This is my thesis!

Spis treści

ROZDZIAŁ 1

Introduction

1.1 Motivation

I was motivated to write a Phd thesis because I did not want to work directly after finishing my study [1]

1.2 Organization

This thesis is organized as follows, ... We refer to Schön [2] for things you ...

ROZDZIAŁ 2

System eliminacji zakłóceń

2.1 Wstęp

Realizacja systemu eliminacji zakłóceń w odbiorniku polegała na stworzeniu elementu kompatybilnego z radiem USRP. Modułu, który pobiera sygnał z radia i dokonuje przetwarzania, a wynik tego przekierowuje do demodulatora. Przetwarzanie zakłada filtrację adaptacyjną w oparciu o algorytm LMS.

2.1.1 Środowisko GnuRadio

Głównym środowiskiem wspierającym radio USRP jest GnuRadio. GnuRadio pozwala na implementacje własnych bloków w oparciu o języki programowania Python i C++. W jego skład wchodzi narzędzie o nazwie `gnuradio-companion`, które służy do budowy diagramów z bloków o różnym przeznaczeniu. Czynniki te były kluczowe w podjęciu decyzji o doborze tego narzędzia do realizacji projektu.

Diagramy Diagram nazywany jest również projektem i łączy się z wielu bloków funkcjonalnych. Zawiera przynajmniej jedno źródło i jedno ujście sygnału. Tworzy łańcuch w którego ogniwach zachodzi przetwarzanie sygnału. Nadzoruje poprawność połączeń i zarządza wykonywaniem programu.

Bloki Mogą być źródłem sygnału (np. generator), ujściem (np. wyjście audio w PC) lub elementem pośredniczącym (np. filtry) w procesie przetwarzania. Każdy z bloków określa swój typ danych wejściowych i wyjściowych, co ogranicza możliwość połączeń, lecz gwarantuje kompatybilność typów.

2.2 Blok funkcjonalny

2.2.1 Założenia

Pierwszym założeniem projektowym bloku było zapewnienie interfejsu zgodnego ze schematem filtra. Powinien posiadać dwa wejścia: dla sygnału zakłóconego i sy-

gnału odniesienia. Wyjścia również dwa: pierwsze- sygnału odfiltrowanego, drugie- wektora błędu do monitorowania szybkości zbieżności algorytmu.

Kolejny parametr określa liczbę próbek wyjściowych do wejściowych. Blok nie będzie wpływał na liczbę próbek sygnału wejściowego, zatem stosunek n_{wej}/n_{wyj} powinien wynosić 1 : 1. W nomenklaturze środowiska GnuRadio takie przetwarzanie określa się jako synchroniczne.

Typem danych wejściowych jak i wyjściowych będą wartości zespolone próbek. Parametrami do ustawienia przez użytkownika w środowisku graficznym będą:

1. rząd N filtru w postaci liczby całkowitej
2. długość kroku μ w postaci liczby zmiennoprzecinkowej
3. liczba iteracji w postaci liczby całkowitej
4. początkowe współczynniki filtra w postaci wektora liczb o długości N

Parametry zostaną podzielone na wymagane (1, 2) i opcjonalne (3, 4)

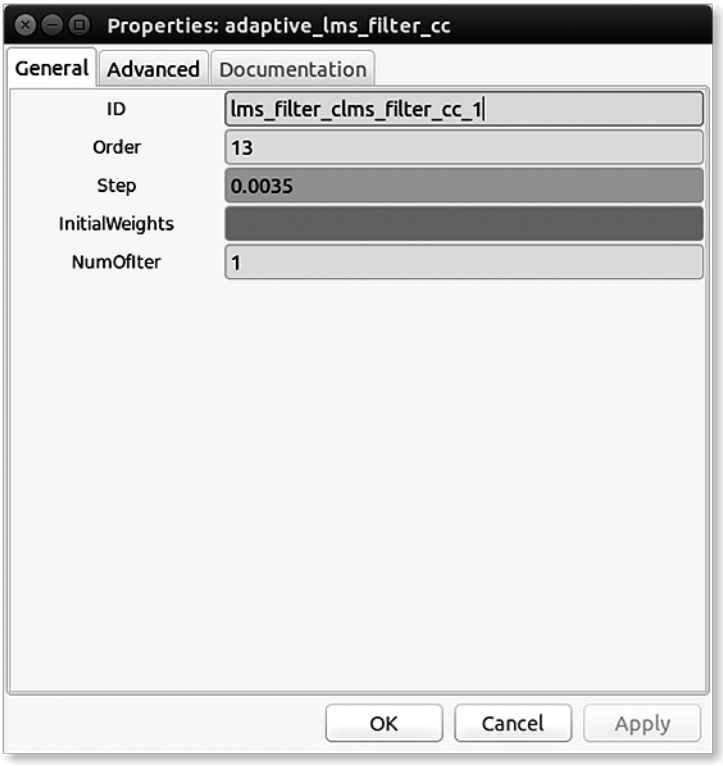
2.2.2 Szablon projektu

Środowisko GnuRadio dostarcza szereg narzędzi wspomagających pracę w rozwijaniu własnych modułów tak, aby były kompatybilne z już istniejącymi. Jednym z nich jest `gr_modtool`, który został wykorzystany do stworzenia struktury plików bloku filtra adaptacyjnego. Dzięki temu nasz blok będzie w stanie łączyć się z innymi, przekazywać i odbierać sygnały oraz będzie rozumiał polecenia wydawane przez środowisko w którym zostanie umieszczony. W projekt włącza się katalog z testami jednostkowymi, sprawdzającymi elementarne funkcjonalności bloku. Dobrze zaimplementowane i przemyślane testy jednostkowe informują o wykrytych błędach powstających podczas tworzenia algorytmu. Oprócz testów w projekcie tworzy się plik `.xml`, który określa reprezentację bloku w środowisku graficznym `gnuradio-companion`. W zależności od wybranego języka programowania, również plik z kodem źródłowym bloku.

2.2.3 Klasa obiektu bloku

Serce modułu odpowiedzialnego za przetwarzanie wejściowych strumieni danych znajduje się w pliku `adaptive_lms_filter_cc.py`. W tym miejscu zdefiniowana jest klasa bloku wraz z jej zmiennymi i funkcjami.

adaptive_lms_filter_cc
-input_items -output_items -weights -order -step -num_of_iter
+work() +consume() +set_history()



Rysunek 2.1: Panel parametrów bloku

Funkcja `work` uruchamiana cykliczne przez proces `scheduler` zawiera instrukcje odpowiedzialne za całe przetwarzanie wewnątrz bloku. Jest miejscem w którym należy umieścić implementację filtra. Pobiera z wejścia nadchodzące elementy i na ich postawie oblicza produkty, które kieruje na wyjście.

Należy zauważyć, że w niektórych przypadkach do obliczenia wartości próbki wyjściowej potrzeba więcej niż jedną próbkę wejściową. Jest tak w przypadku filtracji, gdzie filtr o długości N potrzebuje $N-1$ próbek minionych. Konieczne jest stworzenie bufora z którego można będzie pobierać te wartości. Funkcja `set_history` tworzy miejsce do przechowywania wcześniejszych próbek w pamięci, a jako parametr przyjmuje długość tego bufora.

2.2.4 Inicjalizacja

Następuje zawsze po uruchomieniu projektu. W każdym bloku (obiekcie) na diagramie tworzą się instancje do których przypisuje się wartości początkowe. Parametry mogą być dostosowywane przez użytkownika na etapie projektowania (w czasie przed uruchomieniem), lub zmieniane w trakcie wykonywania programu. Zmiana parametrów w biegu wiąże się z ponowną inicjalizacją bloku i może prowadzić do utraty danych przechowywanych wewnątrz tego obiektu. W środowisku `gnuradio-companion` okno właściwości pojawia się po podwójnym kliknięciu na blok. Zamiast stałej wartości parametru można przypisać referencję do jednego z dostępnych elementów sterujących np. suwaka lub pokrętła. Dzięki temu dostosowywanie bloku możliwe jest z poziomu graficznego panelu użytkownika (R).

Projekt pozwala na sterowanie dwoma parametrami podczas pracy filtra:

1. `order` - rząd $M = \{5, 6, ..., 30\}$

2. `step` - krok adaptacji $\mu \in [0.0002, 0.005]$

2.2.5 Implementacja

Nieodłączną częścią modułu wykorzystywaną przez `gnuradio-companion` jest plik `adaptive_lms_filter_cc.xml`, który określa parametry wejściowe bloku, układ doprowadzeń i wyprowadzeń; gwarantuje kompatybilność połączeń z innymi blokami przez definiowanie typów danych dla wszystkich wejść, wyjść i parametrów. Pozwala również ustawić właściwości istotne dla poprawnego kategoryzowania w zasobach bibliotek.

Poniżej fragment struktury pliku `.xml`:

```
<?xml version="1.0"?>
<block>
  <name>adaptive_lms_filter_cc</name>
  <key>lms_filter_adaptive_lms_filter_cc</key>
  <category>[lms_filter]</category>
  <import>import lms_filter</import>
  <make>lms_filter.adaptive_lms_filter_cc
  ($order, $step, $weights $num_of_iter)</make>

  <param>
    <name>Order</name>
    <key>order</key>
    <type>int</type>
  </param>

  <param>
    <name>Step</name>
    <key>step</key>
    <type>float</type>
  </param>

  <sink>
    <name>reference</name>
    <type>complex</type>
  </sink>

  <source>
    <name>out</name>
    <type>complex</type>
  </source>
```

Nazwy parametrów zostały dobrane tak, aby blok dobrze integrował się ze środowiskiem i mógł zostać powtórnie wykorzystany w projektach innych użytkowników. Stąd język angielski, jako wspólny dla inżynierów na całym świecie.

Algorytm Aby zwiększyć efektywność i wygodę działań algebraicznych na macierzach, zastosowano narzędzia `numpy` będące rozszerzeniem języka Python.

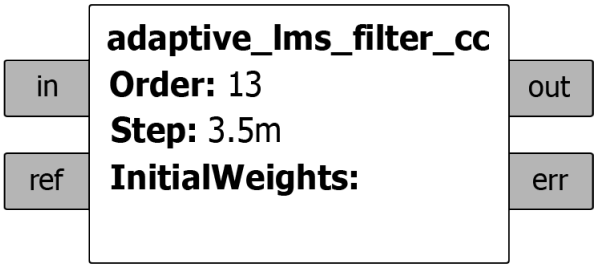
Algorytm pobiera z dwóch wejść próbki sygnałów i zapisuje je do bufora o wymiarze $b = 2 \times M$, którego każdy wiersz to jeden sygnał, a każda kolumna to kolejna próbka tego sygnału. Filtr odczytuje z bufora, wykonuje operacje algebraiczne, adaptacyjnie zmieniając swoje współczynniki. Odfiltrowany sygnał przekierowywany jest na wyście, a z bufora wymazywane są próbki b_{1M}, b_{2M}.

Require: $w \leftarrow w_0 \leftarrow [0, 0, ..., 0]^T$,
  buffer $\leftarrow$ SET HISTORY$(2 \times M)$
  while Program running **do**
    repeat buffer $\leftarrow$ CONSUME SAMPLES(reference input, noisy signal input)
    until buffer length $<$ filter order
    FILTER(buffer)
    REMOVE FROM BUFFER(oldest x, d)
  end while
  function FILTER(samples)
    for all $x, d \leftarrow$ samples **do**
      $y \leftarrow x \cdot w$
      $e \leftarrow d - y$
      $w \leftarrow w \cdot step \cdot x \cdot e$
      $y \leftarrow x \cdot w$
      return y, e
    end for
  end function
  function CONSUME SAMPLES(reference input, noisy signal input)
    $x \leftarrow$ reference input
    $d \leftarrow$ noisy signal input
    buffer $\leftarrow x, d$
    return buffer
  end function

2.2.6 Instalacja

Kiedy wszystkie wymagane funkcjonalności zapisane w kodzie pomyślnie przechodzą etap testowania, implementację uznaje się za zakończoną. Moduł należy dołączyć do bibliotek GnuRadio, aby jego graficzna reprezentacja znalazła się w środowisku `gnuradio-companion`, a sam blok mógł zostać wykorzystany jako element w systemie odbiornika.

Proces instalacji nadzoruje narzędzie `cmake`, które wykonuje instrukcje z pliku `Makefile` powstałego podczas tworzenia szablonu projektu. Gotowy blok przedstawia Rysunek 2.2.



Rysunek 2.2: Reprezentacja graficzna bloku w `gnuradio-companion`

ROZDZIAŁ 3

Modelowanie warunków propagacji

3.1 Symulowanie zakłóceń

3.1.1 Źródła szumów

Szum w systemach telekomunikacyjnych można podzielić na dwie główne kategorie w zależności od ich źródła. Szum generowany przez elementy układu takie jak rezystory lub wzmacniacze aktywne zwany jest szumem wewnętrznym. Drugą kategorią,

3.1.2 GnuRadio

3.1.3 Fala ciągła

3.1.4 Szum addytywny

3.1.5 Wielodrogowość

ROZDZIAŁ 4

ROZDZIAŁ 5

Podsumowanie

ROZDZIAŁ 6

Zakończenie

Bibliografia

[1] Simon Haykin. *Adaptive Filter Theory (3rd Ed.)*. Prentice-Hall, Inc., Upper Saddle River, NJ, USA, 1996.

[2] Torsten Schoen and Co Author. Ten things you better not say to your wife. *Optimizing Husbands*, 21:85–91, 2013.