:

Do implementacji algorytmu potrzeba jest 1107 zmiennych, z czego na :

- Stałe Struktury Danych : 1001
- Historie Sygnałów : 42
- Stos : 64

Większa część zmiennych jest zarezerwowana dla stałych struktur danych.
Tak duża liczba zmiennych wynika z dużej zajętości macierzy K. Jest to spowodowane występowaniem opóźnień w obiekcie.

5.2 Ulepszenie algorytmu w celu zmniejszenia ilości zmiennych

Z poprzedniego rozdziału można wyciągnąć wniosek, że większą część obszaru pamięci zajmują *Stałe Struktury Danych*. Ulepszenie algorytmu będzie polegało na zmniejszeniu ich ilości. Najłatwiej będzie usunąć zbędne ilości współczynników wielomianu B. Z równania 3.4

wynika, że uzupełniane współczynniki wielomianu B (do rzędu całej macierzy wielomianów B) są stałe i równe zero. Wykorzystując tą wiedzę można korzystać jedynie z 2 wartości

współczynników dla każdej z par wejście – wyjście (w przypadku modelu jednoinercyjnego). Ilość współczynników redukuje się do $2 \cdot n_u \cdot n_y = 8$! Pociąga to jednak konieczność komplikacji algorytmu: wszędzie w równaniach, gdzie występują współczynniki b, należy sprawdzić czy nie są one równe 0.

Mianowicie do wzorów:

- $$s_k^{m,j} = \sum_{i=1}^{min\{k-1,n_A\}} a_i^m s_{k-i}^{m,j} + \sum_{i=0}^{min\{k-1,n_B\}} b_i^{m,j}$$

- $$y_{k+p|k}^{(0)\,m} = \sum_{i=1}^{min\{n_A,p-1\}} a_i^m y_{k+p-i/k}^{(0)\,m} - \sum_{i=min\{n_A,p-1\}+1}^{n_A} a_i^m y_{k+p-i}^m + - \sum_{j=1}^{n_u}\left[\sum_{i=0}^{min\{n_B,p\}} b_i^{m,j} u_{k-1}^j + \sum_{i=min\{n_B,p\}}^{n_B} b_i^{m,j} u_{k-1+p-i}^j\right] + d_k^m$$

- $$d_k^m = y_k^m - \left[-\sum_{i=1}^{n_A} a_i^m y_{k-i}^m + \sum_{i=1}^{n_u}\sum_{i=0}^{n_B} b_i^{m,j} u_{k-1-i}^j\right]$$

należy włączyć dodatkowy warunek :

$$\begin{aligned} &if\ i < delay(m,j) \quad OR \quad i > delay(m,j)+1 \quad then \\ &\quad b_i^{m,j} = 0 \\ &end \end{aligned}$$

W rozdziale 7.1 pokazano zapis powyższych równań w języku *Matlab*.
Po komplikacji algorytmu ilość wykorzystanych zmiennych algorytmu maleje do 1047.

Kolejny krok zmniejszenia ilości zmiennych bazuje na informacji, że implementowana będzie wersja analityczna algorytmu.
Rozwiązaniem zadania optymalizacji w wersji analitycznej jest wektor zmian sterowania:

$$\Delta U = K \cdot \left[Y_k^{zad} - Y_k^0 \right] \tag{5.5}$$

W bieżącej chwili wykorzystywane jest natomiast tylko pierwsze n_u jego wartości. Pozostałe wartości nie są brane pod uwagę. Można więc wykorzystać tą informację i odpowiednio zmodyfikować postać powyższych macierzy. Zgodnie z zasadą mnożenia macierzy, ilość kolumn K musi być równa ilości wierszy drugiego czynnika występującego w równaniu (5.5). Z

tego względu wymiar wyrażenia $Y_k^{zad} - Y_k^0$ nie może ulec zmianie (a co za tym idzie wymiar poszczególnych składników). Można natomiast zmniejszyć ilość wierszy macierzy K.

Z informacji, że w bieżącej chwili wykorzystywane jest tylko pierwsze n_u wartości macierzy ΔU, wynika konieczna ilość wierszy macierzy K. Jest ona równa n_u. W takim przypadku wymiar macierzy K będzie wynosić:

$$\text{wymiar macierzy } K = \{n_u \quad x \quad N_u \cdot n_u\}$$

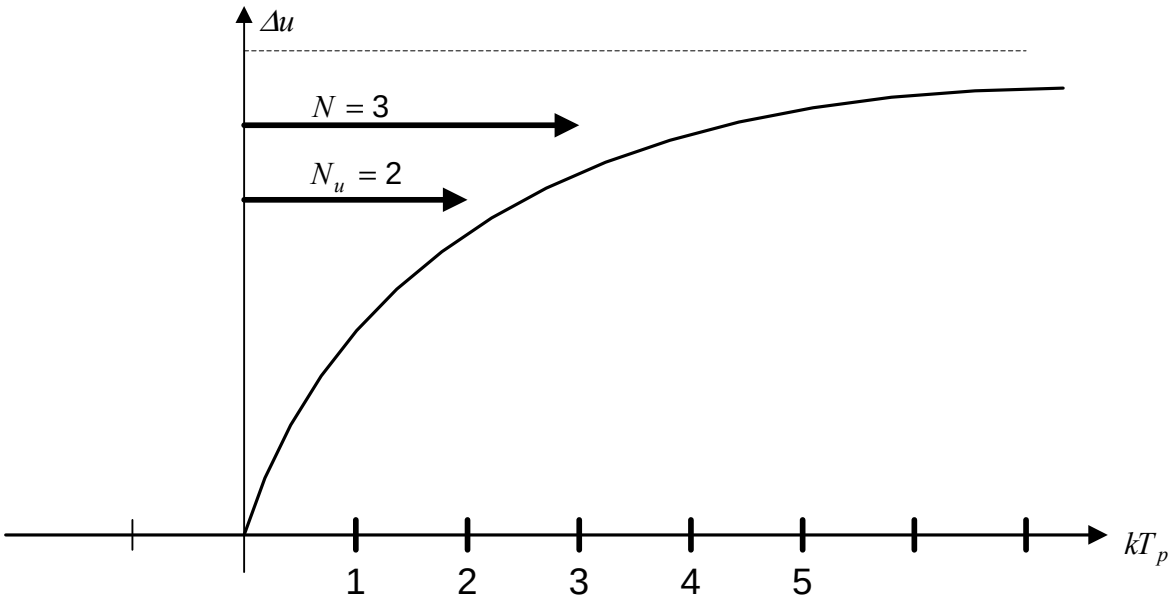
Natomiast ilość zmiennych zarezerwowana na nią 2*15*2 = 60 !
W ten sposób udało się zmniejszyć ilość zmiennych macierzy K z 900 do 60.
Sumaryczna ilość zmiennych, po zastosowaniu opisanych kroków zmniejszyła się do 207 (od wartości początkowej równej 1107).

Opisany sposób zmniejszenia ilości zmiennych daje bardzo dobre rezultaty. Liczba zmiennych wykorzystywana przez algorytm jest rzędu 200 i nie wydaje się ona zbyt wygórowana do jeśli chodzi o zasoby urządzenia.
Kolejny rozdział traktuje o wpływie okresu próbkowania na ilość wykorzystanych zmiennych przez algorytm regulacji.

5.3 Analiza wpływu okresu próbkowania

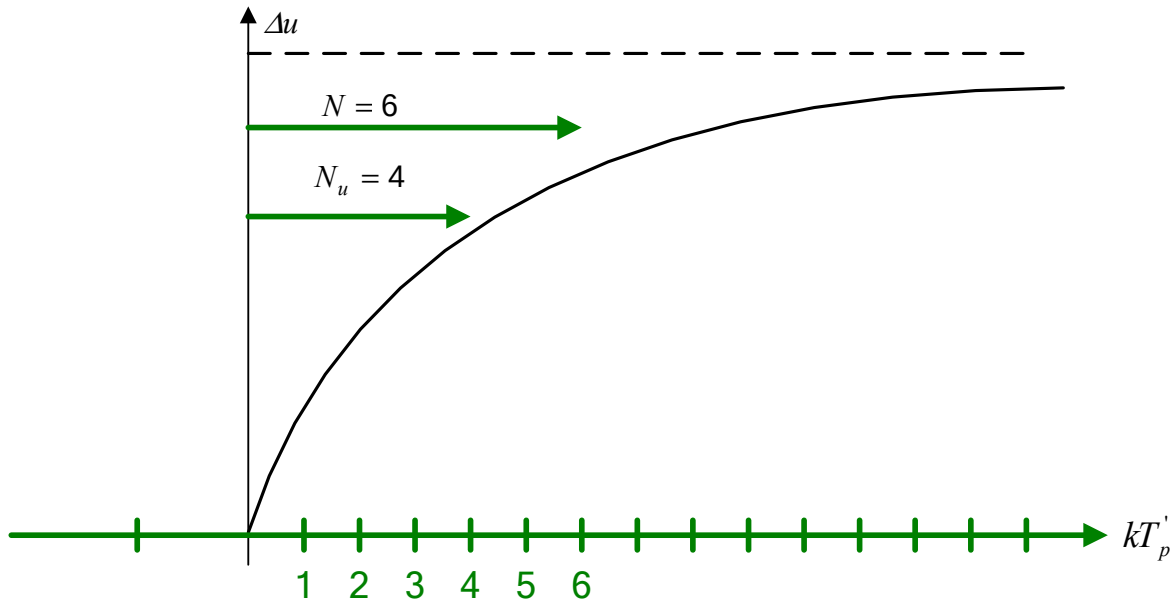
Jak zostało to już wielokrotnie wspomniane, regulacja predykcyjna opiera się na określeniu parametrów modelu obiektu. W przypadku algorytmu GPC ma to o tyle istotne znaczenie, że sterowanie jest obliczane na podstawie obliczeń wyjścia układu w chwilach przyszłych na horyzoncie predykcji. Oznacza to, że użytkownik potrafi określić jak może zachować się obiekt w przypadku skoku wartości sygnałów sterujących. Istotną kwestią, którą należy rozważyć jest wybór okresu próbkowania wyjścia obiektu (lub wyjścia układu regulacji).

Ideą regulacji predykcyjnej z przesuwanym horyzontem jest przewidywanie zachowania się obiektu w przyszłych chwilach. Horyzont predykcji, wyrażony najczęściej w ilości okresów próbkowania, określa na ile chwil do przodu będzie obliczane wyjście układu regulacji. Z kolei horyzont sterowania określa na ile chwil do przodu będą liczone wartości sygnałów sterujących (dla wartości równej 3 obliczane będą wartości sterowanie w chwili bieżącej i na dwie chwile do przodu). Wartości tych parametrów ustalane są przez operatora regulatora. Ich właściwy dobór ma zasadnicze znaczenie dla osiągnięcia dobrych rezultatów regulacji. Pozwalają one bowiem określić zachowanie się obiektu w taki sposób, aby można było wyciągnąć wnioski co do charakteru zmian. Do wyjaśnienia tej kwestii posłużmy się następującym przykładem odpowiedzi skokowej obiektu (na skok sygnału sterującego) :



Rys. 5.4 Przykładowa odpowiedź skokowa obiektu

Okres próbkowania wynosi T_p, a poprawnie dobrany horyzont predykcji jest równy $N = 3$, a horyzont sterowania $N_u = 2$. W przypadku zmiany okresu próbkowania na mniejszy $T_p^{'} = \frac{1}{2} \cdot T_p$ (*Rys. 5.4*), aby zapewnić poprawną pracę regulatora potrzeba już większych wartości horyzontu: predykcji $N = 6$ i sterowania $N_u = 4$.



Rys. 5.5 Przykładowa odpowiedź skokowa obiektu ze zmienionym okresem próbkowania

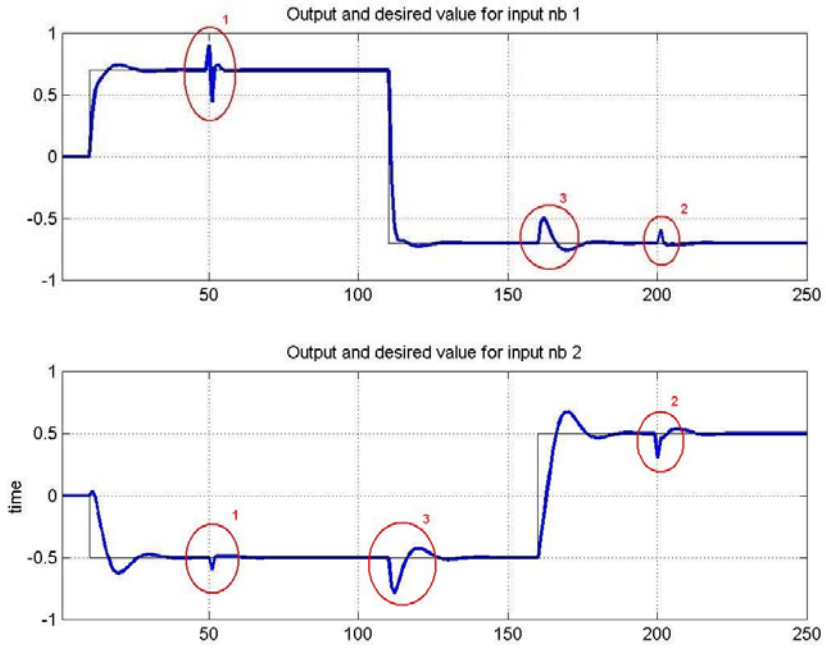
Z powyższych rozważań płynie wniosek, że :

Wybór okresu próbkowania, z jakim będzie mierzone wyjście układu regulacji ma ogromne znaczenie przy rozważaniach na temat ilości zużywanych zmiennych przez algorytm GPC.

W celu odpowiedzi na pytanie: „Ile zmiennych potrzebuje algorytm ze zmniejszonym czasem próbkowania ?" posłużono się przykładem 5.1 bez opóźnień w obiekcie i bez ograniczeń na sygnał sterujący. Zastosowano wspomniane ulepszenia aby ograniczyć liczbę używanych zmiennych macierzy K oraz wielomianu B. Wybór parametrów był następujący:

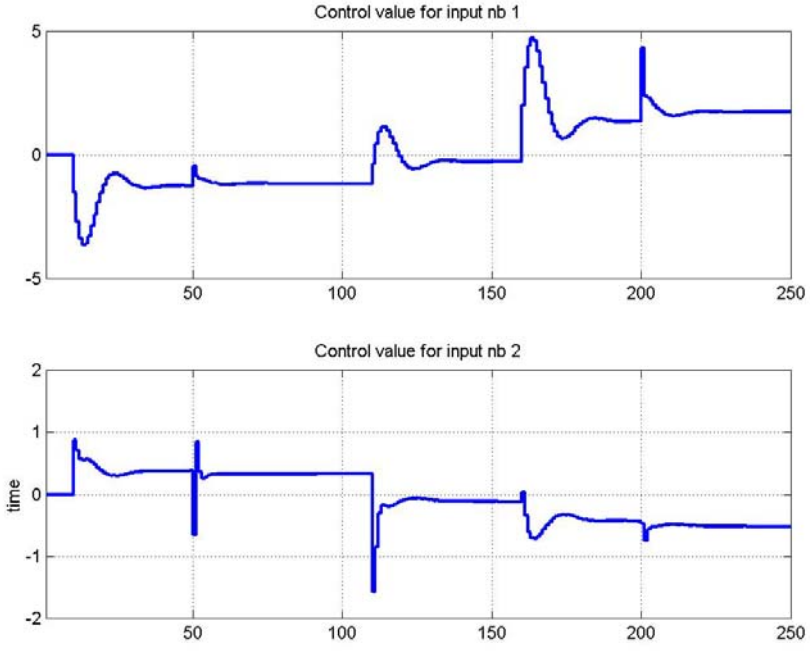
- $T_p = 0.03$ min
- $N = 3$
- $N_u = 2$

Wyniki symulacji przedstawiają poniższe rysunki. Przebiegi pokazane na odcinku czasu odpowiadają 250 okresom próbkowania, przy czym w chwilach 10, 110 i 160 następują skoki wartości zadanych, w chwilach 50 i 200 skoki zakłócenia niemierzalnego odpowiednio na wyjściu 1 i wyjściu 2.



Rys. 5.5 Sygnał wyjściowy i zadany

W chwilach oznaczonych jako 1 i 2 nastąpiły skoki zakłócenia niemierzalnego. Algorytm szybko poradził sobie z wytłumieniem zakłóceń. Charakterystyczne piki, oznaczone na rysunku jako 3, są spowodowane zmianą wartości zadanych na poszczególnych wyjściach. Algorytm szybko doprowadził do wartości zadanych.

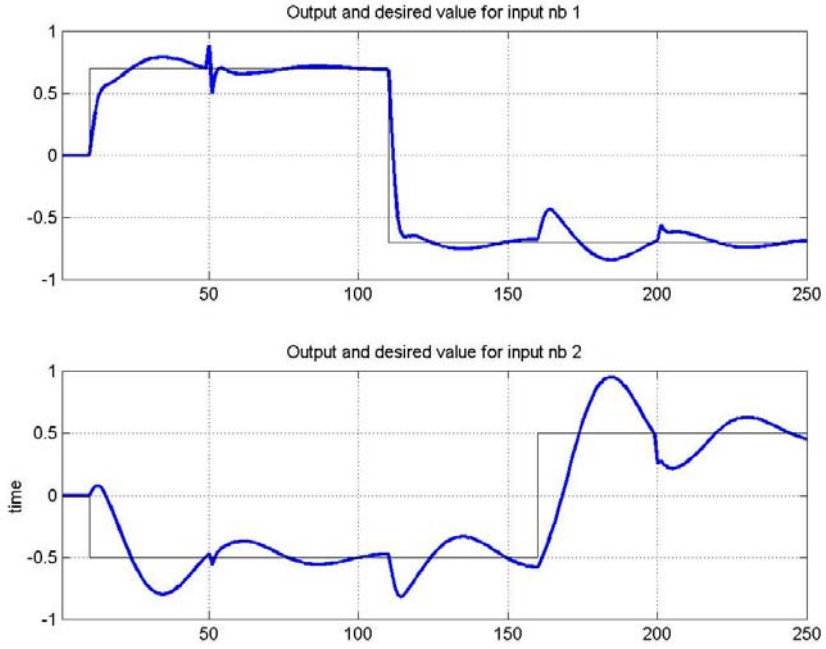


Rys. 5.6 Sterowanie dla układu regulacji z nie zmienionym okresem próbkowania

Przebiegi sterowania są spokojne. Dobra jakość regulacji wynika z odpowiednio dobranych relacji:

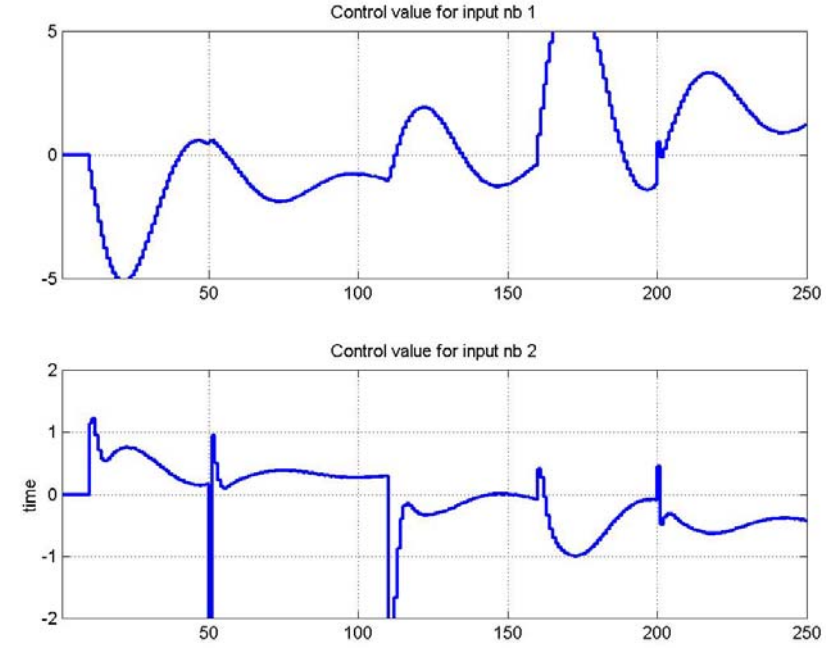
dynamika obiektu – okres próbkowania – horyzont predykcji.

Ilość zmiennych wykorzystywana przez algorytm wynosi 81. Następnie zmniejszono okres próbkowania do wartości $\frac{1}{3} \cdot T_p$. Wyniki symulacji z nie zmienionymi wartościami horyzontów ilustrują kolejne wykresy :



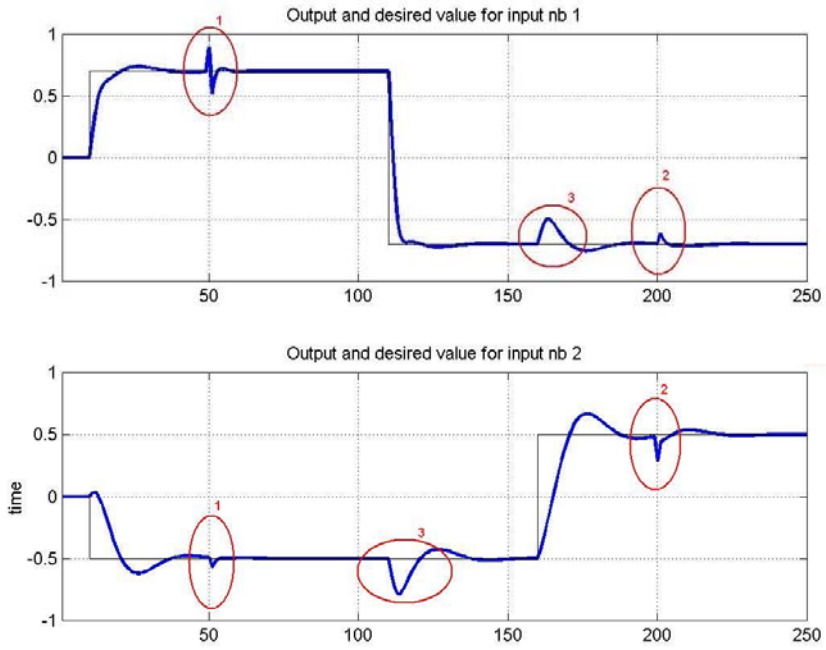
Rys. 5.7 Sygnał wyjściowy i zadany

W tym przypadku algorytm nie radzi sobie z regulacją. Występują duże przeregulowania, szczególnie dla wyjścia numer dwa.



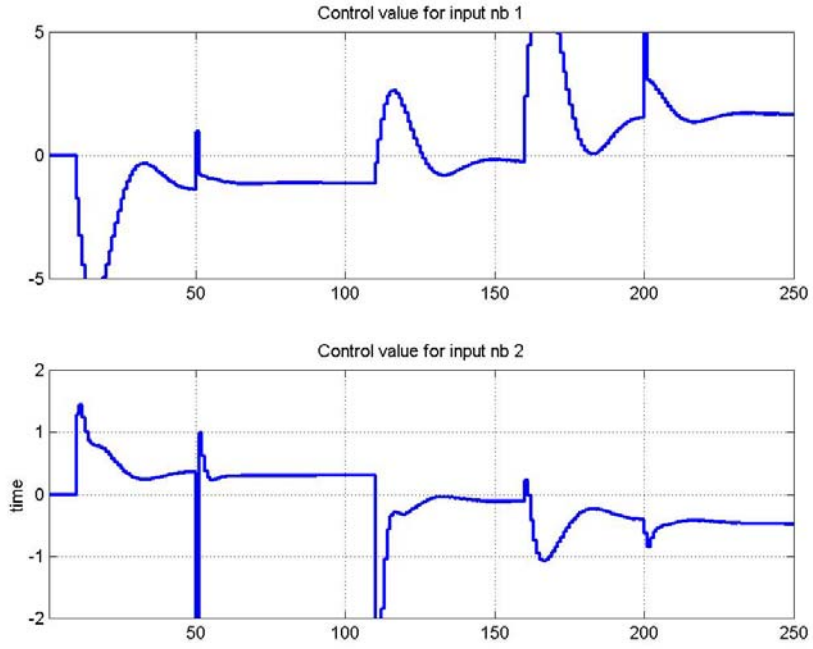
Rys. 5.8 Sterowanie układu regulacji ze zmniejszonym okresem próbkowania

Sygnał sterujący przybiera oscylacyjny charakter. Algorytm radzi sobie już gorzej z regulacją. Wina leży po stronie zbyt małych wartości horyzontów. Jeśli zwiększymy ich wartości: predykcji do $N = 6$ i sterowania do $N_u = 4$ to algorytm poradzi sobie lepiej z regulacją, co prezentują kolejne dwa wykresy :



Rys. 5.9 Sygnał wyjściowy i zadany

Wyjście układu regulacji jest bardzo podobne do tego, z Rys. 5.5.



Rys. 5.10 Sterowanie układu regulacji ze zwiększonymi wartościami horyzontów

Przebiegi sterowania są spokojne. O ile wyjście układu zachowuje się podobnie do tego z rys. 5.5, to sygnały sterujące różnią się w większym zakresie. Obiekt jest próbkowany z większą

dokładnością więc sterowanie może szybciej reagować na działanie zakłóceń lub zmianę wartości zadanych. Stąd przebiegi mają bardziej dynamiczny charakter.

Ilość zmiennych wykorzystywana w tej wersji algorytmu wynosi 105. Powiększeniu uległ rozmiar wektora Y^o (trajektorii swobodnej), wartości zadanych oraz macierzy K.

Sytuacja komplikuje się jeszcze bardziej, jeśli w obiekcie występują opóźnienia. Jeśli chcemy uzyskać dobrą jakość regulacji, oprócz zwiększenia rozmiaru opisanych macierzy powiększeniu ulegnie również ilość zmiennych potrzebna na historię sygnałów sterujących i wyjść układu. Jak wiadomo (por. rozdz. 5.1) rozmiar historii sygnałów sterujących i wyjścia jest uzależniony od rzędu macierzy B i A, a te z kolei od opóźnień występujących w obiekcie. Dla obiektu, w którym opóźnienie $T_0 = 2$ [s] a okres próbkowania $T_p = 1$ [s], rząd wielomianu B jest równy 3:

$$B = 0 + 0 \cdot z^{-1} + b_2 \cdot z^{-2} + b_3 \cdot z^{-3}$$

Jeśli zechcemy próbkować ten sam obiekt z okresem próbkowania równym $T_p^{'} = \frac{1}{2} \cdot T_p = 0.5$ [s] , to aby otrzymać to samo opóźnienie, równe $4 \cdot T_p^{'} = 2$ [s] , rząd macierzy B będzie równy 5:

$$B = 0 + 0 \cdot z^{-1} + 0 \cdot z^{-2} + 0 \cdot z^{-3} + b_4 \cdot z^{-4} + b_5 \cdot z^{-5}$$

Powróćmy do przykładu 5.1 i opóźnieniem występującym w obiekcie. Całkowita ilość zmiennych wykorzystana przez algorytm GPC (po wykonaniu ulepszeń algorytmu) wyniosła 207. Stosując ulepszenie algorytmu zmniejszające liczbę zmiennych wielomianu B i macierzy K oraz chcąc uzyskać dobre rezultaty regulacji (przy zmniejszonym czasie próbkowania $T_p^{'} = \frac{1}{2} \cdot T_p$) dla tego samego obiektu, potrzebna ilość zmiennych wzrośnie do wartości 343 (wzrost liczby zmiennych spowodowany jest większymi wartościami horyzontów predykcji i sterowania). Jeśli nie zastosowano by ulepszeń, to liczba zmiennych zwiększyłaby się jeszcze bardziej, a spowodowane by to było zwiększeniem się rzędu (wielomianu B) n_B oraz dużym wzrostem wymiaru macierzy K.

Kiedy będą miały miejsce opisywane, negatywne sytuacje ? Wtedy, jeśli z pewnych względów regulator nie będzie próbkował z nominalną wartością wyjścia układu. Pozostaje do wyjaśnienia kwestia co należy rozumieć poprzez „nominalny czas próbkowania obiektu”. Określa on czas, jaki jest wystarczający dla regulatora do poprawnej pracy. Określenie to jest bardzo nieprecyzyjne, bo trudno z góry założyć kiedy regulator pracuje poprawnie. Decyzję tą trzeba zostawić operatorowi, który powinien wiedzieć jak zachowuje się obiekt regulacji i z jakim okresem próbkowania wystarczy mierzyć wyjście układu. W warunkach rzeczywistych nie jest konieczny częsty pomiar wartości wyjścia układu, bo powoduje większy nakład obliczeń regulatora a nie poprawia jakości regulacji. Z drugiej strony zbyt długi okres próbkowania nie odda charakteru dynamiki obiektu, co również nie będzie korzystne z punktu widzenia jakości regulacji. Dlatego tak ważny jest odpowiedni dobór okresu próbkowania. Zazwyczaj jego wybór nie nastręcza większego problemu, warto natomiast zdawać sobie sprawę ile powinien on wynosić.

Z przeprowadzonych symulacji można wyciągnąć kilka wniosków praktycznych (por. 7.2.3):

1. długość horyzontu sterowania spełnia ograniczenia $0 < N_u \leq N$. Z reguły przyjmuje się $N_u < N$, aby zmniejszyć wymiar macierzy K a stąd nakład obliczeń,

2. wyboru wartości horyzontów można dokonać metodą symulacyjną,

3. obiekt nie powinien być próbkowany częściej niż jest to konieczne.

Poniżej zaprezentowano możliwy sposób postępowania, który pokazuje kroki, jakie musi przejść użytkownik, aby zaimplementować algorytm predykcyjny dla obiektu. Część kroków wymaga współpracy z konstruktorem (serwisantem), który potrafi określić wpływ życzeń użytkownika na jakość regulacji i nakład obliczeń algorytmu.

1. określić okres próbkowania obiektu
   Można tego dokonać np. symulacjami zachowania się modelu obiektu.

2. określić wartości horyzontów: predykcji i sterowania
   Krok ten jest bezpośrednio związany z wyznaczonym okresem próbkowania. Wybór wartości horyzontów może być konsultowany z konstruktorem. Użytkownik nie musi wiedzieć jakimi własnościami charakteryzuje się obiekt.

3. określić wpływ parametrów regulacji na jakość regulacji i ilość zmiennych
   Wyznaczone w poprzednich punktach parametry regulacji jednoznacznie określają ilość użytych zmiennych. Ma to o tyle duże znaczenie, że regulator ma ograniczone zasoby pamięci. Niepoprawnie dobranie parametrów może uniemożliwić fizyczną implementację ze względu na brak wystarczającej ilości pamięci na zmienne regulowanego procesu. Z drugiej strony, „bezpieczne" ich wartości mogą powodować jakość regulacji nie akceptowaną przez użytkownika.

4. dokonać analizy jakości regulacji dla wybranych parametrów
   Jeśli użytkownik akceptuje wyniki badań dla wybranych parametrów można dokonać fizycznej implementacji.

Punkt 3 schematu postępowania uzależnia parametry regulacji od zasobów fizycznych urządzenia. Kolejny rozdział opisuje zasoby krytyczne regulatora LB – 600.

Rozdział 6

Analiza zasobów regulatora LB – 600

Zasoby krytyczne urządzenia to pamięć i czas obliczeń. W rozdziale tym zostaną opisane zasoby krytyczne pod kątem implementacji algorytmu GPC w wersji analitycznej.

6.1 Zasoby pamięci

W poprzednich rozdziałach została przeprowadzona analiza ilości zmiennych wykorzystywanych przez algorytm predykcyjny. W przypadku obiektu z przykładu 5.1 wynosiła ona 207. Ilość taką udało się uzyskać nieznacznie komplikując algorytm. W przypadku obiektu bez opóźnień ilość zmiennych maleje do 81.

Regulator LB – 600 został wyposażony w moduł pamięci FLASH 128 kB. Może on być dowolnie programowalny a zapisywane parametry mogą dotyczyć obiektu jak i regulatora. Dowolność programowania pozwala na dużą swobodę w dostosowywaniu regulatora do potrzeb użytkownika. Można wyobrazić sobie sytuację, kiedy część zmiennych zostanie usunięta, aby uruchomić nowe funkcje i algorytmy regulacji dostosowane do specyficznych obiektów lub układów regulacji. W programie *LB600mod* uzyskujemy podgląd na wszystkie parametry regulatora, których na chwilę obecną jest ponad 11000. W momencie oddawania pracy, ilość zmiennych, które można wykorzystać do regulacji predykcyjnej przedstawiała się następująco:

- Rejestr Stanów Binarnych (informacja logiczna 0 lub 1) → 59 zmiennych
- Zmienne Tablicowe (typu *float*) → 2000 zmiennych

Struktura Zmiennych Tablicowych oparta jest na tablicy 400 – to elementowej. Każda z takiej macierzy składa się dodatkowo z 5 parametrów. Struktura ta wykorzystywana jest m.in. do regulacji rozmytej PID (ang. *fuzzy PID*), programu dwustrefowego czy regulacji programowej. Poszczególne macierze mogą zawierać parametry regulowanego procesu, inne dla każdego typu regulacji. Z tego względu zmienne tablicowe nadają się bardzo dobrze do regulacji predykcyjnej, która opiera się na zmiennych macierzowych. Ilość 2000 zmiennych wydaje się być wystarczająca do implementacji, tym bardziej, że z poprzednich rozważań wynikało, że maksymalna liczba potrzebnych zmiennych wynosi 205.

6.2 Cykl obliczeń

Obliczenia regulatora przeprowadzane są w pojedynczym cyklu. Jest to okres od momentu odczytu wejść do ponownego odczytu wejść. Kolejne kroki pracy urządzenia w pojedynczym cyklu wyglądają następująco:

- Obsługa wejść analogowych,
- Obsługa wejść binarnych,
- Modyfikacja sygnałów wejściowych,
- Algorytmy regulacji,
- Obsługa wyjść analogowych,
- Obsługa wyjść binarnych.

Obsługa wejść i wyjść dokonywana jest dla każdego kanału regulacji, o ile taki kanał jest aktywny. W przeciwnym wypadku obsługa takiego wejścia lub wyjścia jest pomijana.
Najważniejszą cechą pracy regulatora jest fakt, że cykl obliczeń nie jest stały. Możliwa jest sytuacja, że regulator w pewnym kroku przeprowadzi obliczenia szybciej niż w innym. W ogólności jest to bardzo niepożądane zjawisko. Z punktu widzenia implementacji algorytmów regulacji istotne jest aby od momentu przeczytania wyjścia układu do przeczytania wyjścia układu w chwili kolejnej przez urządzenie minęło tyle samo czasu. Algorytm predykcyjny GPC opiera się na prognozowaniu wyjść układu regulacji w przyszłych chwilach na horyzoncie predykcji N. Dlatego istotne jest aby odczyt wyjścia układu następował w chwilach, na które dokonano obliczeń. W przeciwnym wypadku algorytm nie będzie działał poprawnie.

Zapewnienie odczytu wyjścia układu w stałym rytmie opisano przy okazji implementacji algorytmu w rozdz. 7.2.

Znaczenie, przy omawianiu zasobów krytycznych, ma również czas obliczeń pojedynczego cyklu. Badania przeprowadzone przez firmę LAB – EL wykazały, że cykl pracy urządzenia wyposażonego w standardowy zestaw wejść i wyjść analogowo – binarnych wynosi ok. 0.1 s. Dla obiektów występujących w warunkach przemysłowych wydaje się to wartość wystarczająco mała w stosunku do czasu próbkowania obiektu (w typowych urządzeniach chemicznych np. w reaktorze przepływowym czas próbkowania jest rzędu 1.8 s [Tat02]).
Szczegółowe wyjaśnienie tego problemu znajduje się w rozdz. 7.2).

Rozdział 7

Implementacja regulacji predykcyjnej

Niniejszy rozdział jest omówieniem przykładów implementacji opisanego algorytmu. Pierwszy z nich to zbiór funkcji napisanych w języku *Matlab.* Algorytm posłużył do sterowania obiektem z przykładu 5.1. Wyniki symulacji zostały przedstawione w rozdziale 7.2. Drugi przykład opisuje próbę implementacji w regulatorze LB – 600.

7.1 Implementacja układu regulacji w programie *Matlab*

Niniejszy rozdział przedstawia implementację modelu obiektu oraz algorytmu GPC w wersji analitycznej w programie *Matlab*.
Program *Matlab* to potężne narzędzie programistyczne, które umożliwia nie tylko napisanie równań opisujących prawo regulacji w języku programowania, lecz również pozwala na bezpośrednie wykorzystanie bibliotek już istniejących algorytmów. Środowisko *Matlaba* (bo tak należy postrzegać ten program) wyposażone jest m.in. w algorytmy optymalizacji, regulacji PID czy regulacji rozmytej (ang. *fuzzy control*). Podstawową zaletą programów pisanych w omawianym programie jest fakt, że obliczenia dokonywane są na macierzach. Pozwala to na szybkie przeprowadzenie obliczeń (bez konieczności pisania specjalizowanych funkcji operujących na macierzach). Poszczególne funkcje umieszcza się w tzw. *m – plikach* (plikach z rozszerzeniem *.m* o takiej samej nazwie jak nazwa funkcji *)*, a wnętrza funkcji napisane są właśnie w języku *Matlab*.

Program składa się z 6 funkcji:

- `gpc`
- `compute_Y0`
- `compute_K`
- `compute_M`
- `compute_S`
- `compute_ab`
- `compute_d`
- `measure_Yob`
- `MATRIX`

Główną część programu stanowi plik `gpc`, w którym opisano parametry obiekty, regulatora oraz zmienne wykorzystywane przez algorytm. Zawiera on kod realizujący wspomniany algorytm. Kolejnym plikiem jest `compute_Y0`, w którym następuje obliczenie trajektorii swobodnej. Do obliczenia jej potrzebna jest znajomość zakłócenia niemierzalnego, które obliczane jest w pliku

`compute_d`. Te trzy pliki stanowią główną część implementowanego algorytmu.
Funkcja `measure_Yob` realizuje pomiar wyjścia obiektu. Program wyposażono w możliwość obliczeń macierzy K (plik `compute_K`). Do jej obliczeń wykorzystywane są kolejne 2 funkcje: `compute_M` oraz `compute_S`. Plik `compute_ab` oblicza współczynniki macierzy A i B na podstawie transmitancji obiektu.

Aby umożliwić przenośność kodu na inne środowiska programistyczne, do obliczeń wykorzystano funkcję `MATRIX`. Realizuje ona operacje na macierzach, takie jak dodawanie, odejmowanie i mnożenie.

Układ regulacji składał się z obiektu i regulatora GPC. Zarówno obiekt jak i regulator zaimplementowane zostały w komputerze PC. Poniżej przedstawiono opis poszczególnych części układu.

7.1.1 Implementacja modelu obiektu

Model regulowanego obiektu jest podany macierzą transmitancji:

$$\begin{bmatrix} Y_1(s) \\ Y_2(s) \end{bmatrix} = \begin{bmatrix} \frac{1 \cdot e^{-2s}}{1+0.7 \cdot s} & \frac{5 \cdot e^{-4s}}{1+0.3 \cdot s} \\ \frac{1 \cdot e^{-13s}}{1+0.5 \cdot s} & \frac{2 \cdot e^{-15s}}{1+0.4 \cdot s} \end{bmatrix} \cdot \begin{bmatrix} U_1(s) \\ U_2(s) \end{bmatrix} \tag{7.1}$$

Aby móc skorzystać z algorytmu GPC należy doprowadzić powyższą równość do postaci :

$$A(z^{-1})Y_k^m = B(z^{-1})U_{k-1}$$

czyli *de facto* wyznaczyć wartości współczynników macierzy A oraz B. Funkcja

```
function [a, b] = compute_ab(K1, T2, Tp, n_u, n_y, delay, n_B)
```

oblicza te wartości. Jako argumenty wywołania podaję parametry modelu obiektu będące macierzami: wzmocnienie K_1, stałe czasowe T_2 i macierz opóźnień *delay* oraz skalary: ilość wejść sterujących n_u, ilość wyjść regulowanych n_y, okres próbkowania T_p oraz rząd macierzy B - n_B. Zwracanymi parametrami są macierze współczynników a i b. Interesujący jest sposób obliczania współczynników b :

```
% computing b
d = 0;
for m=1:n_y
   for k=1:n_m

      b(m, k, 1) = numinator(m,k);
      b(m, k, 2) = numinator(m,k) * denuminator(m, 1 + mod(k,n_u));

   end
end
```

Dla obiektu jednoinercyjnego przechowywane są jedynie dwie wartości współczynników (dla każdej pary wejście – wyjście). Warto przy okazji zaznaczyć, że indeksy macierzy w *Matlabie* zaczynają się od 1. Z tego względu współczynnik $b_0^{i\,j}$, występujący we wzorze, będzie miał w *Matlabie* indeks $b_1^{i\,j}$.

Mając wartości powyższych współczynników (a i b) można już obliczyć wyjście obiektu. Oczywiście w algorytmie regulacji w chwili bieżącej następuje odczyt wartości wyjścia obiektu. Jednak ze względu na fakt, że nie dysponujemy rzeczywistym obiektem, a jedynie jego modelem matematycznym, odczyt wyjścia w bieżącej chwili działania algorytmu będzie polegał na obliczeniu jego wyjścia.

```
function Y_ob = measure_Yob(a, b, Y_ob, U, delay, k)
```

Symulując zachowanie się obiektu, użyto wartości sygnału sterującego w postaci wektora, którego rozmiar zwiększa się z każdym krokiem algorytmu (a nie rejestru przesuwnego). Zastosowanie rejestru przesuwnego w żaden sposób nie wpływa na wynik obliczeń, natomiast zastosowanie tak prezentowanego sygnału sterującego miało jedynie na celu pokazanie, że modelowany jest obiekt rzeczywisty, który w żaden sposób „nie przesuwa" wartości sterowania. Zamieszczony poniżej kod programu obrazuje stopień komplikacji obliczeń dla pierwszego wyjścia:

```
% obliczenie wyjścia numer 1
Y_ob(1, k) = -a(1,1)*Y_ob(1,k-1) - a(1,2)*Y_ob(1,k-2);

% pierwszy tor sterowania
way   =  1;
if k-1 - delay(1,1) <= 3

   Y_ob(1, k)  =  Y_ob(1, k) + b(1,1,way)*0  +  b(1,1,way+1)*0;

elseif k-1 - delay(1,1)  <= 4

   Y_ob(1, k)  =  Y_ob(1, k) + b(1,1,way)*U(1,k-1 - delay(1,1))   +
               +  b(1,1,way+1)*0;
else

   Y_ob(1, k)  =  Y_ob(1, k) + b(1,1,way)*U(1,k-1 - delay(1,1))   +
               +  b(1,1,way+1)*U(1,k-1 - delay(1,1)-1);
end

% drugi tor sterowania
way   =  1;
if k-1 - delay(1,2) <= 3

   Y_ob(1, k)  =  Y_ob(1, k) + b(1,2,way)*0                       +
               +  b(1,2,way+1)*0;

elseif k-1 - delay(1,2) -1 <= 3

   Y_ob(1, k)  =  Y_ob(1, k) + b(1,2,way)*U(2, k-1 - delay(1,2))  +
               +  b(1, 2, way+1)*0;
else
   Y_ob(1, k)  =  Y_ob(1, k) + b(1,2,way)*U(2, k-1 - delay(1,2))  +
               +  b(1, 2, way+1)*U(2, k-1 - delay(1, 2)-1);
end
```

Podobne obliczenia należy wykonać dla drugiego wyjścia.
W ten sposób symulowano pomiar wyjścia obiektu w bieżącej chwili k. Kolejny rozdział opisuje, w którym miejscu algorytmu następuje taki pomiar.

7.1.2 Algorytm GPC w wersji analitycznej

Jak już wielokrotnie wspominano, przed przystąpieniem do właściwej regulacji, należy wyznaczyć macierz K:

```
s  =  compute_S(a, b, n_u, n_y, n_A, n_B, N, N1, delay);

M  =  compute_M(s, N, Nu, N1, n_u, n_y);

K  =  compute_K(M, lambda, n_u);
```

Przedstawiona sekwencja wywołań prowadzi do wyznaczenia macierzy K, z tą różnicą, że występują w niej wszystkie wiersze ze wzoru 5.5. Zgodnie z tym co zostało powiedziane w rozdz. 5.2 ilość wierszy macierzy K można ograniczyć do n_u :

```
K  =  K(1:n_u, :);                  %  improvement
```

Mając wyznaczoną macierz K można przystąpić do właściwych obliczeń. Każdy kolejny krok algorytmu implementowano przy użyciu pętli, gdzie kolejne iteracje wyznaczane są dla kolejnychchwil różniących się o T_p

```
T_end = 250;

for k = 4:T_end

      ...
      ...
      ...

end
```

W pierwszej kolejności należy pobrać próbkę wyjścia obiektu (`measure_Yob`). Jeśli występują zakłócenia niemierzalne, można je zasymulować w następujący sposób :

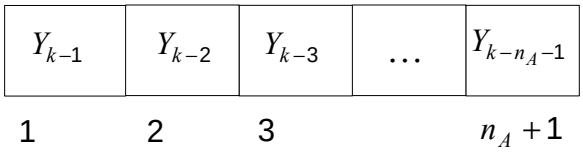
```
d1 = 0;
d2 = 0;
if k >= 50
   d1 = 0.2;
end
if k >= 200
   d2 = 0.2;
end

Y(1, k) = Y_ob(1, k) + d1;
Y(2, k) = Y_ob(2, k) - d2;
```

W tym miejscu algorytm „dysponuje” już wiedzą o wyjściu układu regulacji. Następnym etapem jest aktualizacja rejestru (przesuwnego) : aktualna wartość wyjścia układu (zmierzona przed

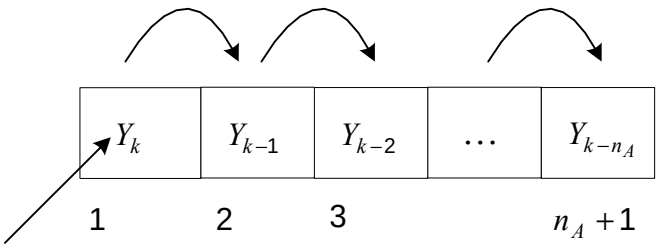
chwilą) jest wpisywana na pozycję odpowiadającą bieżącej chwili, a wartości sterowania

przesuwane na pozycje o jeden dalsze.

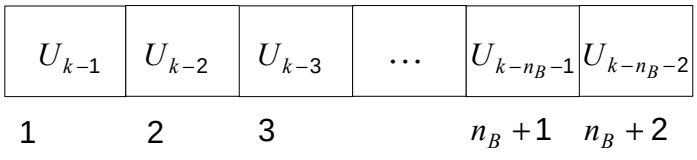
```
for iter = n_A+1 : -1 : 2
   Y_mov(1, iter) =  Y_mov(1, iter - 1);
   Y_mov(2, iter) =  Y_mov(2, iter - 1);
end
for m=1:n_y
   Y_mov(m, 1) =  Y(m, k);
   Y_mov(m, 1) =  Y(m, k);
end

for iter = n_B+2 : -1 : 2
   U_mov(1, iter) =  U_mov(1, iter - 1);
   U_mov(2, iter) =  U_mov(2, iter - 1);
end
```



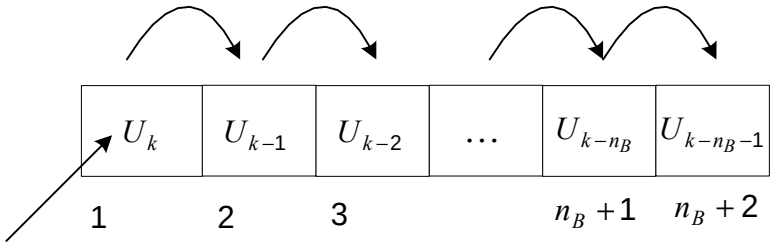
Przy obliczaniu ilości potrzebnych zmiennych dla algorytmu GPC została uwzględniona aktualna wartość wyjścia układu regulacji oraz n_A tych wartości do tyłu. W bieżącej chwili k na pozycji 1 występuje wartość wyjścia obliczona w chwili $k-1$. Należy ją przesunąć na pozycję 2. Z kolei wartość Y_{k-2} z pozycji należy przesunąć na pozycję 3. Ilustruje to rysunek :



Podobne kroki należy przedsięwziąć w przypadku wartości sterowania



Podobnie, należy zapewnić miejsce na wartość sterowania obliczoną w chwili bieżącej.



Na zakończenie obliczeń (po obliczeniu zmiany sterowania) trzeba zaktualizować rejestr i wpisać aktualną wartość.

W każdym kroku algorytmu obliczana jest trajektoria swobodna :

```
Y_0 = compute_Y0(a, b, U_mov, Y_mov, n_y, n_u, n_A, n_B, N, N1, delay);
```

W celu zmniejszenia ilości współczynników macierzy B zastosowano ograniczenie opisane w rozdziale 5.2 . Przy obliczeniu składowej swobodnej $Y^{(0)m}_{k+p|k}$ ograniczenie to można zapisać w postaci :

```
sum4 = 0;
if (i < delay(m, j)) | (i > delay(m, j) + 1)
   sum4 = sum4 + b(m, j, 1) * u(j, 2);
else
   sum4 = sum4 + b(m, j, j+1-delay(m, j)) * u(j, 2);
end
```

wpisanie aktualnie obliczonej wartości

Obliczenie zmian wartości sygnałów sterujących następuje w linii :

```
delta_U = MATRIX(K, MATRIX(Y_Z, Y_0, '-'), '*')
```

Funkcja `MATRIX` dokonuje obliczeń na macierzach (dwa pierwsze argumenty). Wykonywane jest działanie sprecyzowane w trzecim argumencie.

Etapem końcowym obliczenia sterowania jest wprowadzenie ograniczeń na sygnały sterujące. W pierwszej kolejności sprawdzane są przyrosty sygnałów :

```
for m=1:n_y
  if delta_U(m) < delta_Umin(m)
      delta_U(m) = delta_Umin(m);
  end

  if delta_U(m) > delta_Umax(m)
      delta_U(m) = delta_Umax(m);
  end
end
```

Następnie wartości sygnałów:

```
for m=1:n_y
      if U_mov(m, 2) + delta_U(m) < Umin(m)
            delta_U(m) = Umin(m) - U_mov(m, 2);
      end

      if U_mov(m, 2) + delta_U(m) > Umax(m)
            delta_U(m) = Umax(m) - U_mov(m, 2);
      end
end
```

Na koniec obliczana jest wartość sygnałów sterujących:

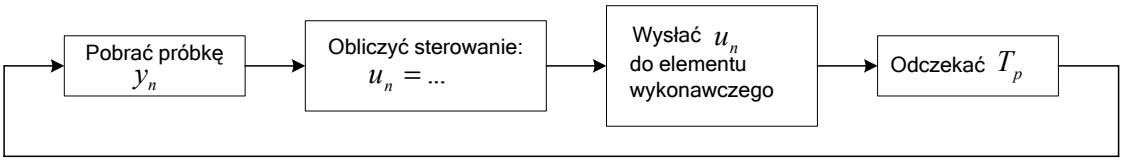
```
for m=1:n_y
      U_mov(m, 1) = U_mov(m, 2) + delta_U(m);
end
```

7.2 Próba implementacji w regulatorze LB - 600

Poniższe rozdziały opisują trudności z jakimi spotkał się autor pracy implementując algorytm predykcyjny z przesuwanym horyzontem GPC w regulatorze LB – 600. Wynikały one przede wszystkim z budowy urządzenia (ograniczonych zasobów pamięci) oraz oprogramowania regulatora (cykl pracy). W algorytmie zdecydowano się na podawanie opóźnienia występującego w obiekcie w ilości okresów próbkowania. Takie założenie jest podyktowane łatwością sprawdzenia poprawności działania algorytmu w sytuacji kiedy osoba pisząca kod źródłowy algorytmu regulacji nie ma możliwości śledzenia zachowania się algorytmu w urządzeniu fizycznym (umieszczenia algorytmu w regulatorze będzie dokonywała firma LAB – EL).

7.2.1 Uwzględnienie cyklu pracy regulatora

Schemat cyfrowego regulatora można przedstawić w postaci następującego schematu blokowego (dla jednego wejścia i jednego wyjścia):



Rys. 7.1 Schemat cyfrowego regulatora

W pierwszej kolejności następuje pobranie próbki sygnału wyjściowego układu regulacji y_n. Na podstawie tej wartości liczony jest sygnał sterujący u_n. Sygnał ten zostanie wysłany do elementu wykonawczego regulatora, który spowoduje oddziaływanie na obiekt. Zasadniczym punktem całego schematu regulacji jest fakt, że regulator odczekuje, od momentu wysłania sygnału sterującego do obiektu, do odczytanie próbki wyjścia układu, okres próbkowanie T_p.

Istotne jest aby regulator przeczytał kolejną próbkę wyjścia układu po upływie okresu próbkowania, natomiast fizycznie może to być zrealizowane na kilka sposobów:

- odczekanie poprzez „zaśnięcie”

  Po wysłaniu sygnału sterującego do elementu wykonawczego można uruchomić funkcję, która będzie czekać przez okres T_p. Przez cały ten okres regulator będzie wysyłał na swoje wyjście stałą wartość sterowania. Po upływie okresu oczekiwania regulator przeprowadzi kolejny cykl obliczeń, czyli pobierze wyjście obiektu, obliczy sygnał sterujący, wyśle obliczony sygnał sterujący do elementu wykonawczego i ponownie „zaśnie” na okres T_p.

- wykonanie kolejnego cyklu obliczeń bez obliczenie sterowania i bez wysłania go na wyjście regulatora

  Po wysłaniu sygnału sterującego na swoje wyjście regulator przeprowadza kolejne cykle obliczeń, jednak nie wysyła wartości sterowania do elementu wykonawczego, dopóki nie upłynie czas T_p. Jeśli w kolejnym cyklu pracy regulatora, czas mierzony przez niego wskaże chwilę różniącą się od poprzedniego momentu wysłania sterowania o czas próbkowania, nastąpi wysłanie obliczonego sygnału sterującego na swoje wyjście (do elementu wykonawczego). Reasumując: obliczenie sterowania można wykonywać tylko w chwilach $k \cdot T_p$ (nominalnych) a pomiędzy nimi wysyłać sterowanie z poprzedniej chwili $(k-1) \cdot T_p$.

Z powyższych rozważań wynika, że odczekanie chwili T_p może być realizowane na kilka sposobów. Pierwsze podejście jest proste w implementacji i nie zużywa zasobów procesora w takiej ilości co podejście drugie. Z kolei sposób drugi wykorzystuje cykl obliczeń regulatora i nie jest uzależniony od funkcji realizującej „usypianie”. Dokonując wyboru jednego z powyższych kryteriów należy uwzględnić jeszcze długość wykonania się jednego cyklu pracy regulatora. Ma on istotne znaczenie ze względu na jakość regulacji. Jeśli cykl obliczeń regulatora (od momentu pobrania wyjścia układu regulacji do momentu wysłania sterowania na swoje wyjście) będzie porównywalny z okresem próbkowania obiektu, to układ regulacji nie będzie działał prawidłowo. W takim przypadku pomiar wyjścia układu regulacji nie będzie następował co okres próbkowania obiektu tylko będzie dłuższy o długość wykonania się jednego cyklu obliczeń regulatora. Jeśli długość cyklu obliczeń jest mała (w stosunku do długości okresu próbkowania) można ją pominąć. Z badań przeprowadzonych przez firmę LAB – EL wynika, że cykl obliczeń regulatora (dla standardowego wyposażenia regulatora) wynosi 0.1 s ale możliwe są do uzyskania wartości nawet 0.06 s (poprzez redukcję pakietów wejściowo – wyjściowych). Cykl obliczeń można więc pominąć.
Ze względu na prostotę implementacji zdecydowano się uwzględnić podejście pierwsze omawianego problemu.

7.2.2 Przykładowa implementacja w języku C

Program napisany w języku Matlab przepisano na język C w celu zaprogramowania regulatora. Program powstał w środowisku Microsoft ® Visual Studio 6.0. Również w tym przypadku zasymulowano zachowanie się układu regulacji złożonego z implementowanego algorytmu i regulowanego obiektu (por. 7.1).

Główne macierze występujące w algorytmie alokowano dynamicznie, aby nie ograniczać wykorzystania kodu źródłowego w przyszłości. Przykładowa deklaracja macierzy K przedstawia się następująco :

```
// rozmiar macierzy K = n_u  x   n_y*(N-N1+1)
double ** K = (double**) malloc(n_u * sizeof(double *));
for( i = 0; i < n_u; i++)
      K[i] = (double*) malloc((n_y*(N-N1+1)) * sizeof(double));
```

Do obliczeń na macierzach użyto funkcji `Matrix` :

```
void Matrix(char sign) {
      int iter;
      switch (sign)
      {
      case '-':

            for (iter=0 ; iter < n_y*(N-N1+1) ; iter++)
            {
                  C_2[iter] = Y_Z[iter] - Y_0[iter];
            }
            break;

      case '*':
            double sum;

            int iter4, i, j;

            i=n_u;
            j=n_y*(N-N1+1);
            sum = 0;
            for (iter = 0 ; iter<i ; iter++)
            {
                  sum = 0;
                  for (iter4 = 0 ; iter4<j ; iter4++)
                        sum = sum + K[iter][iter4] * C_2[iter4];
                  delta_U[iter] = sum;
            }

            break;
      }
}
```

Podobnie implementowano inne funkcje matematyczne: obliczenie rzędu macierzy B :

```
      int compute_nB(int a [2][2])
      {
            int max_v=a[0][0];
            int i, j;
```

```
            for( i=0;i<2;i++)
           {
                  for( j=0;j<n_u;j++)
                {
                       if(a[i][j]>max_v)  max_v=a[i][j];
                }
           }
            return max_v + 1;
      }
```

Obliczenie wartości parametru N_1

```
int parametry::compute_N1(int dela [2][2])
{
      int min_v=dela[0][0];
      int i, j;

      for( i=0;i<2;i++)
      {
        for( j=0;j<2;j++)
          {
                 if(dela[i][j]<min_v)    min_v=dela[i][j];
          }
      }

      return min_v + 1;
}
```

Przepisany algorytm miał posłużyć jako wzorzec do implementacji, dlatego wszystkie wyniki obliczeń zostały zapisane do pliku w celu sprawdzenia ich poprawności. Dokonując testów stwierdzono, że oba programy działają identycznie.

7.2.3 Programowanie regulatora

Zaprogramowanie regulatora LB – 600 tak, aby realizował program regulacji predykcyjnej z przesuwanym horyzontem polega na umieszczeniu w pamięci regulatora odpowiedniego oprogramowania. Oprogramowanie to początkowo napisane w języku C jest kompilowane na język asemblera. Taki sposób postępowania umożliwia przetestowanie algorytmu zanim zostanie umieszczony w pamięci urządzenia. Kolejna bardzo ważna zaleta to możliwość sprawdzenia działania całego oprogramowania w programie *Regulator*. Na program *Regulator* składają się implementowane w urządzeniu fizycznym algorytmy napisane w C.

Program regulacji napisany w języku C zostanie przeniesiony do programu *Regulator* przez informatyka pracującego w firmie LAB – EL. Na potrzeby algorytmu można wykorzystać strukturę regulacji tablicowej. Ustawienia parametrów można wpisywać do kolejnych wartości adresów regulatora

```
K1[0][0] - 1 (0011)
K1[0][1] - 2 (0021)
K1[1][0] - 3 (0031)
K1[1][1] - 4

T2[0][0] - 5
T2[0][1] – 6
```

```
T2[1][0] - 7
T2[1][1] - 8

delta_Umin[0] - 9
delta_Umin[1] - 10

delta_Umax[0] - 11
delta_Umax[1] - 12


Umin[0] - 13
Umin[1] - 14

Umax[0] - 15
Umax[1] - 16

delay[0][0] - 17
delay[0][1] - 18
delay[1][0] - 19
delay[0][1] - 20

Tp - 21
N  - 22

Y_min1 - 23
Y_max1 - 24
Y_min2 - 25
Y_max2 - 26


K[0][0] - 100 (1001)
K[0][1] - 101 (1011)
K[0][2] - 102
K[0][3] - 103
...
K[1][0] - 200
K[1][1] - 201
K[1][2] - 202
K[1][3] - 203
```

Na elementy macierzy K przeznaczono 200 wartości zmiennych. Można przeprowadzić analizę, dla jakich wartości parametrów regulacji można stosować algorytm (ze względu na ograniczoną ilość pamięci przeznaczoną na macierz K) :

$$200 = (N - N_1 + 1) \cdot n_y$$

Jeśli $N - N_1 = 99$ to algorytm będzie działał poprawnie. Dla najbardziej niekorzystnego przypadku, kiedy $N_1 = 1$, aby osiągnąć dobre rezultaty regulacji, wartość horyzontu predykcji musiałaby być równa $N = 100$.

Z powyższych rozważań płynie pierwszy wniosek odnośnie zakresu stosowania algorytmu: jest on dobry dla obiektów, dla których poprawnie wyznaczony horyzont predykcji jest nie większy niż 100.

7.3 Wyniki symulacji

Niniejszy rozdział przedstawia wyniki symulacji, jakie udało się otrzymać wykorzystując analityczny algorytm GPC. Regulowany obiekt jest następującej postaci (por. przykład 5.1) :

$$H(s) = \begin{bmatrix} \frac{1 \cdot e^{-2s}}{1+0.7 \cdot s} & \frac{5 \cdot e^{-4s}}{1+0.3 \cdot s} \\ \frac{1 \cdot e^{-13s}}{1+0.5 \cdot s} & \frac{2 \cdot e^{-15s}}{1+0.4 \cdot s} \end{bmatrix}$$

Wykorzystany algorytm uwzględniał redukcję ilości współczynników B oraz macierzy K, ograniczenia na sygnały sterujące oraz wpływ zakłóceń niemierzalnych. Zastosowano również przesuwny rejestr, którego zadaniem było ograniczenie wymiaru wyjścia układu regulacji oraz sterowania. Uwzględniono cykl pracy regulatora mniejszy od czasu próbkowania obiektu.
Symulowany był algorytm przeznaczony do implementacji w regulatorze.

Przyjęto parametry regulatora :

- horyzont predykcji : $N = 17$
- horyzont sterowania : $N_u = 15$
- horyzont związany z opóźnieniem : $N_1 = 3$
- parametr : $\lambda = 0.075$
  $u_{1\,min} = -2.5$, $u_{1\,max} = 2.5$
- $u_{2\,min} = -0.6$, $u_{2\,max} = 0.6$
  $\Delta u_{1\,max} = 0.5$, $\Delta u_{2\,max} = 0.3$
- ponadto założono, że $\Psi = I$ oraz $\Lambda = \lambda \cdot I$

Po dyskretyzacji i sprowadzeniu elementów każdego wiersza macierzy (obiektu) do wspólnego mianownika otrzymano model w postaci:

$$A(z^{-1})y(k) = B(z^{-1})u(k-1) \quad , \qquad \text{gdzie}$$

$$A(z^{-1}) = \begin{bmatrix} 1 + a_1^1 z^{-1} + a_2^1 z^{-2} & 0 \\ 0 & 1 + a_1^2 z^{-1} + a_2^2 z^{-2} \end{bmatrix}$$

$$B(z^{-1}) = \begin{bmatrix} b_0^{1,1} + b_1^{1,1} z^{-1} & b_0^{1,2} + b_1^{1,2} z^{-1} \\ b_0^{2,1} + b_1^{2,1} z^{-1} & b_0^{2,2} + b_1^{2,2} z^{-1} \end{bmatrix}$$

Elementy odpowiedzi skokowej, dla każdej pary wejście (*j* - te) – wyjście (*m* - te) wyrażają się następująco :

$$s_k^{m,j} = \sum_{i=1}^{min\{k-1,n_A\}} a_i^m s_{k-i}^{m,j} + \sum_{i=0}^{min\{k-1,n_B\}} b_i^{m,j}$$

Stąd macierz M :

$$M = \begin{bmatrix} S_{N_1} & S_{N_1-1} & \dots & S_1 & \dots & 0 \\ S_{N_1+1} & S_{N_1} & \dots & 0 & \dots & 0 \\ \dots & & & & & \\ S_N & S_{N-1} & \dots & S_{N-N_1} \dots & S_{N-N_u+1} & \end{bmatrix}$$

Składowa swobodna Y_k^0 wyjść prognozowanych wynosi:

$$Y_k^0 = \begin{bmatrix} y_1^0(k+N_1) \\ y_2^0(k+N_1) \\ \dots \\ y_1^0(k+N_u) \\ y_2^0(k+N_u) \end{bmatrix}$$

, gdzie

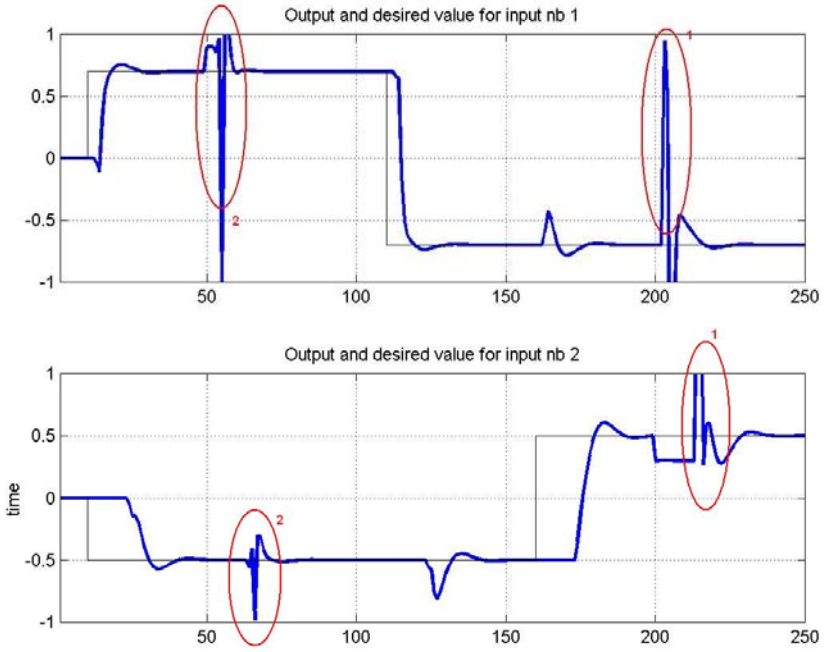
$$y_m^0(k+p \mid k) = \sum_{i=1}^{\min\{n_A, p-1\}} a_i^m y_m^0(k+p-i \mid k) - \sum_{i=\min\{n_A, p-1\}+1}^{n_A} a_i^m y_m(k+p-i) + \\ - \sum_{j=1}^{n_u} \left[\sum_{i=0}^{\min\{n_B, p\}} b_i^{m,j} u_j(k-1) + \sum_{i=\min\{n_B, p\}}^{n_B} b_i^{m,j} u_j(k-1+p-i) + \right] + d_m(k)$$

oraz

$$d_m(k) = y_m(k) - \left[-\sum_{i=1}^{n_A} a_i^m y_m(k-i) + \sum_{i=1}^{n_u} \sum_{i=0}^{n_B} b_i^{m,j} u_j(k-1-i) \right]$$

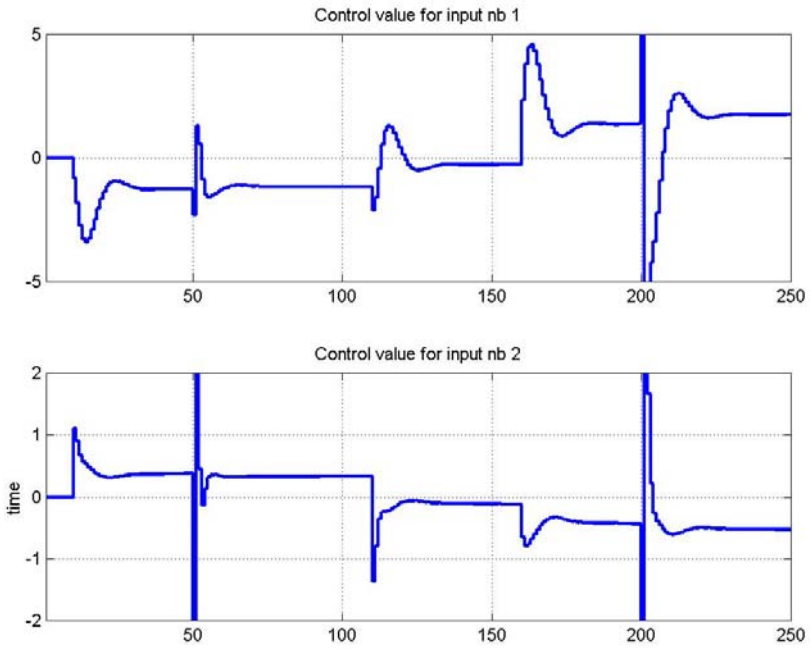
Dla $T_p = 0.03\,min$ oraz przedstawionego modelu obiektu przesymulowano zachowanie się układu regulacji.

W pierwszej kolejności zbadano zachowanie się układu bez ograniczeń sygnałów sterujących. Przebiegi pokazane są na odcinku czasu odpowiadającym 250 okresom próbkowania, przy czym w chwilach 10, 110 i 160 następują skoki wartości zadanych, w chwilach 50 i 200 skoki zakłócenia niemierzalnego odpowiednio na wyjściu 1 i wyjściu 2.



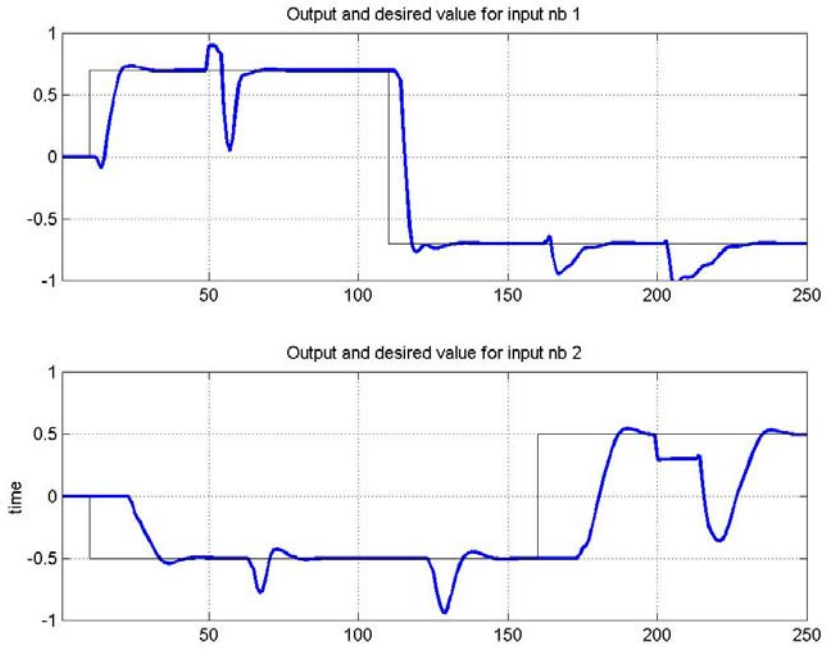
Rys. 7.2 Sygnał wyjściowy i zadany

Chwile oznaczone na rysunku jako 1 i 2 odpowiadają sytuacji skoku zakłócenia niemierzalnego. Powodują one duże przeregulowania w sygnale wyjściowym układu – powstają charakterystyczne piki. Są one spowodowane brakiem ograniczeń nałożonych na sygnały sterujące. W momencie skoku zakłócenia, sygnały sterujące reagują natychmiast powodując dużą zmianę swoich wartości, w chwilach skoku zakłócenia równych 50 i 200.
Warto zauważyć, że wyjście numer 2 reaguje z opóźnieniem równym 13 na zmiany sygnałów sterujących.

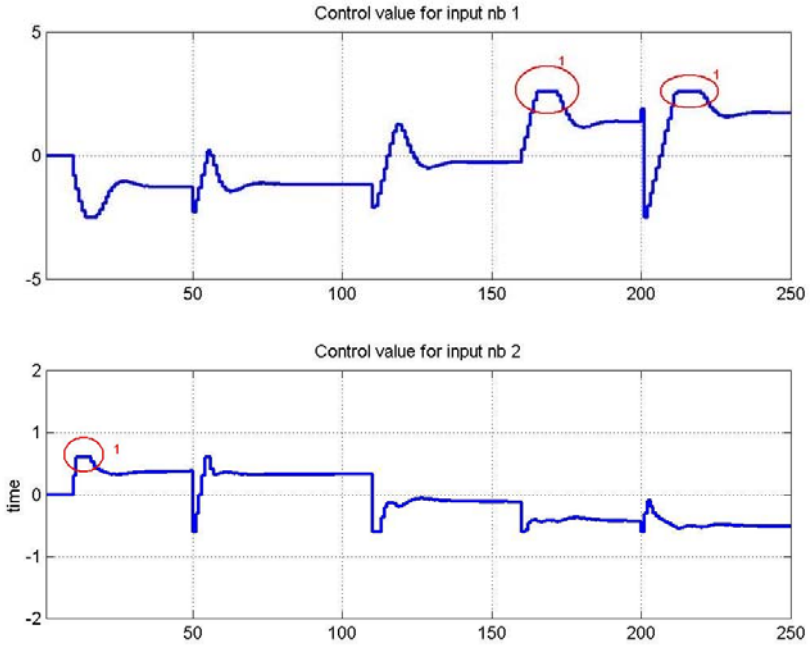


Rys. 7.3 Sygnał sterujący

Przebiegi sterowania gwałtownie reagują na zmianę zakłócenia niemierzalnego. W praktyce takie zachowanie może niebezpieczne dla obiektu, a szybkość zmian fizycznie nierealizowalna.
W sytuacji bez ograniczeń szybszemu nadążaniu za zmianami sygnałów zadanych towarzyszą szybsze i większe zmiany sygnałów sterujących.
Aby zapobiec takim gwałtownym skokom sygnałów sterujących wprowadzono ograniczenia na ich wartości, co prezentują kolejne wykresy.



Rys. 7.4 Sygnał wyjściowy i zadany



Rys. 7.5 Sygnał sterujący

Wprowadzenie ograniczeń na przyrost i wartość sygnałów sterujących powoduje wolniejsze nadążanie za zmianami wartości zadanych oraz wolniejsze tłumienie zakłóceń, jednak przebiegi sygnałów sterujących są spokojniejsze. Fragmenty oznaczone na rysunku jako 1 pokazują ograniczenie sygnałów sterujących do wartości maksymalnych.

Rozdział 8

Podsumowanie wyników pracy i wnioski

Pierwsza część pracy wprowadza w zagadnienia regulacji predykcyjnej z przesuwanym horyzontem. Przedstawiono zasadę działania oraz najczęściej implementowane algorytmy. Opisane są dwie wersje analitycznych regulatorów: DMC i GPC. Opis prawa regulacji regulatora GPC został przedstawiony dokładniej ze względu na implementacje właśnie takiego algorytmu w regulatorze. Uwzględniono w nim wpływ zakłóceń niemierzalnych oraz ograniczeń sygnałów sterujących: ze względu na wartość jak i przyrost wartości. Starano się pokazać, że algorytm identyfikacji obiektu zaimplementowany w urządzeniu może zostać wykorzystany do wyznaczenia parametrów regulowanego obiektu. Stąd wybór implementowanego algorytmu wydaje się być zasadny. W kolejnej części przedstawiono budowę regulatora oraz najważniejsze oprogramowanie do jego obsługi. Oprogramowanie to, napisane przez firmę LAB - EL pozwala na zaprogramowanie struktury regulatora oraz na przesyłanie danych do i z urządzenia. Dzięki temu obliczenia macierzy K może odbyć się na szybszych komputerach (wyposażonych w programy dokonujące obliczeń na macierzach, np. wspomniany program *Matlab*) a tak obliczone wartości przesłać do urządzenia. Analiza zajętości pamięci przez algorytm przeprowadzona jest w kolejnym rozdziale. Ma ona zasadnicze znaczenie przy próbie implementacji algorytmu. Wyniki symulacji dla obiektu jednoinercyjnego z opóźnieniem przedstawiono w rozdziale 7. Prezentują one wyniki implementacji algorytmu na komputerze PC. Przeniesienie algorytmu do fizycznego urządzenia to kwestia zaadresowania zmiennych obiektu w pamięci regulatora i ustawienia poszczególnych części tego algorytmu w stosunku do działań przeprowadzanych przez regulator w pojedynczym cyklu. Przeniesienie algorytmu z poziomu języka C do regulatora zobowiązała się przeprowadzić firma LAB – EL. Jest to podyktowane chęcią chronienia oprogramowanie wykorzystanego do obsługi urządzenia.

W pracy przeprowadzono analizę ilości potrzebnych zmiennych do uruchomienia regulacji. Wykorzystaną pamięć na zmienne podzielono na 3 rodzaje w zależności od pełnionej przez zmienne funkcji: stałe struktury danych (parametry regulatora i obiektu), historie sygnałów (sterujących i wyjścia układu) oraz stos (bieżące obliczenia). Zaproponowano również skuteczne metody na ograniczenie ich ilości. W ten sposób implementacja algorytmu w regulatorze LB – 600 stała się możliwa bez konieczności rezygnowania z części funkcjonalności urządzenia. Zaproponowano wykorzystanie struktury zmiennych tablicowych, używanej pierwotnie do regulacji *fuzzy PID* lub *regulacji programowej*. Napisany algorytm GPC w wersji analitycznej uzupełniono o funkcję symulującą zachowanie się obiektu. Umożliwia to sprawdzenie poprawności zachowania się algorytmu w przypadku samodzielnej implementacji przez osoby zainteresowane. Kody źródłowe algorytmu jak i symulacji obiektu, napisane w języku C, wraz z przykładowym uzupełnieniem parametrów obiektu i regulacji zastały przesłane do firmy LAB – EL w celu umieszczenia ich w programie *Regulator* i ostatecznie: w urządzeniu fizycznym.

Przedstawiona w pracy analiza wykorzystania zasobów urządzenia, pozwala wysnuć wniosek, że implementacja w regulatorze jest możliwa bez żadnych czynności przygotowujących lub zmieniających istniejącą funkcjonalność urządzenia. Przedstawiony opis postępowania w przypadku chęci implementacji algorytmu pozwala użytkownikowi ocenić czy algorytm predykcyjny z przesuwanym horyzontem jest dla niego dobrym rozwiązaniem, bo np. bazuje na informacji o regulowanym obiekcie i uwzględnia ograniczenia na sygnały sterujące.
Napisany kod źródłowy został przetestowany i nie stwierdzono w nim błędów. Wyniki badań zostały przesłane do firmy LAB – EL, która podjęła się próby umieszczenia kodu w urządzeniu fizycznym. W przypadku implementacji zakończonej sukcesem, firma uzyskuje nowoczesny algorytm regulacji, tym samym dołączając do światowych firm oferujących zaawansowane techniki regulacji.

Rozdział 9

Literatura

[Plam06] Sebastian Palmowski, Rozprawa doktorska; WEiTI PW, Warszawa 2005

[Mar02] Piotr M. Marusak, *Regulacja predykcyjna obiektów nieliniowych z zastosowaniem techniki DMC i modelowania rozmytego*, Rozprawa doktorska, Warszawa 2002

[Tat02] Piotr Tatjewski, *Sterowanie zaawansowane obiektów przemysłowych. Struktury i algorytmy*, Warszawa, 2002

[SzkA05] Wojciech Szkolnikowski, *Instrukcja oprogramowania LBX_LB600,* Warszawa 2005

[SzkB05] Wojciech Szkolnikowski, *Instrukcja eksploatacyjna i tablice konfiguracyjne LB-600*, Warszawa 2005

[KSŁ99] Z. Komor, W. Szkolnikowski, W. Łobzowski *Regulator LB – 600*

[Jar03] G. Jarmoszewicz. *Optymalizacja pracy elektrociepłowni przemysłowej z uwzględnieniem dynamiki zmian obciążenia.* Rozprawa doktorska, Politechnika Warszawska. Warszawa 2003.

[Nun01] G. C. Nunes. *Design and Analysis of multivariable predictive control applied to an oil-water-gas separator: a polynomial approach,* Rozprawa doktorska, University of Florida, 2003.

[BL93] A. Ben-Abdennour, K. Y. Lee. Multivariable Robust Control of a Power Plant Deaerator*, IEEE Transactions on Energy Conversion, Vol. 8, No. 1*, 1993

[BGD05] M. A. Brdyś, M. Grochowski, K. Duzinkiewicz, P. Deinrych, J. Wang. *Miękko przełączane sterowanie predykcyjne w zastosowaniu do systemów ściekowych*. *KKAXV*, Tom III, s. 221-226,Warszawa 2005

[Fin74] W. Findeisen. *Wielopoziomowe układy sterowania.*. PWN, Warszawa 1974.

www.label.com.pl