Politechnika Wrocławska
Wydział Elektroniki
Sprawozdanie z laboratoriów NiDUC2

Transmisja w systemie FEC (Forward Error Correctioin)

Autorzy:
Alicja Wróbel, 238894
Justyna Skalska, 225942
Dariusz Tomaszewski, 235565

wtorek TP, 15:15
dr inż. Jacek Jarnicki

15 czerwca 2018

Spis treści

Rozdział 1

Wstęp

1.1 Opis FEC

FEC, czyli Forward Error Correctioin to technika dodawania nadmiarowych bitów do transmisji informacjicyfrowej. Umożliwia ona detekcję błędów powstałych w wyniku zakłóceń, a także na ich całkowitą lub częściową korekcję. Dzięki jej zastosowaniu nie trzeb korzysać z kanału zwrotnego, aby prosić o ponowne przesłanie uszkodzonych informacji.

1.2 Zastosowania

Kodowanie korekcyjne jest wykorzystywane wtedy kiedy retransmisja informacji jest kłopotliwa, kosztowna lub niemożliwa. Przykładem zastosowania może być użycie w warstwie fizycznej oraz warstwie łącza danych, gdzie sprawdzane i poprawiane są błędnie przesłane bity w sieci WAN. Jest to wykorzystywane, aby zapewnić odebranie przez warstwę przez warstwę protokołu aplikacji pakietów wolnych od błędów. Kodowanie jest także używane w transmisji radiowej, gdzie w większości przypadków korekta błędów bitowych może być zaimplementowana za pomocą kodowania FEC. Ma ono dobry czas rzeczywisty i nie wymaga informacji zwrotnej, co jest odpowiednie dla komunikacji bezprzewodowej.

1.3 Typy kodowania FEC

W rodzinie kodów FEC istnieje wiele różnych kodów korekcji błędów, które używane są w zależnośći od sytuacji. Oto kilka szeroko wykorzystywanych kodów:

- Reeda-Solomona
- Golay'a
- potrajanie bitów
- Hamminga
- BCH (Bose–Chaudhuri–Hocquenghem)
- cykliczny kod nadmiarowy
- kod splotowy

Rozdział 2

Cel i założenia projektu

Głównym celem projektu było napisanie programu ukazującego poszczególne etapy transmisji w systemie FEC, a także ich działanie.
Symulator systemu miał składać się z fragmentów odpowiedzialnych za:

- generację danych przewidzianych do transmisji
- kodowanie wysyłanych danych w nadajniku
- zmianę niektórych danych w wyniku zakłóceń w kanale transmisyjnym
- dekodowanie odebranych danych w odbiorniku

Analiza statystyczna projektu miała polegać na napisaniu programów rejestrujących i analizujących parametry statystyczne transmisji takie jak:

- obliczanie BER dla różnych parametrów systemu
- wyznaczanie rozkładu błędów transmisji w czasie
- wyznaczanie parametrów opisujących efektywność transmisji

Symulacja FEC wymagała napisania dwóch programów, z których każdy dostarczał wyniki na temat jednego sposobu kodowania danych.
Pierwszy program polegał na napisaniu kodera potrajającego każdy bit. Dzięki temu istniało mniejsze ryzyko odebrania niepoprawnej wiadomości po jej odkodowaniu. Polegało ono na ustaleniu czy w ciągu trzech bitów znajduje się więcej 1 lub 0. Dzięki temu, że jest to liczba nieparzysta nie istniała możliwość kiedy system nie był w stanie stwierdzić jaką cyfrę otrzymał. Dzięki temu wiadomość mogła zostać odtworzona pomimo zasymulowanych w kanale zakłóceń. Przekłamania zostały spoodowane losowaniem wartośći z rozkładu Gaussa przy ustalonym prawdopodobieństwie wystąpienia błędu.

Kolejne zadanie wymagało napisania kodera oraz dekodera BCH (Bose–Chaudhuri–-Hocquenghem). Zdolność korekcyjna tego kodowanie jest większa niż tego wykorzystanego w pierwszym zadaniu. Zależy ona od wybranych parametrów bitów danych k oraz bitów korekcyjnych n-k, które przedstawia tablica poniżej *(Rysunek 2.1)*. Przekłamania zostały spoodowane losowaniem wartośći z rozkładu Gaussa przy ustalonym prawdopodobieństwie wystąpienia błędu.

m	*n*	*k*	*t*	*m*	*n*	*k*	*t*	*m*	*n*	*k*	*t*	*m*	*n*	*k*	*t*
3	7	4	1	6	63	10	13	7	127	15	27	8	255	123	19
4	15	11	1			7	15			8	31			115	21
		7	2	7	127	120	1	8	255	247	1			107	22
		5	3			113	2			239	2			99	23
5	31	26	1			106	3			231	3			91	25
		21	2			99	4			223	4			87	26
		16	3			92	5			215	5			79	27
		11	5			85	6			207	6			71	29
		6	7			78	7			199	7			63	30
6	63	57	1			71	9			191	8			55	31
		51	2			64	10			187	9			47	42
		45	3			57	11			179	10			45	43
		39	4			50	13			171	11			37	45
		36	5			43	14			163	12			29	47
		30	6			36	15			155	13			21	55
		24	7			29	21			147	14			13	59
		18	10			22	23			139	15			9	63
		16	11							131	18				

Rysunek 2.1: Parametry kodów BCH

Zadanie projektowe polegało na wykonaniu wielokrotnej symulacji, aby uzyskać pary *(n, k)* wraz z odpowiadającymi im parametrami BER - Bit Error Rate oraz E - Efficency. Prawdopodobieństwo przekłamania jest stałym parametrem, a badane kody BCH odpowiadają n = {31, 63, 127, 255}.

Rozdział 3

Kody źródłowe

3.1 Potrajanie bitów

3.1.1 Generator ciągu bitów

```
function wynik = generator(ilosc_bitow) %ilosc_bitow -
    ilosc_bitow do wygenerowania
  wynik=rand(1,ilosc_bitow); %Losowanie liczb w zakresie
     0.00-1.00
  wynik=round(wynik); %Zaokraglanie do calkowitych (0 i 1)
end
```

3.1.2 Koder

```
function wynik = koder(A) %A - wektor bitow
  wynik=repmat(A,3,1); %Potrajanie bitow (forma macierzy)
  wynik=wynik(:)'; %Odczytuje kolejno elementy macierzy i
     zapisuje je do wektora (' = transpose)
end
```

3.1.3 Kanał transmisyjny z zakłóceniami

```
function wynik = kanal(A,P) %A - wektor bitow, P - prawdop.
    przeklamania
  wynik = A; %Wpisz wektor A jako wynikowy (dalsze operacje
     wykonywane beda na wynikowym)
  r = rand(1,length(wynik)); %Utworzenie wektora losowych
     wartosci o dl. = dl. wektora wynik
  for i=1:1:length(wynik) %Dla kazdego z elem. wek. wykonaj:
    if r(i) < P %Jesli elem. wektora losowych wartosci jest
       mniejszy od P:
      wynik(i) = ~wynik(i); %Zrob negacje elementu w A
    end
  end
end
```

3.1.4 Dekoder

```
function wynik = dekoder(A) %A - wektor bitow potrojonych
  w = vec2mat(A,3); %Stworzenie macierzy z kolejnych trojek
     z wektora A
  wynik = zeros(0); %Inicjalizacja wektora i wpisanie go
     jako wynikowego
  for i=1 : 1 : length(A)/3 %Dla kazdej kolumny w macierzy "
     w" wykonaj:
    ilosc_zer = 0; %Wyzeruj stan
    ilosc_jedynek = 0; %Wyzeruj stan

    for j=1 : 1 : 3 %W wybranej kolumnie macierzy zliczaj:
      if w(i,j) == 0 %Jezeli element = 0
        ilosc_zer=ilosc_zer+1; %Policz kolejne zero
      else %W przeciwnym razie:
        ilosc_jedynek=ilosc_jedynek+1; %Policz kolejna
           jedynke
      end
    end

    if ilosc_jedynek > ilosc_zer %Jesli jedynek jest wiecej:
      wynik = [wynik, 1]; %Do wektora wynikowego dopisz bit
         1
    else %W przeciwnym razie:
      wynik = [wynik, 0]; %Do wektora wynikowego dopisz bit
         0
    end
  end
end
```

3.1.5 BER

```
function wynik = ber(bity_nadajnika, bity_odbiornika)
%bity_nadajnika - wektor bitow po stronie nadawczej
%bity_odbiornika - wektor bitow po stronie odbiorczej
  wektor_bledow = xor(bity_nadajnika, bity_odbiornika); %
     Stworz wektor z informacja o przeklamaniach
  wynik = nnz(wektor_bledow(1,:)==1)/length(bity_nadajnika);
     %Oblicz: ilosc_przeklaman / dlugosc_ciagu_bitow
end
```

3.1.6 Główny kod programu

```
%Ilosc pomiarow do wykonania (odwrotnosc kroku
    prawdopodobienstwa przeklamania):
ilosc_pomiarow = 100;
%Ilosc bitow do wygenerowania przed zakodowaniem:
ilosc_bitow = 1000;
wektor_ber = zeros(0); %Inicjalizacja wektora BER

for i=linspace(0,1,ilosc_pomiarow)
   wygenerowane_bity = generator(ilosc_bitow); %Generowanie
      ciagu bitow
   %disp(wygenerowane_bity);
   zakodowane_bity = koder(wygenerowane_bity); %Kodowanie
      ciagu bitow technika FEC
   %disp(zakodowane_bity);
   odebrane_bity = kanal(zakodowane_bity, i); %Wyslanie ciagu
       bitow do odbiornika poprzez kanal
   %disp(odebrane_bity);
   zdekodowane_bity = dekoder(odebrane_bity); %Dekodowanie
      ciagu bitow zakodowanego technika FEC
   %disp(zdekodowane_bity);
   wektor_ber = [wektor_ber, ber(wygenerowane_bity,
      zdekodowane_bity)]; %Zapisanie wyniku BER
end

plot(linspace(0,1,ilosc_pomiarow), wektor_ber); %Generowanie
    wykresu
title(sprintf("Wykres dla %d pomiarow po %d bitow",
   ilosc_pomiarow,ilosc_bitow));
xlabel("Prawdopodobienstwo bledu");
ylabel("BER");
grid on;
```

3.2 Kodowanie BCH

3.2.1 Kanał transmisyjny z zakłóceniami

```
function noisecode = disrupt(code,n,k,pp)
    noise_number = floor(n*k*pp); % Ilosc zaklocen
    noise_num_row = randi(k,1,noise_number);
    noise_num_col = randi(n,1,noise_number);
    noise_matrix = zeros(k,n);
    for i=1:noise_number
        noise_matrix(noise_num_row(i),noise_num_col(i)) = 1;
    end
    noisecode = code;
    for i = 1:k
        for j = 1:n
            if(noise_matrix(i,j) == 1)
                %noisecode(i,j) = ~noisecode(i,j);
                if(noisecode(i,j) == 1)
                    noisecode(i,j) = 0;
                else
                    noisecode(i,j) = 1;
                end
            end
        end
    end
end
```

3.2.2 Główny kod programu

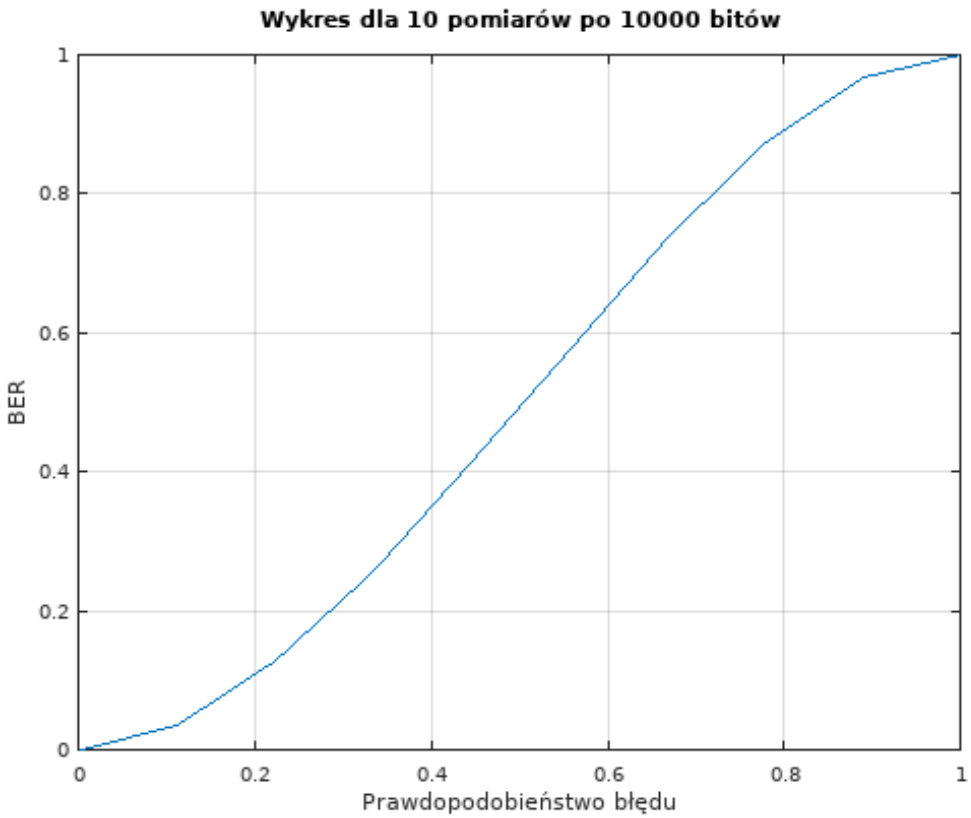
```
clear all;
tests = 5; % ilosc testow
Pp = 0.05; % prawd. przeklamania
m = 5; % od ktorego "m" zaczynamy

file=fopen('m5678.txt','at');
fprintf(file,'K;N;BER;E\n');
for test=1:tests
    while(m >= 5 && m <=8)
        if m==5 %n=31
            k=[26 21 16 11 6];
            t=[1 2 3 5 7];
        end
        if m==6 %n=63
            k=[57 51 45 39 36 30 24 18 16 10 7];
            t=[1 2 3 4 5 6 7 10 11 13 15];
        end
        if m==7 %n=127
            k=[120 113 106 99 92 85 78 71 64 57 50 43 36 29
               22 15 8];
            t=[1 2 3 4 5 6 7 9 10 11 13 14 15 21 23 27 31];
        end
        if m==8 %n=255
            k=[247 239 231 223 215 207 199 191 187 179 171
               163 155 147 139 131 123 115 107 99 91 87 79
               71 63 55 47 45 37 29 21 13 9];
            t=[1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 18 19 21
               22 23 25 26 27 29 30 31 42 43 45 47 55 59
               63];
        end
        n=2^m-1;
        for i=1:numel(k)

            % Generowanie
            msg = gf(randi(1,k(i)));
            % Generowanie wielomianu
            [genpoly, t(i)] = bchgenpoly(n,k(i));

            % Kodowanie
            code = bchenc(msg,n,k(i));

            % Dodawanie zaklocen
            noisecode = disrupt(code,n,k(i),Pp);

            % Dekodowanie
            [newmsg,err,corrected_code] = bchdec(noisecode,n
                ,k(i));
```

```
            BER_one(i,test) = numel(find(msg ~= newmsg))/(k(
                i)*n); % BER
            E_one(i,test) = (k(i)^2-numel(find(msg ~= newmsg
                )))/(k(i)*n); % Efektywnosc
        end
        avrBER = mean(BER_one,2);
        avrE = mean(E_one,2);
        for i = 1: numel(avrE)
            fprintf(file,'%d;%d;%f;%f\n',k(i),n,avrBER(i),
                avrE(i)); % Wypisanie do pliku
        end
        disp('m='); disp(m); disp(' zakonczone!\n');
        m=m+1; % Kolejne "m"
    end
end
fclose(file);
%%
clear all;
uiimport('-file');
%%
scatter(E,BER,10,'filled');
xlabel('E [%]');
ylabel('BER [%]');
title("Obszar Pareto");
grid on;
K_string = char(K);
N_string = char(N);
points = strcat('  (', N_string, ',' , K_string, ')');
t = text(E,BER,points,'FontSize',6);
fclose('all');
```

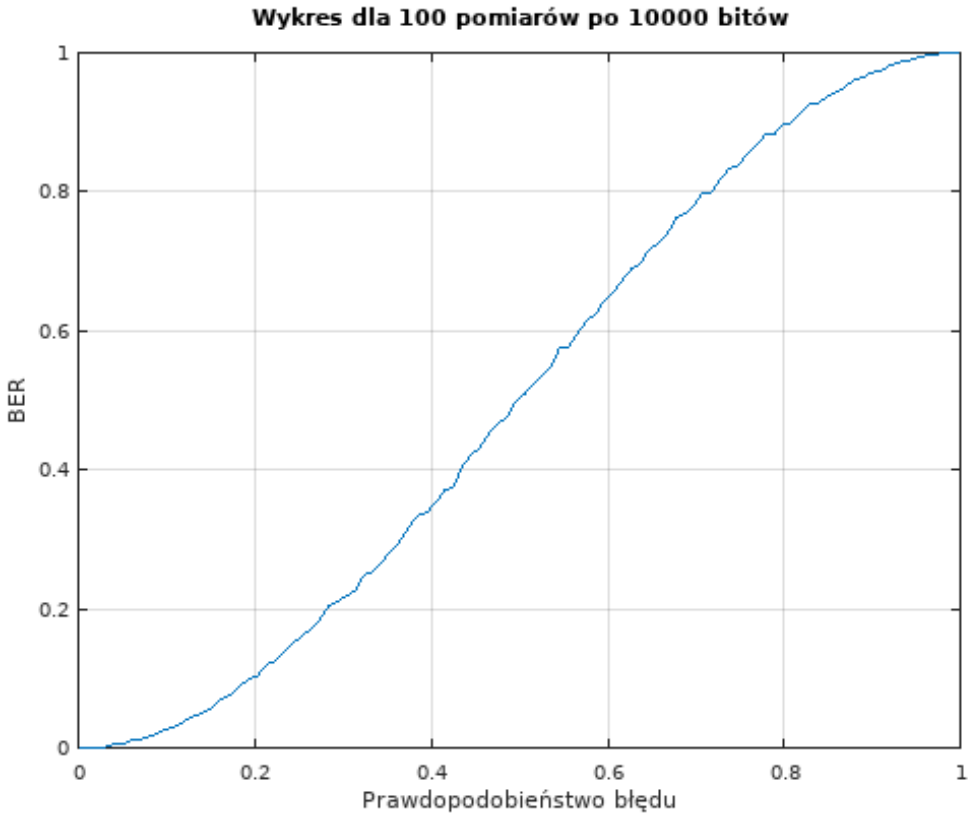
Rozdział 4

Wyniki eksperymentu

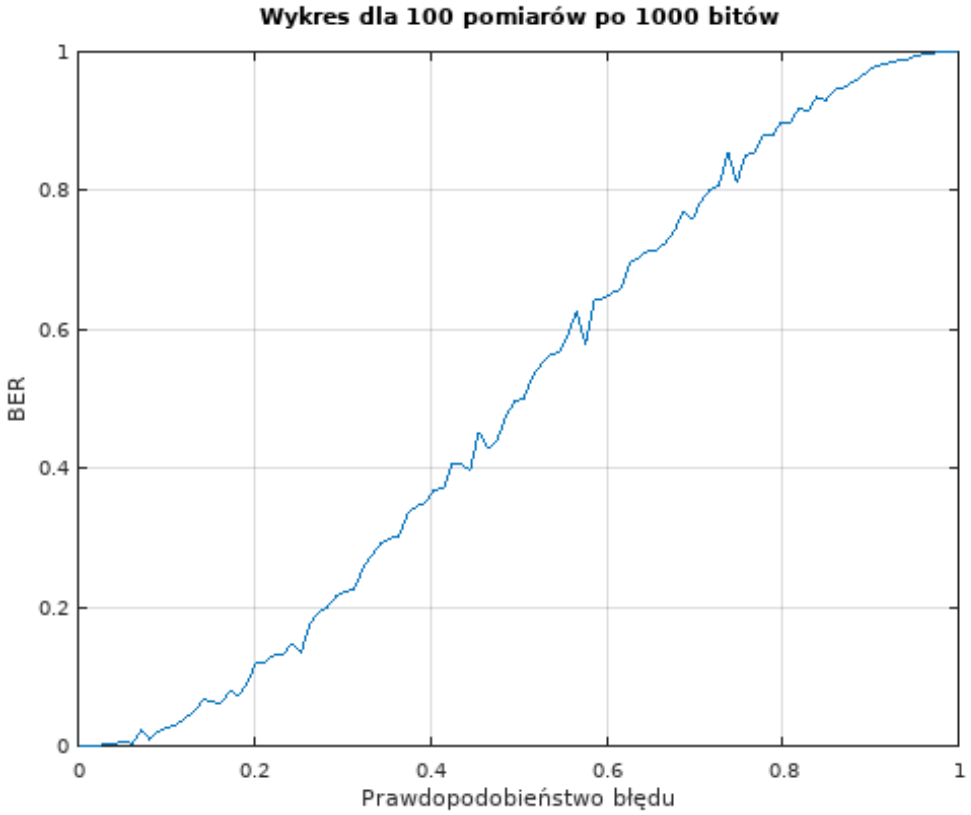
4.1 Potrajanie bitów



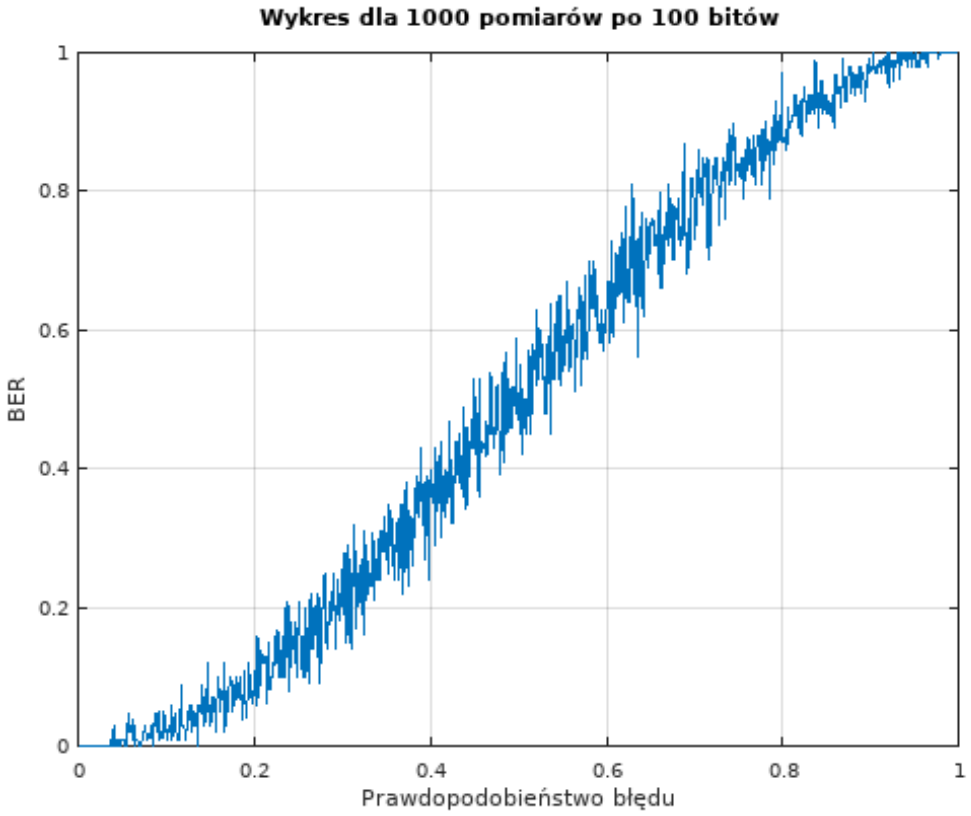
Rysunek 4.1: Liczba błędnie przesłanych pakietów w zależności od prawdopodobieństwa przekłamania w kanale transmisyjnym dla 10 pomiarów po 10000 bitów.



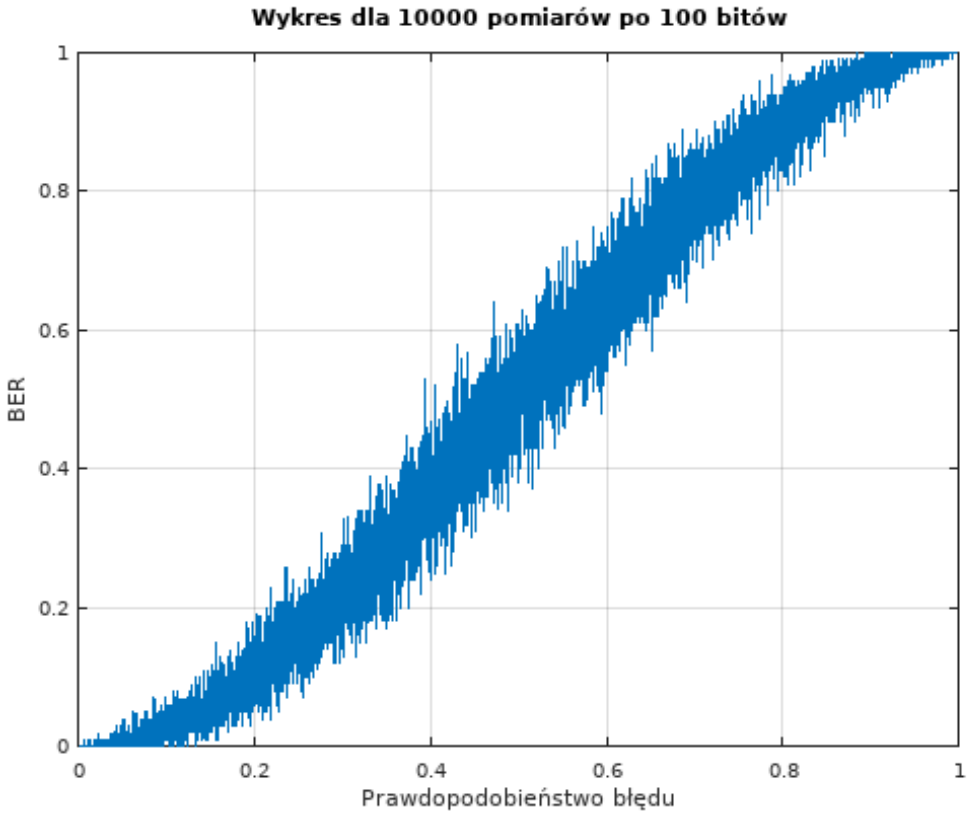
Rysunek 4.2: Liczba błędnie przesłanych pakietów w zależności od prawdopodobieństwa przekłamania w kanale transmisyjnym dla 100 pomiarów po 10000 bitów.



Rysunek 4.3: Liczba błędnie przesłanych pakietów w zależności od prawdopodobieństwa przekłamania w kanale transmisyjnym dla 100 pomiarów po 1000 bitów.



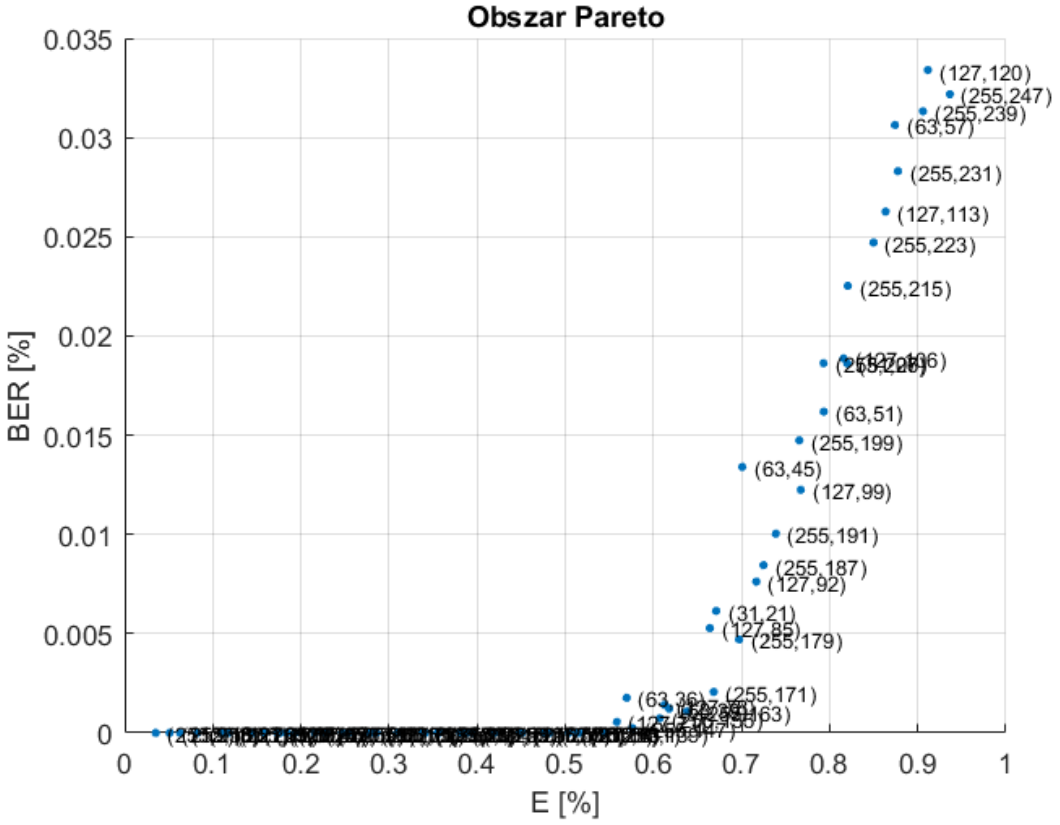
Rysunek 4.4: Liczba błędnie przesłanych pakietów w zależności od prawdopodobieństwa przekłamania w kanale transmisyjnym dla 1000 pomiarów po 100 bitów.



Rysunek 4.5: Liczba błędnie przesłanych pakietów w zależności od prawdopodobieństwa przekłamania w kanale transmisyjnym dla 10000 pomiarów po 100 bitów.

4.2 Kodowanie BCH

Zadanie projektowe polegało na zasymulowaniu korekcji dla znanych kodów BCH, przy n = {31, 63, 127, 255} oraz prawdopodobieństwie przekłamania wynoszącym 0.03. Nie dia się jednoznacznie określić jaki kod BCH jest najbardziej optymalny dla zadanej wartości przekłamania, zaznaczyliśmy, więc na *Rysunek 4.6* najlepsze kody BCH dla naszego kanału transmisyjneg, jest to obszar Pareto.



Rysunek 4.6: Obszar Pareto dla prawdopodobieństwa przekłamania = 0.03

Rozdział 5

Wnioski

Bardzo ważnym aspektem przy cyfrowej transmisji danych jest jej niezawodność. Im więcej dobrze odebranych bitów w stosunku do ilości przesłanych tym lepiej. Podczas tworzenia zadania projektowego mogliśmy zobaczyć na czym polega kodowanie i dekodowanie informacji oraz jak duży wpływ na nią może mieć kanał transmisyjny. Dzięki temu wiemy jak ważną rolę przy przesyłaniu danych odgrywa korekcja błędów. Poznaliśmy podstawowe sposoby naprawy źle odebranej informacji.

5.1 Potrajanie bitów

Pierwszym zaimplementowanym przez nas kodowaniem było potrajanie bitów. Jest to jeden z najprostszych sposobów kodowania informacji. Jest on jednak mało skuteczny względem innych kodowań, ponieważ korekcja stosunek ilości błędów do ilości przesłanych bitów jest większa niż w pozostałych kodowaniach. Ten elementarny sposób może zostać wykorzystany, gdy wymagania projektu nie oczekują dużej korekcji błędów przesłanej informacji.

5.2 Kodowanie BCH

Drugie zaimplementowane przez nas kodowanie cechuje się znacznie większą korekcją błędów w stosunku do poprzednika. Wachlarz zastosowań może być szerszy, ponieważ jesteśmy w stanie poprawić więcej źle odebranych bitów informacji. Kodowanie to wymaga podania parametrów n - długości słowa kodowego oraz k - liczby bitów wiadomości w słowie kodowym. Od nich właśnie zależy zdolność korekcji błędów. Dzięki zadaniu projektowemu mogliśmy nauczyć się jak właściwie zaznaczyć obszar Pareto oraz dzięki temu poprawnie dobrać wartości obu parametrów do wymagań projektowych.

Bibliografia

[1] https://www.techopedia.com/definition/824/forward-error-correction-fec

[2] https://ieeexplore.ieee.org/document/6201904/

[3] https://www.mathworks.com/help/comm/ref/bchenc.html