|  |  |
| --- | --- |
|  | Warszawa, dnia 16 września 2020 r. |
|  |  |

Pan

Wiktor Książek

stażysta

Symulacyjne badania algorytmu regulacji dla napędu silnika prądu stałego z mostkiem H w modelu SIL oraz PIL

Projekt wykonano we współpracy z Panem Bartłomiejem Wachem

Z wyrazami szacunku

Opiekun

dr Dariusz Świerczyński

Główny Specjalista

Spis treści

[1. Model napędu silnika prądu stałego z mostkiem H w programie PLECS 3](#_Toc51234108)

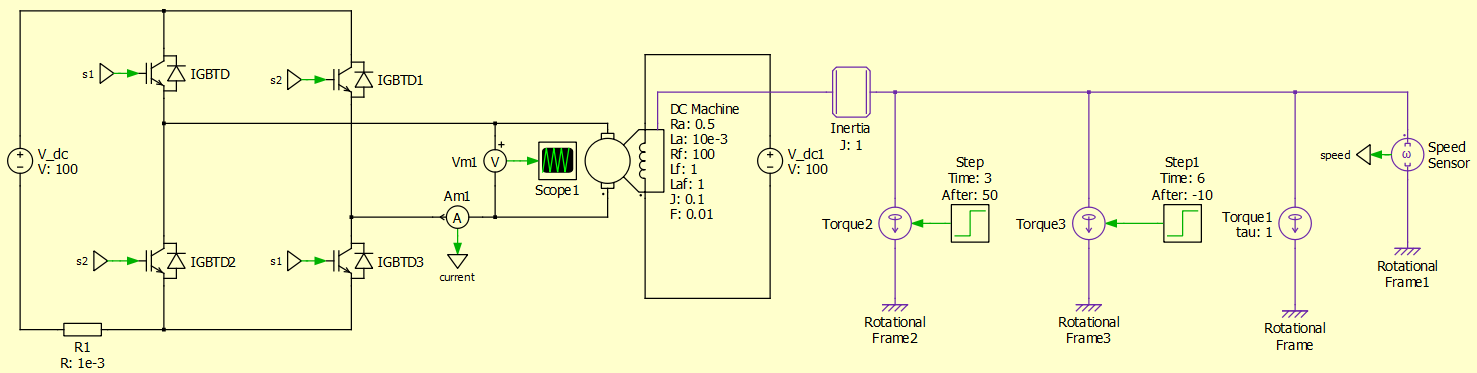
[2. Wykonanie układu regulacji napędu DC w dziedzinie ciągłej s. 7](#_Toc51234109)

[3. Wykonanie układu regulacji napędu DC w dziedzinie dyskretnej z. 10](#_Toc51234110)

[4. Wykonanie układu regulacji napędu w języku C i uruchomienie w SIL. 12](#_Toc51234111)

[5. Utworzenie programu regulującego w języku C w CCS oraz uruchomienie w PIL 18](#_Toc51234112)

# 1. Model napędu silnika prądu stałego z mostkiem H w programie PLECS

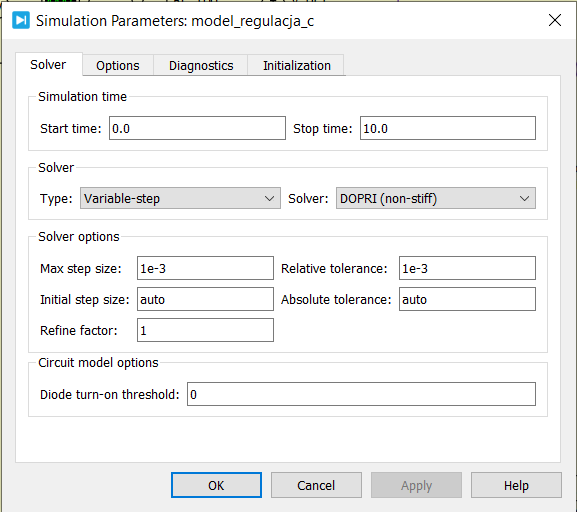


Do rozpoczęcia prac nad stworzeniem regulatora do napędu silnika DC konieczne jest zamodelowanie układu. Topologia układu napędowego wraz z najważniejszymi jego parametrami jest pokazana na rysunku wyżej. Jak widać układ zasilany jest ze źródła napięcia stałego o wartości 100V i rezystancji wewnętrznej o wartości 1e-3 Ohm. Regulacja napięcia na zaciskach twornika maszyny DC jest realizowana na mostku H zbudowanym na idealnych łącznikach (co dyktuje nam oprogramowanie PLECS). Pomimo że, na rysunku umieszczono tranzystory IGBT stanowią one łączniki idealne. Mostek H jest to układ energoelektroniczny pozwalający na zmianę napięcia na swoim wyjściu w zależności od współczynnika wypełniania sygnału PWM sterującego naprzemiennym załączaniem i rozłączaniem par łączników S1 i S2 według wzoru:

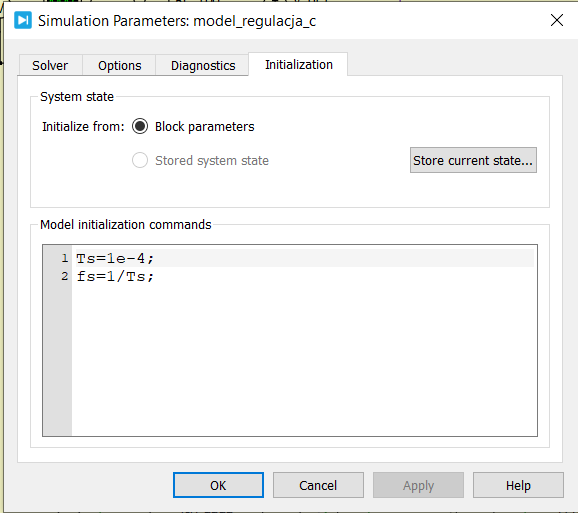
,gdzie D <0,1> nazywamy współczynnikiem wypełnienia, który jest stosunkiem czasu załączenia danego łącznika do sumy czasu załączenia i rozłączenia.

Parametry obwodu twornika ustalono na Ra =0.5 ohm oraz La = 10mH, a obwodu wzbudzenia na Rf= 100ohm, Lf = 1H oraz zasilania z idealnego źródła napięciowego o wartości 100V. Przyjmuje się idealne sprzężenie indukcyjności obydwu obwodów. Parametry mechaniczne maszyny przyjęto na J = 0.1 kg\*m2 (moment bezwładności) oraz F=0.01 (współczynnik tarcia). Maszyna nie posiada odgórnie wyznaczonej wartości prędkości, napięcia czy mocy znamionowej, ponieważ stanowi ona wyłącznie uproszczony model rzeczywistego urządzenia, który wystarcza jednak do przeprowadzenia koniecznych symulacji.

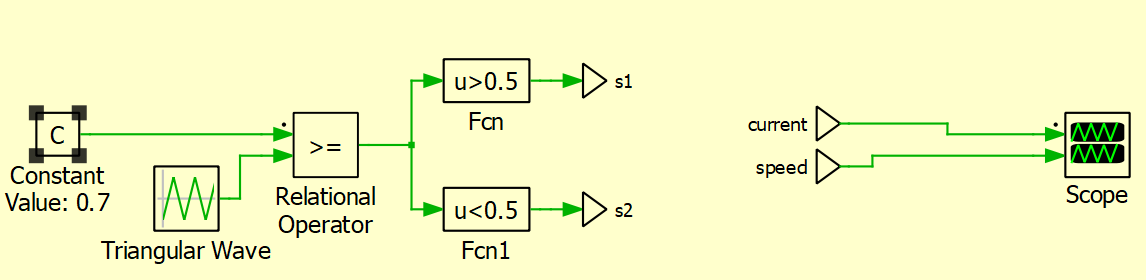
W trakcje trwania dziesięciosekundowej symulacji obciążenie maszyny zmienia się trzykrotnie. W ciągu pierwszych 3 sekund symulacji maszyna pracuje w stanie jałowym (obciążenie stanowi wyłącznie przyłączony wał o momencie bezwładności J=1kg\*m2 oraz moment o wartości M = 1N\*m). W czasie od 3 do 6 sekund po rozpoczęciu symulacji dodane do obciążenia jest 50 N\*m, a po 6 sekundzie odjęte jest 10 N\*m obciążenia. Ważne jest, aby pamiętać o tym, żeby nastawić czas regulacji na odpowiednią długość w oknie „simulation parameters”:



Następnie należy nastawić częstotliwość sygnału nośnego w modulacji PWM, którą wkrótce wykonamy. Wykonujemy to deklarując wartości zmiennych w języku C w zakładce „Initialization”. Częstotliwość (lub okres) tą należy potem także wpisać w parametrach regulatorów dyskretnych, kiedy je wykonamy.

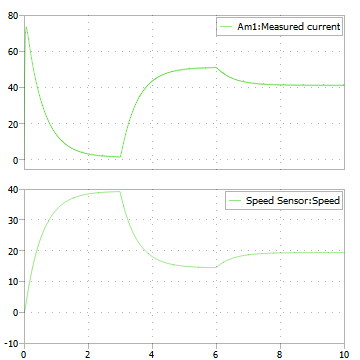


Po wykonaniu modelu układu napędowego konieczne jest wykonanie układu PWM do wysterowania bramek mostka H. Po wykonaniu odpowiedniego układu przedstawionego na poniższym rysunku uzyskujemy w pełni funkcjonalny nieregulowany układ napędowy.



Należy pamiętać o tym, żeby pary S1 oraz S2 pod żadnym pozorem nie były w tym samym stanie logicznym. Taka sytuacja doprowadziłaby do zwarcia na mostku H.

Zgodnie z założeniem na wyjściu mostka H po zadaniu wypełnienia D=0.7 pojawia się w uśrednieniu za okres napięcie o wartości 40V. Brak regulacji prądu oraz prędkości obrotowej maszyny elektrycznej powodują, że po zmianie obciążenia te dwie wielkości zmieniają się w przestawiony poniżej sposób. Prędkość obrotowa zmienia się w zakresie od 0 do 40 rad/s, a prąd w zakresie od 0 do 70 amperów.

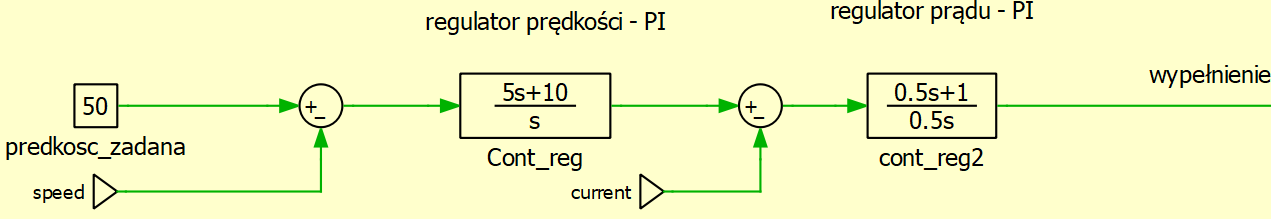


Jeżeli chcemy, aby napęd utrzymywał zadaną prędkość obrotową na wale niezależnie od obciążenia konieczne jest zastosowanie układu regulacji. Układ ten będzie zmieniał wypełnienie sygnału PWM (a w konsekwencji napięcie na wyjściu mostka H), w taki sposób, aby wymusić na tworniku maszyny taki prąd, który odpowiadać będzie założonej prędkości obrotowej.

# 2. Wykonanie układu regulacji napędu DC w dziedzinie ciągłej s.

Sterowanie z wykorzystaniem układu regulacji automatycznej z definicji wykorzystuje sprzężenie zwrotne. W układzie z takim sterowaniem dokonuje się pomiaru wartości wielkości regulowanej, a następnie ten pomiar wykorzystuje się do odpowiedniego wysterowania aktuatora, który wpływa na wielkość regulowaną. W naszym przypadku aktuatorem będzie mostek H, a czujnikami będzie amperomierz (pracujący na tworniku maszyny DC) oraz prędkościomierz (mierzący prędkość obrotową wału maszyny DC).

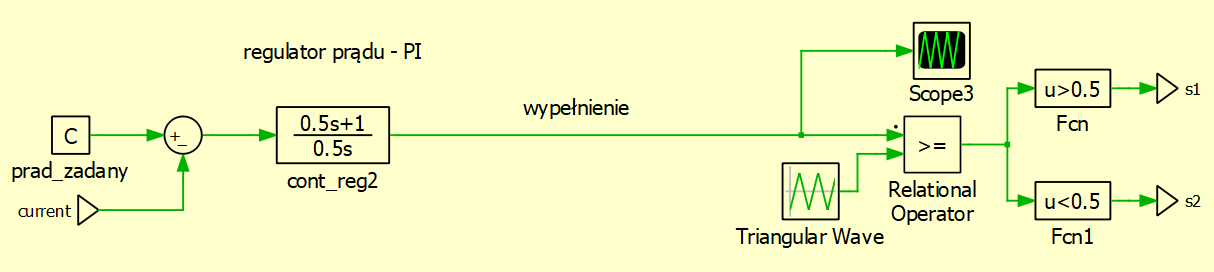
Do regulacji napędu DC wykorzystany zostaną wykorzystane szeregowo połączone regulatory prędkości i prądu. Obydwa regulatory mają formę równoległego regulatora PI. Topologia układu sterowania przedstawiona została na poniższym rysunku:



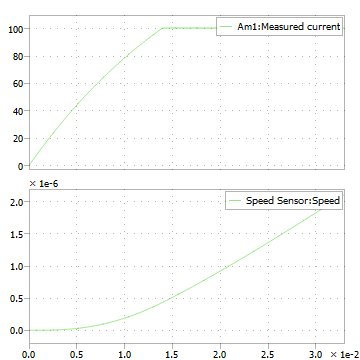
Zastosowanie regulatora PI powoduje, że uchyb ustalony prędkości oraz prądu jest praktycznie zerowy.

W celu zbudowania odpowiednio działającego regulatora konieczny jest odpowiedni dobór nastaw PI.

Na początku dobrano nastawy regulatora prądu metodą eksperymentalną. Po zadaniu prądu referencyjnego prąd płynący przez twornik nie powinien ulegać zmianom w stanie ustalonym. W celu wykonania odpowiedniego badania należy zablokować wał silnika. Można tego dokonać zwiększając do bardzo dużej wielkości moment bezwładności (np. 1e6).



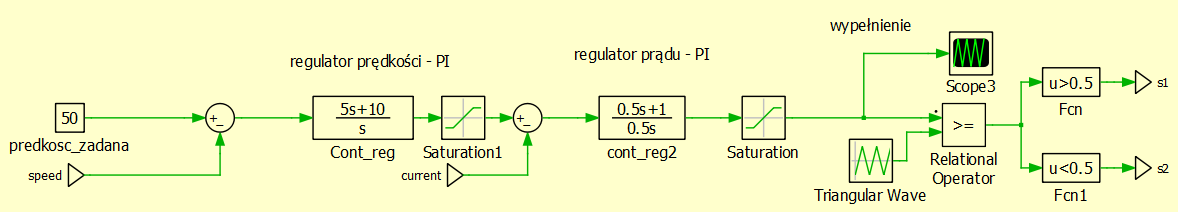
Dobrze działający regulator powinien nie mieć uchybu ustalonego oraz przesadnych tętnień prądu. Dla przedstawionych wyżej nastaw poprawny przebieg prędkości i prądu przedstawiono poniżej:



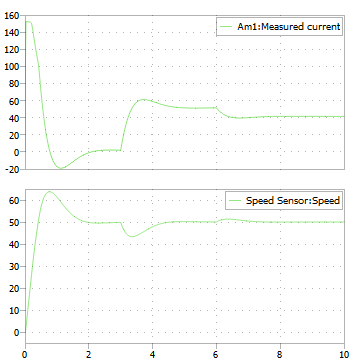
Warto zwrócić uwagę na to, że ten układ regulacji zawiera nasycenie. Wartość wypełnienia poniżej 0 lub powyżej 1 nie wpływa już na zmianę napięcia na wyjściu mostka H. Jest to więc układ nieliniowy.

W przypadku zbyt dużych tętnień prądu wyjściowego należy zmniejszyć wzmocnienie członu proporcjonalnego. Poprawa dynamiki następuje po zwiększeniu wzmocnienia na członie całkującym, ale może prowadzić ono również do zmniejszenia stabilności układu oraz zwiększenia wartości przewartościowania.

Po odpowiednim nastawieniu regulatora prądu można do tego regulatora szeregowo dołączyć regulator prędkości. Jako, że ten regulator nie powinien wymuszać przepływu prądu większego niż maksymalny chwilowy prąd w tworniku to dodano w tym regulatorze nasycenie. Wartość maksymalną prądu przyjęto na 150A.



W celu wyboru odpowiednich nastaw tego regulatora konieczne jest odblokowanie wału maszyny elektrycznej (wartość J wraca do 1). Wzmocnienia członu całkującego oraz różniczkującego należy dobrać z identycznymi kryteriami co regulator prądu. Odpowiednio zaprojektowany układ sterowania powinien oddawać odpowiedź podobną do przedstawionej poniżej:

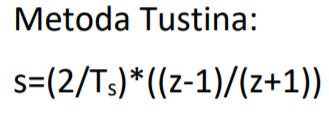


Na wykresach widać, że układ utrzymuje zadaną wartość prędkości obrotowej niezależnie od obciążenia. Oznacza to, że układ regulacji został odpowiednio wykonany w dziedzinie s (regulacja ciągła). Warto zwrócić uwagę, że wartość prądu nie wzrasta powyżej 150A, co oznacza, że nasycenie działa poprawnie.

# 3. Wykonanie układu regulacji napędu DC w dziedzinie dyskretnej z.

Wykonany dotychczas układ sterowania pomimo swojej poprawności nie jest praktycznie implementowany. Przeważająca większość regulatorów stosowanych współcześnie realizowana jest nie na analogowych, lecz cyfrowych układach. W przeciwieństwie do zbioru rezystorów, cewek, kondensatorów oraz wzmacniaczy operacyjnych stanowiących regulator analogowy, regulator cyfrowy oparty na mikrokontrolerze, może przyjąć wartości oraz zwrócić wartość wyjściową regulatora tylko pewnych chwilach czasowych.

Konieczne jest więc przekształcenie transmitancji regulatorów z postaci ciągłej na postać dyskretną. Z paru dostępnych metod dyskretyzacji (metoda Eulera w przód/tył, metoda Tustina) wybrano metodę Tustina. Metoda ta polega na podstawieniu operatora wyrażenia z operatorem z w miejsce operatów z zgodnie ze wzorem:



gdzie Ts to okres odpowiadający częstotliwości fs pracy regulatora.

Przekształcenie transmitancji ciągłych na dyskretne mogą zostać wykonane „na papierze”, ale wymaga to pewnej ilości czasu i skupienia. Alternatywną metodą rozwiązania problemu dyskretyzacji jest wykorzystanie poniższego skryptu w języku Matlab:

clc

clear

s = tf('s');

G1 = (10/s)+5;

D1 = c2d(G1,0.0001,"tustin"); % wyliczanie transmitacji dyskretnej

G2 = (1/(0.5\*s))+1;

D2 = c2d(G2,0.0001,"tustin"); % wyliczanie transmitacji dyskretnej

% wyświetlenie wyliczonych transmitacji

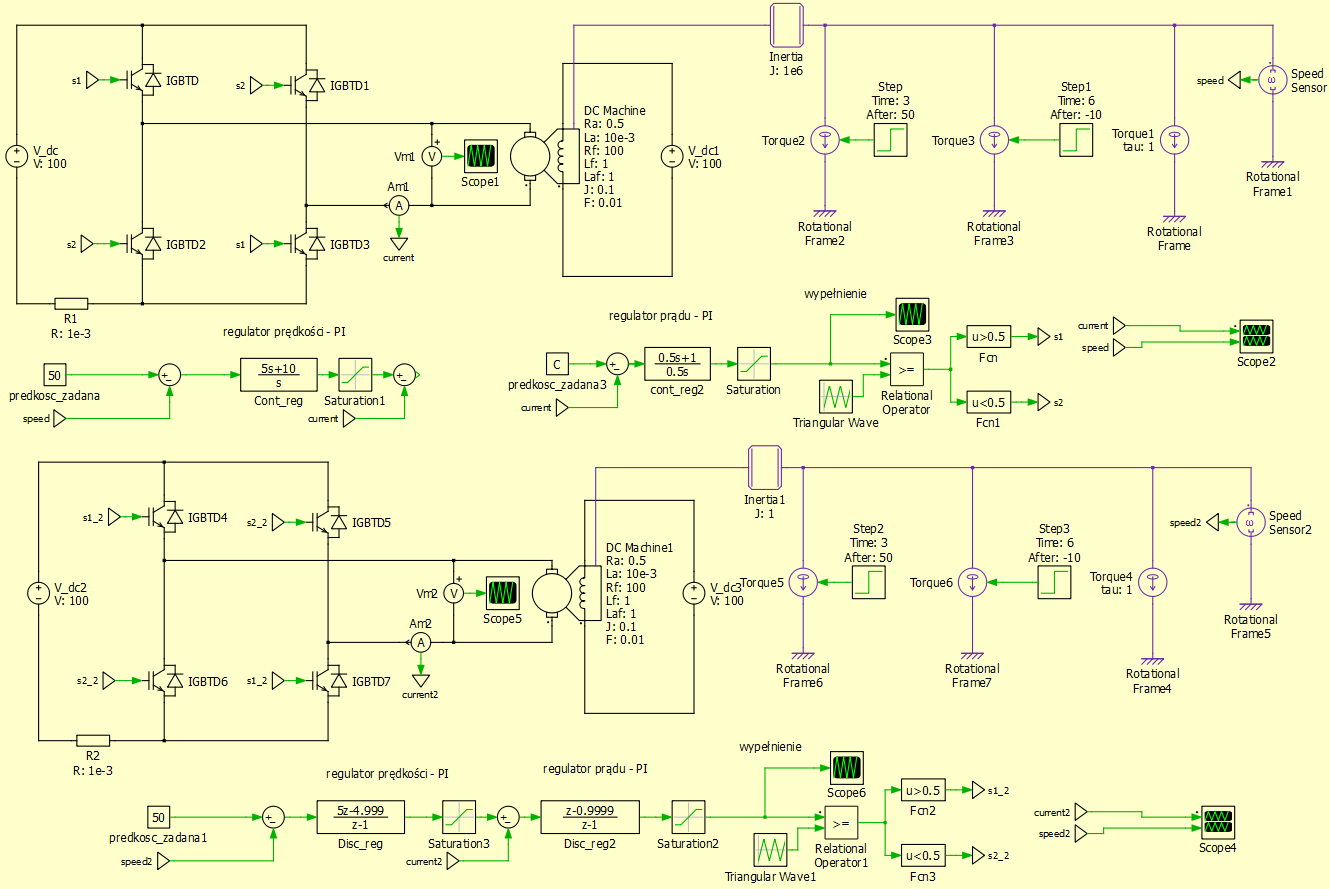
G1

D1

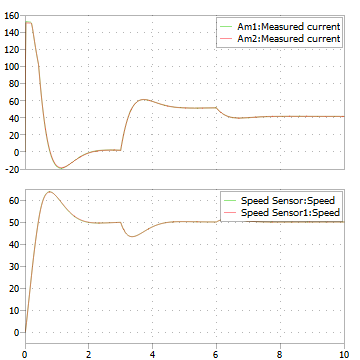
G2

D2

Otrzymane transmitancje dyskretne wyznaczone wyżej należy zamienić z transmitancjami ciągłymi, co pokazano poniżej.



Odpowiedź układu powinna niemal idealnie pokrywać się z odpowiedziami uzyskiwanymi dzięki regulatorom analogowym (ciągłym, czyli w dziedzinie s); oczywiście w momentach odległych od siebie o Ts, co pokazano na wykresach poniżej:

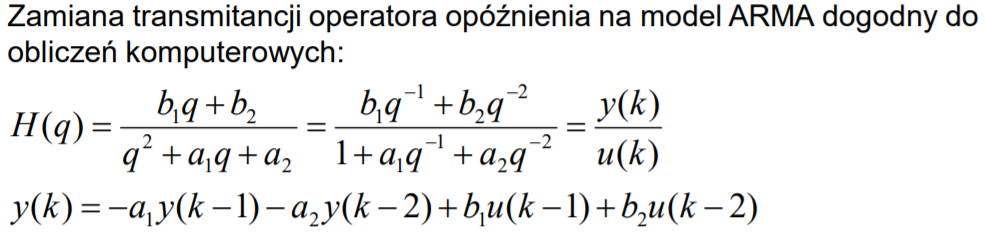


Jak widać na wykresach przebiegi prądu i prędkości napędu z regulatorem ciągłym Am1/Speed Sensor pokrywają się praktycznie idealnie z napędem z regulatorem dyskretnym Am2/Speed Sensor 1.

# 4. Wykonanie układu regulacji napędu w języku C i uruchomienie w SIL.

Po uzyskaniu odpowiednio działających regulatorów w dziedzinie z można już łatwo przekształcić układ regulacji z postaci blokowej do kodu C. W tym celu należy przekształcić transmitancję dyskretną w dziedzinie z na równanie opisane operatorem wyprzedzenia k(który w realizowalnym układzie zamieniamy na operator opóźnienia q), korzystając z modelu autoregresyjnego z ruchomą średnią (ARMA). Pozwoli nam to w prosty sposób wykonać potrzebny algorytm.

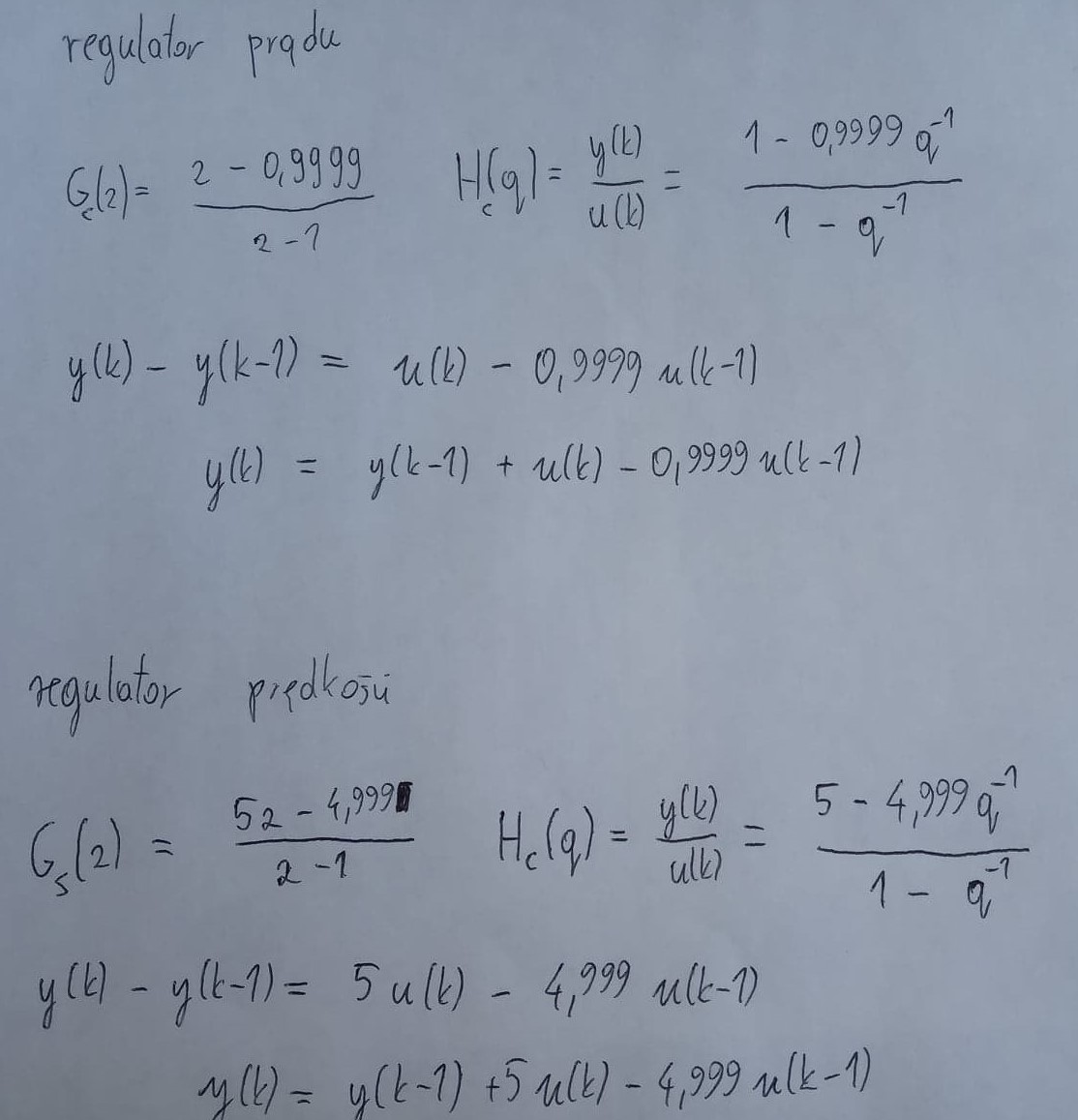
W celu przekształcenia transmitancji regulatora zamieniamy operator z na q (przez podstawienie) oraz wykonujemy przekształcenia zgodnie z poniższym przykładem:



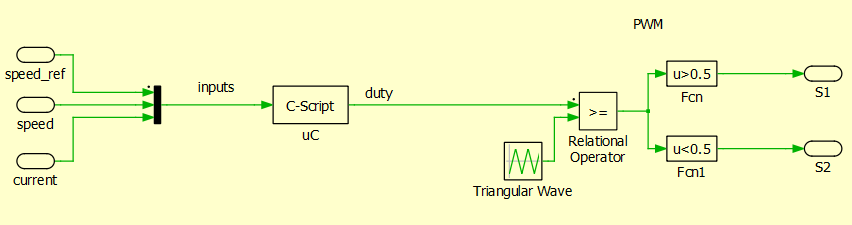
Przykład pochodzi z wykładu Dr inż. Wojciecha Kozińskiego o nazwie „Komputerowe sterowanie obiektami/Regulacja cyfrowa”

Należy pamiętać, że pomnożenie wyjścia/wejścia(y(k)/u(k)) przez operator opóźnienia o potędze -n powoduje, że wejście/wyjście to w równaniu będzie pobierane jako opóźnione o n próbek.

Przekształcenia wykonane dla regulatora prędkości i prądu użytych w przykładowym układzie napędowym przedstawiono poniżej:



Tak wykonane równania wyjściowe można już bezpośrednio użyć w kodzie. Aby dodatkowo uprościć schemat blokowy obwodu regulacji zintegrowano w kodzie C nasycenia na wyjściu (patrz przykład). Tak wykonany układ regulacji zawarto w pojedynczym bloku C-script. Układ regulacji w języku C:



Blok C-Script należy ustawić na 3 wejścia i 1 wyjście. W odpowiednich zakładkach w kategorii „Code” napisano odpowiedni program:

- „Code declarations”:

struct EXAMPLE\_STRUCT{

float s\_ref;

float s;

float diff1;

float diff1\_1;

float reg\_s;

float reg\_s\_1;

float c;

float diff2;

float diff2\_1;

float out;

float out\_1;

};

void Calculate(struct EXAMPLE\_STRUCT \*data){

data->diff1 = data->s\_ref - data->s; //wyliczanie uchybu predkosci

data->reg\_s = data->reg\_s\_1+5 \* data->diff1 - 4.999 \* data->diff1\_1; //regulator prędkości

data->diff1\_1 = data->diff1; //zapisywanie ostatniej próbki

data->reg\_s\_1 = data->reg\_s; //zapisywanie ostatniej próbki

if(data->reg\_s > 150) data->reg\_s = 150; //wykonanie nasycenia prądu 150A

else if(data->reg\_s < -150) data->reg\_s = -150; /wykonanie nasycenia prądu -150A

data->diff2 = data->reg\_s - data->c; //wyliczanie uchybu pradu

data->out=data->out\_1+data->diff2-0.9998\*data->diff2\_1; //regulator prądu

data->diff2\_1=data->diff2; //zapisywanie ostatniej próbki

data->out\_1=data->out; //zapisywanie ostatniej próbki

if(data->out > 1) data->out = 1; //nasycenie wypełnienia

else if(data->out < 0) data->out = 0; //nasycenie wypełnienia

}

struct EXAMPLE\_STRUCT reg1;

- „Start function code”:

reg1.s\_ref=0;

reg1.s=0;

reg1.diff1=0;

reg1.diff1\_1=0;

reg1.reg\_s=0;

reg1.reg\_s\_1=0;

reg1.c=0;

reg1.diff2=0;

reg1.diff2\_1=0;

reg1.out=0;

reg1.out\_1=0;

- „Output function code”:

Output(0)=reg1.out; //wyprowadzenie zmiennej out na wyjście

- „Update function code”:

//sczytanie wejść

reg1.s\_ref=Input(0);

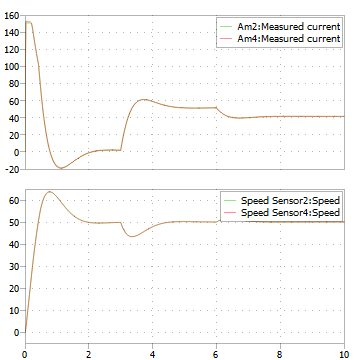
reg1.s=Input(1);

reg1.c=Input(2);

//obliczanie wyjścia

Calculate(&reg1);

W celu sprawdzenia poprawności działania bloku C-Script wraz z programem regulacyjnym (oraz poprawności algorytmu ARMA) porównano przebiegi prądu i prędkości tego układ z układem wykorzystującym regulator dyskretny w postaci blokowej. Wyniki porównania przedstawiono poniżej:



Jak widać układ regulacyjny w postaci blokowej jest praktycznie równoważny do układu zapisanego w kodzie C. Jedyną różnicą między dwoma przebiegami jest to, że układ regulacyjny w kodzie C wolniej wchodzi w nasycenia (w praktyce oznacza to, że maksymalny prąd jest o około 2A większy niż dla układu blokowego). Wynika to najpewniej z tego, że w kodzie C nasycenie jest sprawdzane raz na okres głównej pętli, co może być zbyt rzadkie w porównaniu do układu blokowego.

# 5. Utworzenie programu regulującego w języku C w CCS oraz uruchomienie w PIL

W celu testowania praktycznej implementacji algorytmu sterowania napisano program realizujący układ regulacji na rzeczywistym mikrokontrolerze w środowisku CCS. Projekt wykonany w CCS został wykonany specjalnie z myślą o możliwości symulacji pracy mikrokontrolera wraz z symulacją napędu w programie PLECS. Taki sposób realizacji testowania algorytmu sterowania nazywany jest PIL (Processor in a Loop) i stanowi on następny krok w stosunku do wcześniej pokazanych symulacji SIL (Software in a Loop), które polegają na realizacji algorytmu sterowania wyłącznie na komputerze realizującym symulację całego napędu. Proces rozwoju algorytmu sterowania mikrokontrolerowego przebiega przeważnie (szczególnie w większych projektach) według przedstawionego poniżej schematu:

