Politechnika
Śląska

**POLITECHNIKA ŚLĄSKA**
**WYDZIAŁ AUTOMATYKI, ELEKTRONIKI I INFORMATYKI**
**KIERUNEK AUTOMATYKA I ROBOTYKA**

Projekt inżynierski

# Sprzętowa implementacja regulatora MPC

Autor: Szymon Zosgórnik
Kierujący pracą: dr hab. inż., prof. PŚ Jarosław Śmieja

Gliwice, styczeń 2020

## Streszczenie

Lorem ipsum.

# Spis treści

# Rozdział 1. Wstęp

## 1.1 Motywacja projektu

Lorem ipsum.

## 1.2 Cel pracy

Lorem ipsum.

# Rozdział 2. Idea regulatora MPC

## 2.1 Wstęp

Model predictive control (MPC) jest to zaawansowana metoda sterowania, która polega na takim dobraniu sterowania, aby spełniało ono szereg ograniczeń. Od lat 80. XX wieku algorytm ten wykorzystywany jest przemyśle procesowym w zakładach chemicznych i rafineriach ropy naftowej. W ostatnich latach MPC znalazło zastosowanie także w elektrowniach i elektronice mocy. Sterowanie predykcyjne wykorzystuje dynamiczny model obiektu, najczęściej jest to empiryczny model pozyskany za pomocą indetyfikacji systemów. Główną zaletą MPC jest optymalizacja obecnego przedziału czasowego, biorąc pod uwagę przyszłe stany obiektu. Jest to osiągnięte poprzez optymalizację funkcji jakości w przedziale skończonego horyzontu czasowego, ale z wykorzystaniem jedynie sterowania wyliczonego dla obecnej chwili czasu. Proces ten jest powtarzany z każdą iteracją algorytmu rozwiązującego układ równań różniczkowych opisujących dany układ. Taki schemat regulacji powoduje, że istnieje możliwość przewidzenia przyszłych zdarzeń (występujących zgodnie z podanym modelem wartości zadanej) i podjęcia odpowiednich działań regulujących pracę układem we wcześniejszych chwilach. Sterowanie predykcyjne jest zazwyczaj zaimplementowane jako dyskretny regulator, lecz obecnie prowadzone są badania mające na celu uzyskanie szybszej odpowiedzi przy użyciu specjalnie do tego przygotowanych układów analogowych.

## 2.2 Sposób działania

Zasada pracy regulatora MPC polega na minimalizacji różnic między wartościami predykowanymi: $y_{k+i|k}$ w chwili obecnej $k$ na przyszłą $k+i$, a wartościami zadanymi dla tych wyjść $r(i)$. Przez minimalizację tychże różnic rozumiana jest minimalizacja określonego kryterium jakości $J$. W następnej chwili czasu $(k+1)$ następuje kolejny pomiar sygnału na wyjściu obiektu, a cała procedura powtarzana jest z takim samym horyzontem predykcji $N_p$. W tym celu stosowana jest więc zasada sterowania repetycyjnego bazującego na przesuwnym horyzoncie czasu. W algorytmie regulacji MPC obecny jest także tzw. horyzont sterowania $N_c$ (gdzie $N_c \leqslant N_p$), po którego upływie przyrost sygnału sterującego wynosi zero. W ten sposób zapew-

nione są własności całkujące układu regulacji predykcyjnej.
Algorytmy MPC cechują się następującymi wymogami i właściwościami:

- Wymaganie wyznaczenia wartości przyszłych sygnału sterującego.
- Sterowanie według zdefiniowanej trajektorii referencyjnej dla wielkości wyjściowej.
- Uwzględnienie przyszłych zmian wartości zadanej. Wcześniejsza reakcja regulatora na przyszłą zmianę wartości referencyjnej kompensuje negatywny wpływ opóźnienia na działanie układu.
- Stabilna regulacja obiektów, które nie są minimalnofazowe bez uwzględnienia tego faktu podczas syntezy regulatora.

Realizację metody sterowania predykcyjnego można zapisać w czterech następujących krokach:

1. Pomiar lub estymacja aktualnego stanu obiektu.
2. Obliczenie przyszłych próbek wyjść systemu.
3. Zaaplikowanie sygnałów sterujących tylko do następnej chwili czasu.
4. Powtórzenie algorytmu dla kolejnej chwili czasu.

## 2.3 Model obiektu

Do poprawności działania regulatora MPC niezbędna jest identyfikacja modelu obiektu, który ma być wysterowany. Obecnie wykorzystuje się model w postaci równań stanu, podczas gdy w przeszłości korzystano z modelu odpowiedzi skokowej. Takie podejście wymaga także zaprojektowania obserwatora stanu, używając do tego metod znanych z teorii sterowania. Model obiektu może być zarówno liniowy, jak i nieliniowy. Jednakże, użycie modelu nieliniowego prowadzi do nieliniowej optymalizacji, co powoduje zwieloktrotnienie trudności obliczeniowej. Przekłada się to na zwiększenie wymagań odnośnie częstotliwości taktowania procesora w implementacji sprzętowej. Wynika z tego stwierdzenie, że modele liniowe mają największe znaczenie praktyczne z uwagi na możliwość przeprowadzenia obliczeń w czasie rzeczywistym nawet bez wygórowanych wymagań hardware'owych. Rozwiązaniem tego problemu jest zastosowanie regultaora predykcyjnego w połączeniu z linearyzacją modelu obiektu w konkretnym punkcie pracy. Następnie wyznaczone

są sterowania tak jak dla liniowego przypadku. Tak zrealizowany algorytm gwarantuje jedynie rozwiązanie suboptymalne, jednak nie rzutuje to w żaden sposób na przydatność jego realizacji.

## 2.4 Kryterium jakości regulacji

Jak już wcześniej pokazano w rozdziale 2.2 w celu wyznaczenia wartości sterowań w obecnej i następnych chwilach wyznacza się minimum funkcji celu. Funkcja ta określa jakość pracy regulatora na horyzoncie predykcji. Można stwierdzić, że wartość sygnału sterującego jest wyznaczana poprzez minimalizację wskaźnika jakości regulacji, który jest inheretnie związany z predykcją wyjścia obiektu.
W przypadku skalarnym funkcję celu można opisać następującym równaniem:

$$\begin{cases} J = R_y \sum_{i=1}^{N_p} (r_k - y_{i|k})^2 + R_u \sum_{i=1}^{N_c} (u_{i|k} - u_{k-1})^2 \\ x_{k+1} = Ax_k + Bu_k \\ y_k = Cx_k \end{cases} \tag{2.1}$$

$x_k =$ wektor zmiennych stanu w chwili $k$
$y_k =$ zmienna wyjściowa w chwili$k$
$r_k =$ zmienna referencyjna w chwili$k$
$u_k =$ sterowanie w chwili $k$
$N_p =$ horyzont predykcji
$N_c =$ horyzont sterowań
$i|k =$ predykcja w chwili $k$ odnosząca się do chwili $i$
$R_y =$ współczynnik wagowy wyjścia $y$
$R_u =$ współczynnik wagowy sterowania$u$
$A, B, C =$ macierze przestrzeni stanu

## 2.5 Problem programowania kwadratowego

$$J = \frac{1}{2} U^T H U + W^T U \tag{2.2}$$

$$U = \begin{bmatrix} u_{k|k} - u_{k-1} \\ u_{k+1|k} - u_{k-1} \\ \vdots \\ u_{k+N_c-1|k} - u_{k-1} \end{bmatrix}_{N_c \times 1} \tag{2.3}$$

$$H = \phi^T \phi + R_u \tag{2.4}$$

$$W = \phi^T (R_s - F x_k) \tag{2.5}$$

$$R_u = R_1 \begin{bmatrix} 1 & 0 & 0 & \cdots & 0 \\ 0 & 1 & 0 & \cdots & 0 \\ 0 & 0 & 1 & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \cdots & 1 \end{bmatrix}_{N_c \times N_c} \tag{2.6}$$

$$\phi = \begin{bmatrix} CB & 0 & 0 & \cdots & 0 \\ CAB & CB & 0 & \cdots & 0 \\ CA^2B & CAB & CB & \cdots & 0 \\ \vdots & \vdots & \vdots & & \vdots \\ CA^{Np-1}B & CA^{Np-2}B & CA^{Np-3}B & \cdots & CA^{Np-Nc}B \end{bmatrix}_{N_p \times N_c} \tag{2.7}$$

$$F = \begin{bmatrix} CA \\ CA^2 \\ CA^3 \\ \vdots \\ CA^{Np}B \end{bmatrix}_{N_p \times 1} \tag{2.8}$$

$$R_s = r_k \begin{bmatrix} 1 \\ 1 \\ \vdots \\ 1 \end{bmatrix}_{N_p \times 1} \tag{2.9}$$

## 2.6 Pozostałe rodzaje regulatorów klasy MPC

- Nonlinear MPC
- Explicit MPC
- Robust MPC

## 2.7 Wady i zalety w porównaniu z regulatorem PID

| Cecha | Regulator PID | Regulator MPC |
|---|---|---|
| ograniczenia wartości | brak | uwzględnione w projekcie: twarde albo miękkie |
| optymalność sterowania | brak | charakter optymalny |
| liczba wejść i wyjść | zazwyczaj SISO | MIMO |
| model matematyczny | nie jest konieczny | niezbędny (może być wyliczony iteracyjnie poprzez indetyfikację procesu) |

Tablica 2.1: Porównanie regulacji PID i MPC

# Rozdział 3. Założenia projektowe i wykorzystane narzędzia

## 3.1 Założenia projektowe

- Układ liniowy, dyskretny
- Odpowiedź MCU w czasie rzeczywistym
- Poprawność wyliczonych wartości
- Opisany przypadek MPC do implementacji -> fast gradient method
- Szczególny przypadek regulatora MPC

## 3.2 Architektura systemu

### 3.2.1 Procesor - architektura ARM

ARM (Advanced RISC Machine) jest to rodzina architektur procesorów typu RISC (Reduced Instruction Set Computing). Charakteryzuje się zmniejszoną liczbą instrukcji w porównaniu do CISC, co przekłada się na mniejsze zużycie energii. Wynika z tego zastosowanie architektury ARM w systemach wbudowanych.

### 3.2.2 Platforma STM

STM32 jest to rodzina 32 bitowych mikrokontrolerów produkowana przez STMicroelectronics. Kontrolery są podzielone na odpowiednie serie, jednak łączy je bazowanie na 32 bitowym rdzeniu firmy ARM. Grupy te różnią się m.in. częstotliwością taktowania, obsługiwanymi urządzeniami peryferyjnymi, wsparciem dla arytmetyki zmiennoprzecinkowej, jak i możliwością cyfrowego przetwarzania sygnałów. Użyty w projekcie zestaw uruchomieniowy STM32 Nucleo F401RE jest wyposażony w mikrokontroler STM32F401, który zapewnia wsparcie dla wspomnianej wcześniej jednostki zmiennoprzecinkowej (FPU), jak również instrukcji cyfrowego przetwarzania

sygnałów (DSP). Procesor ten jest oparty na architekturze ARM Cortex M4. Platforma Nucleo za to dostarcza elastyczne możliwości budowania oraz projektowania nowych rozwiązań sprzętowych, zarówno doświadczonym jak i początkującym, użytkownikom. Moduł ten łączy w sobie różne kombinacje aspektów wydajności oraz zużycia energii. W porównaniu do konkurencyjnych platform oferuje znacznie mniejsze zużycie energii podczas pracy z zasilaczem impulsowym. Poniżej zamieszczono pełną specyfikację.

## 3.3 Narzędzia programistyczne

### 3.3.1 Języki programowania C/C++

C jest to proceduralny, strukturalny, statycznie typowany język programowania, który znajduje zastosowanie w implementacji systemów operacyjnych i wbudowanych. C domyślnie zapewnia narzędzia, które w sposób efektywny są kompilowane do kodu maszynowego. Taki kod ma porównywalną wydajność do programów napisanych jedynie za pomocą asemblera. Jest to powiązane ze względną niskopoziomowością języka C, która zrealizowana została m.in. za pomocą ręcznej alokacji dynamicznej pamięci, a także minimalnego wspracia w czasie wykonywania programu.

C++ jest to wieloparadygmatowy, statycznie typowany, kompilowalny język programowania zapewniający znacznie wyższy poziom abstrakcji w porównaniu do języka C. Na potrzeby pracy inżynierskiej wykorzystano najnowszy, w pełni dostępny, standard tego języka - C++17. Wspomniana abstrakcja danych została zrealizowana za pomocą zastosowania paradygmatu programowania obiektowego. Zastosowanie interfejsów i śchowanie"implementacji programu umożliwia lepsze przeprowadzenie testów jednostkowych, a także budowę wieloplatformowych aplikacji. Podejście obiektowe umożliwia ponadto zaprowadzenie znacznie większego porządku w projektowaniu danego rozwiązania. Jednakże największym atutem takiego rozwiązania jest hermetyzacja danych, która zapobiega wprowadzeniu innych wartości do zmiennych przechowywanych w danej klasie w niepożądanym miejscu programu. W przypadku programowania rozwiązań przeznaczonych pod współpracę z systemami wbudowanymi użycie najnowszych możliwości nowszych standardów języka C++ nie jest czasem możliwe. Przykładem jest w tym przypadku obsługa wyjątków, która pochłania znaczne ilości mocy obliczeniowej, jak i potrzebnej pamięci. Zagrożeniem jest także używanie Standard Template Library (STL), biblioteki zawierającej standardowe algorytmy, kontenery i iteratory. Implementacja tejże biblioteki obfita

jest w operacje na stercie, któtych użycie powinno być minimalizowane w świecie systemów wbudowanych ze względu na fragmentację niewielkiej ilości dostępnej pamięci oraz dłuższy czas jej alokacji i dealokacji. Ponadto STL zawiera operacje rzucające wcześniej wymienione wyjątki.

### 3.3.2 Język programowania Python

Wysokopoziomowe ułatwienie testowania z PC.

### 3.3.3 Środowisko MATLAB

Sprawdzone środowisko do obliczeń macierzowych i układów dynamicznych.

### 3.3.4 Cube

Generator kodu do biblioteki HAL.

### 3.3.5 Biblioteka HAL

W miarę wysokopziomowe rozwiązanie jak na standardy mikroprocków, jednak zostały napotkane problemy.

### 3.3.6 CMake

Cross platform make, możliwość kompilacji z różnych systemów operacyjnych.

### 3.3.7 Kompilator i linker

arm none eabi gcc, coś o opensource

### 3.3.8 Regex

Krótko o wyrażeniach regularnych.

## 3.4 Przykład referencyjny

Ten z simulinka + testowanie w matlabie.

## 3.5 Sposób testowania

Skrypt w Pythonie + HIL

# Rozdział 4. Implementacja rozwiązania

## 4.1 Ogólny schemat programu

Przydałby się rysunek z przepływem informacji + jak działa STM w połączeniu z PC.

## 4.2 Szczegóły implementacji - STM

### 4.2.1 Obiektowość

C++, singleton, hermetyzacja danych, clean code

### 4.2.2 Komunikacja

Ramka, schemat, itp

### 4.2.3 Aglorytm

Jak zostało zrealizowane założenie

## 4.3 Szczegóły implementacji - PC

Krótko o skrypcie w Pythonie.

## 4.4 Problemy napotkane podczas realizacji

HAL - przerwania, ramka UARTa Model układu - feasibility

# Rozdział 5. Przykładowe wyniki

## 5.1 Różne parametry układu

## 5.2 Różne parametry regulatora

## 5.3 Różne wartości zadane

# Rozdział 6. Podsumowanie

## 6.1 Wyniki

Krótkie podsumowanie

## 6.2 Wnioski

Zgodność z założeniem

## 6.3 Pomysły na rozwój projektu

Implementacja linearyzacji

# Dodatki

# Dodatek A. Porównanie matlab stm

# Spis rysunków

# Spis tablic

# Spis listingów

# Bibliografia


[1] Model predictive control. https://en.wikipedia.org/wiki/Model_predictive_control, 04.01.2020.

[2] Power method. http://ergodic.ugr.es/cphys/LECCIONES/FORTRAN/power_method.pdf, 04.01.2020.

[3] Understanding model predictive control. https://www.mathworks.com/videos/series/understanding-model-predictive-control.html, 04.01.2020.

[4] Stm32 nucleo-64 boards (mb1136) user manual. https://www.st.com/content/ccc/resource/technical/document/user_manual/98/2e/fa/4b/e0/82/43/b7/DM00105823.pdf/files/DM00105823.pdf/jcr:content/translations/en.DM00105823.pdf, 04.2019.

[5] Stm32f401xd stm32f401xe datasheet. https://www.espruino.com/datasheets/STM32F401xD.pdf, 2015.

[6] G. Wang et al. State-space model predictive control method for core power control in pressurized water reactor nuclear power stations. *Nuclear Engineering and Technology*, strony 3–4, 2016.

[7] Rolf Findeisen Markus Kögel. A fast gradient method for embedded linear predictive control. *Proceedings of the 18th World Congress The International Federation of Automatic Control*, strony 1362–1367, 28.08 - 02.09.2011.