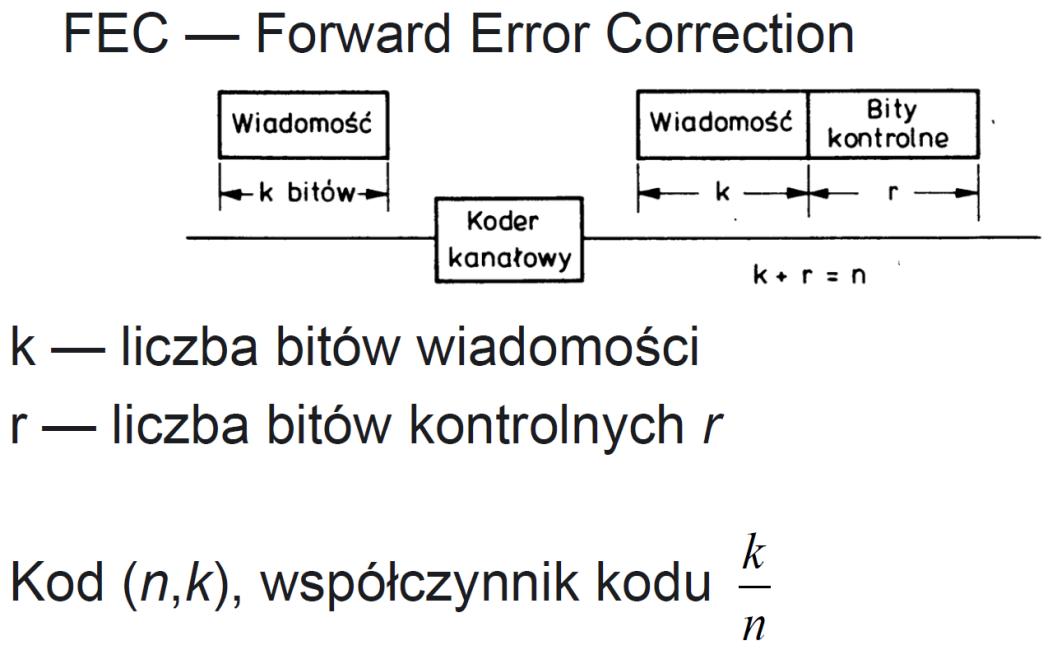
Modulacja i Kodowanie

Labolatorium

Kodowanie Kanałowe – Kody Hamminga

Kody Hamminga należą do grupy kodów korekcyjnych, ich celem jest detekcja I ewentualnie poprawianie błędów. Nazwa tego kody pochodzi od nazwiska twórcy, czyli R. W. Hamming, pracownika laboratoriów Bella. Sam kod został opracowany ok 1950 roku. Kody Hamminga należą ponadto do grupy kodów blokowych, to znaczy, że każdorazowo kodowany jest blok wiadomości o określonej długości, zawierający oprócz bitów informacyjnych także bity kontrolne. W tym przypadku są to bity parzystości, wyliczane z odpowiednich bitów informacyjnych. Po odebraniu takiego bloku danych (dane plus bity kontrolne) możliwa jest jego walidacja pod kątem błędów.

Ogólnie *Kodowanie Korekcyjne* (FEC - Forward Error Correction) - technika dodawania nadmiarowości do transmitowanych cyfrowo informacji. Umożliwia całkowitą lub częściową detekcję i korekcję błędów powstałych w wyniku zakłóceń (Rys. 1).



Rys.1 Ogólny schemat systemów korekcji błędów typu FEC – Forward Error Correction

1. Kod Hamminga (7,4)

W przypadku kodów Hamminga bity kontrolne dodawane są na określonych pozycjach, ogólnie jest to pierwszy bity, a następnie bity będą potęgami liczy 2. Dla kodu Hamming (7,4) liczba bitów informacyjnych wynosi 4, liczba bitów kontrolnych 3, a całkowita długość zakodowanego bloku wynosi 7. Bity kontrolne to bity parzystości, które wyznaczane są dla odpowiednich bitów informacyjnych. Ilustruje to rys. 2:

Bit position	1	2	3	4	5	6	7
	P_1	P_2	1	P_4	1	0	0

Rys. 2: Konfiguracja bitów kontrolnych dla kodu Hamming (7,4)

Natomiast poszczególne bity parzystości są wyznaczone w następujący sposób:

$$b_4 = b_5 + b_6 + b_7$$
$$b_2 = b_3 + b_6 + b_7$$
$$b_1 = b_3 + b_5 + b_7$$

gdzie:
b_n jest sumą jedynek z poszczególnych bitów.

(bit parzystości – bit kontrolny, który przyjmuje taką wartości, aby ciąg złożony z niego samego oraz pewnego ciągu bitów informacyjnych posiadał parzysta liczbę jedynek).

Przykład:

Wiadomość do zakodowania:
1010

Pozycja bitu:	p_1	p_2	p_3	p_4	p_5	p_6	p_7
wiadomość:			*1*		*0*	*1*	*0*
Bity kontrolne:	*1*	*1*		*1*			
Zakodowana wiadomość:	*1*	*1*	*1*	*1*	*0*	*1*	*0*

7 – całkowita liczba bitów w zadokowanej wiadomości (*n-bits*)
4 – liczba bitów informacyjnych (*k-bits*)
n – k – liczba bitów kontrolnych (*m-bits*)

Kody Hammga, ponieważ są kodami liniowymi, to mogą być również przestawione I kodowane w postaci macierzy. Można wyznaczyć dwie macierze macierz generator G oraz macierz kontroli parzystości H:

$$\mathbf{G} := \begin{pmatrix} 1 & 1 & 0 & 1 \\ 1 & 0 & 1 & 1 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}, \qquad \mathbf{H} := \begin{pmatrix} 1 & 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 1 & 1 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 \end{pmatrix}.$$

wiersze 1, 2, oraz 4 macierzy G są bitami kontrolnymi wiadomości.

Dekodowanie :

$$S = v \cdot H^T$$

gdzie:

S – Syndrom,

v- odebrana (zakodowana) wiadomość,

H^T - transponowana macierz kontroli parzystości.

Jeżeli S <> 0 wtedy wiadomość zawiera błędy

2. Cwiczenia: Kodowanie I dekodowania metodą Hamming(7,4)

```
n = 7;                  % Code length
k = 4;                  % Message Length

msg = [1, 0, 1, 0].'    % Message (in binary representation)


% coding message in Hamming(7,4) code
encData = encode(msg,n,k,'hamming/binary')

% decoding message (compare with example !)
decData = decode(encData,n,k,'hamming/binary')
```

```
% check number of error
numerr = biterr(binMessage,decData)
```

Teraz możemy dodać błędny bit, np. na pozycji 4:

```
encData (4) = ~ encData (4);

% decoding message (compare with example !)
decData = decode(encData,n,k,'hamming/binary')

% check number of error
numerr = biterr(binMessage,decData)
```

Kod z minimalnym dystansem Hamminga *Dmin* może wykryć D_{min}*-1* błędów oraz poprawić *(Dmin-1) / 2* lub *Dmin / 2-1*. *Funkcja 'Decode'* automatycznie próbuje poprawiać błędy.

2.1 Transmisja danych w prostym modelu kanału z szumem gaussowskim.

We try send a text over a noisy-channel without and with error correction.

Przygotowanie źródła danych…

```
n = 15;                 % Code length
k = 11;                 % Message Length

data = dec2bin('Some testing message',k);
```

(im dłuższa wiadomosc tym lepiej….)

```
errors = 0;              % number of errors for protection channel
errorsNP = 0;
            % number of errors for non-protection channel
```

Dodatkowe funkcje:

```
% function for convert from string to number array:

function [ numArray ] = str2NumArray( data,dim )
    dataLength = size(data,dim);
    numArray = zeros(1,dataLength);
    for i = 1 : dataLength
        numArray(i) = str2num(data(i));
    end
end

% simple channel simulator:

% Pe - Error Propability

function [ message ] = simplyChannelModel( message, Pe )

    for i = 1 : size(message,2)
        if (rand <= Pe)
            message(i) = ~message(i);
        end
    end

end
```

Główna pętla (główna funkcja uruchamiająca skrypt):

```
for i = 1 : size(data,1)
    binMessage = str2NumArray(data(i,:), 1);
    encData = encode(binMessage,n,k,'hamming/binary');

    Pe = 0.05;
    encData = simplyChannelModel( encData, Pe );
    dataEncoded(i,:) = encData;
    nonProtectiondata = simplyChannelModel( binMessage, Pe );

    decData = decode(encData,n,k,'hamming/binary');

    numerr = biterr(binMessage,decData);
    numerrNP = biterr(nonProtectiondata,decData);
    if (numerr > 0)
        errors = errors + 1;
    end
    if (numerrNP > 0)
        errorsNP = errorsNP + 1;
    end

end

    BER = errorsNP / size(data,1);
    message = strcat('BER [No Error Correction] = [',num2str(BER),']');
    disp(message);

    BER = errors / size(data,1);
    message = strcat('BER [Error Coorection] = [',num2str(BER),']');
    disp(message);
```

Sprawdź współczynnik BER dla różnych P_e. Porównaj BER dla transmisji z korekcją błędów i bez niej (narysuj wykres obrazujący te różnice).

2.2Transmisja danych z wykorzystaniem modelu kanału AWGN oraz modulacji FSK

Korzystając z poprzedniego ćwiczenia zbuduj model system telekomunikacyjnego opartego o kanał AWGN oraz modulację FSK.

Przygotowanie źródła danych…

```
data = dec2bin('Some testing message',k);

for i = 1 : size(data,1)
    binMessage = str2NumArray(data(i,:), 1);
    encData = encode(binMessage,n,k,'hamming/binary');
    dataEncoded(i,:) = encData;
end


% Preparing of 1-dimensional data stream...
[nrMessages, messageLength] = size(data);
dataTxEC = reshape(dataEncoded, [nrMessages * n, 1]);
```

Modulacja oraz demodulacja FSK:

```
M = 2;          % Modulation order
k = log2(M);    % Bits per symbol
EbNo = 5;       % Eb/No (dB)
Fs = 16;        % Sample rate (Hz)
nsamp = 8;      % Number of samples per symbol
freqsep = 10;   % Frequency separation (Hz)

% Modulation
txsig = fskmod(dataTx,M,freqsep,nsamp,Fs);


% Noisy-Channel-Model
rxSig  = awgn(txsig,EbNo+10*log10(k)-10*log10(nsamp),...
    'measured',[],'dB');


% Demodulation
dataOut = fskdemod(rxSig,M,freqsep,nsamp,Fs);
```

```
% get BER

[num,BER] = biterr(dataTx,dataOut);

BER_theory = berawgn(EbNo,'fsk',M,'noncoherent');
[BER BEREC BER_theory]
```

- Sprawdź jaki będzie współczynnik BER dla różnych $E_b/N_{0,}$ sporządź wykres to obrazujący.
- Porównaj BER dla transmisji z korekcją błędów i bez niej.