情報通信基礎1 第7章 演習問題

Author: Takashi Watanabe
Date: 9 June, 2021

課題の提出締切

本課題の提出締切は 7 月 14 日 (水) 23:59 とする．

課題 1: Octave 上でのリードソロモン符号実装

下記に示した assignment1.m は与えられたデータ列に対してリードソロモン符号化・リードソロモン復号化を実行するプログラムである．ただし，assignment1.m においてリードソロモン符号化・リードソロモン復号化をそれぞれ実行する rsEncoder 関数および rsDecoder 関数は完成していないため，現在のままではエラーが出力される．rsEncoder 関数の入力・出力，rsDecoder 関数の入力・出力，Octave で用意されている各種リードソロモン符号化関数を参考にして [7,3] リードソロモン符号を実行する rsEncoder 関数および rsDecoder 関数を完成させなさい．

1. rsEncoder の入力となっている data は assignment1.m から渡されるシンボル配列 [1 2 4] である．それぞれビット列 001, 010, 100 を表している．m はシンボルあたりのビット数，n は符号語数，k は情報記号数である．prim_poly は原始多項式である．

   - data に対して gf 関数を用いて GF(2^m) のガロア体配列 msg を作成しなさい．
     gf 関数への入力および出力は `https://octave.sourceforge.io/communications/function/gf.html` を参照すること
   - rsgenpoly 関数を用いて [n,k] リードソロモン符号の生成多項式 gp を作成しなさい．rsgenpoly の出力は生成多項式の係数を降順に並べたガロア体配列である．
     rsgenpoly 関数への入力および出力は `https://octave.sourceforge.io/communications/function/rsgenpoly.html` を参照すること
   - リードソロモン符号化器 rsenc を用いて [n,k] リードソロモン符号化を msg に対して実行しなさい．
     rsenc 関数への入力および出力は `https://octave.sourceforge.io/communications/function/rsenc.html` を参照すること

2. rsDecoder の入力となっている encData は rsEncoder から出力された符号語配列である．m はシンボルあたりのビット数，n は符号語数，k は情報記号数である．prim_poly は原始多項式である．

   - rxData に対して gf 関数を用いて GF(2^m) のガロア体配列 msg を作成しなさい．
   - rsgenpoly 関数を用いて [n,k] リードソロモン符号の生成多項式 gp を作成しなさい．rsgenpoly の出力は生成多項式の係数を降順に並べたガロア体配列である．
   - リードソロモン復号化器 rsdec を用いて [n,k] リードソロモン復号化を msg に対して実行しなさい．
     rsdnc 関数への入力および出力は `https://octave.sourceforge.io/communications/function/rsdec.html` を参照すること

3. シンボル配列 [1, 2, 4] に対して原子多項式 prim_poly $= x^3 + x + 1$ にしたがう生成多項式を用いた場合，得られた符号語 encData が正しい結果であることを検証しなさい．また，encData を rsDecoder に対して与えた場合，シンボル配列 [1, 2, 4] が得られることを検証しなさい．

※注意点 1: 実行時に “The ’gf’ function belongs to the communications package from Octave Forge which you have installed but not loaded. To load the package, run ’pkg load communications’ from the Octave prompt.” と出力されることがあります．エラーメッセージが発生した場合はコマンドウィンドウで “pkg load communications” と入力してください．
※注意点 2: コメント部分はあくまで参考として記載しているため，そのままプログラムに書き写す必要はない．ただし，どのようなプログラムを記載していたか想起するために，各自がわかりやすい言葉でコメントを記載していくことを推奨します．
※注意点 3: PDF ファイルからそのままプログラムをコピーした場合，意図しない全角文字や記号が含まれてエラーメッセージが出力される可能性が高いため，各自でエディタ上にプログラムを入力することを推奨する．

```
clear

# RS coding parameters [3, 7]
m = 3; % Number of bits per symbol
n = 2^m-1; % Codeword length
k = 3; # Message length
prim_poly=11; #原始多項式 prime poly = x^3+x+1 = [1 0 1 1] = 11


# Original Data
data = [1 2 4]; # 001 010 100 を送信

# Take RS coding using [n, k] RS code with primitive polynomial of prim_poly
encData = rsEncoder(data, m, n, k, prim_poly);
de_encData = gf2dec(encData,m,prim_poly);

# Take RS decoding  using [n, k] RS code with primitive polynomial of prim_poly
decData = rsDecoder(encData, m, n, k, prim_poly);
de_decData = gf2dec(decData,m,prim_poly);

# Calculate Error
Error = sum(data-de_decData) # Confirm zero
```

ソースコード 1: assignment1.m

```
# rsEncoder: dataに対してリードソロモン符号化を実行
# m: 1シンボルあたりのビット数, n: 符号化後のシンボル数, k: 符号化内のオリジナルシンボル数, prim_poly: 原始多項式
function encData = rsEncoder(data, m, n, k, prim_poly)
# dataから GF(2^m), 原始多項式 prim_poly にしたがうガロア体配列を作成
msg=gf();
# [n,k]リードソロモン符号の生成多項式を作成
gp=rsgenpoly();
# RS coding
encData=rsenc();
```

ソースコード 2: rsEncoder.m

```
# rsDecoder: rxDataに対してリードソロモン復号化を実行
# m: 1シンボルあたりのビット数, n: 符号化後のシンボル数, k: 符号化内のオリジナルシンボル数, prim_poly: 原始多項式
function [decData] = rsDecoder(rxData, m, n, k, prim_poly)
# rxDataから GF(2^m), 原始多項式 prim_poly にしたがうガロア体配列を作成
msg=gf();
# [n,k]リードソロモン符号の生成多項式を作成
gp=rsgenpoly();
# RS decoding
decData=rsdec();
```

ソースコード 3: rsDecoder.m

```
%==========================================================================
%gf2dec.m
%Convert a Galois Field Array into a Decimal Array.
%The calling syntax is:
%    [DecOutput] = gf2dec(GFInput,m,prim_poly)
%Inputs:
%    GFInput: gf Array input
%    m: integer between 1 and 16 used in GF(2^m) array
%    prim_poly: integer representation of the primitive polynomial used by GF
%Outputs:
%    DecOutput: Decimal Array
%==========================================================================
function [DecOutput] = gf2dec(GFInput,m,prim_poly)
[row, column] = size(GFInput);
GFInput = GFInput(:)';% force a row vector
GFRefArray = gf([0:(2^m)-1],m,prim_poly);
for i=1:length(GFInput)
    for k=0:(2^m)-1
        temp = isequal(GFInput(i),GFRefArray(k+1));
        if (temp==1)
            DecOutput(i) = k;
        end
    end
end
DecOutput = reshape(DecOutput, [row, column]);
```

ソースコード 4: gf2dec.m

課題 2: リードソロモン符号導入による誤り率の評価

下記に示した assignment2.m はリードソロモン符号化によるビット誤り率をシミュレートするプログラムである．より具体的には，送信側では一様分布にしたがって発生するシンボル配列に対してリードソロモン符号化を実行する rsEncoder 関数を用いて符号語を取得するとともに，二位相偏移変調 (Binary Phase-Shift Keying: BPSK) を用いて変調した無線信号を加算性白色ガウス雑音 (Additive white Gaussian noise: AWGN) が発生する無線伝送路を介して受信側に送信する．受信側では BPSK にしたがって無線信号を復調するとともにリードソロモン復号化を実行する rsDecoder 関数を用いて受信時のシンボル配列を取得するとともに，送信側のシンボル配列と受信側のシンボル配列からビット誤り率を取得する．

assignment2.m は課題 1 で実装した rsEncoder 関数，rsDecoder 関数，BPSK による変調および復調を実行する modulator 関数，demodulator 関数，AWGN が発生する無線伝送路を再現した channel 関数，ビット誤り率が計算できる error_calc 関数がすでに実装されている．また，リードソロモン符号を利用する場合，利用しない場合をそれぞれシミュレートするプログラムが実装されている．assignment2.m を利用して以下の課題に答えなさい．

1. assignment2.m を実行してリードソロモン符号を利用した場合，リードソロモン符号を利用しない場合によるビット誤り率を示す片対数グラフを取得しなさい．ただし，現在のプログラムは片対数グラフを出力するプログラムが実装されていない．下図に片対数グラフの例を示すので，例を参考にして片対数グラフを出力できるプログラムを実装しなさい．横軸は無線伝送路品質を示す Signal-to-Noise Ratio (SNR)，縦軸は対数表記としたビット誤り率（Bit Error Rate: BER）とする．凡例はリードソロモン符号を使用した場合を Coding，リードソロモン符号を利用しない場合を No Coding としなさい．

2. 評価結果から，伝送路品質が良くなるにつれてリードソロモン符号を利用する場合，利用しない場合とでビット誤り率がどのように変化するか考察しなさい．

3. 現在実装されているリードソロモン符号は [7, 3] リードソロモン符号である．これまでのリードソロモン符号の実装を参考にして [15, 11] リードソロモン符号または [15, 9] リードソロモン符号を実装し，これまでと同様にビット誤り率について考察しなさい．

※注意点 1: シンボル配列 data の送信回数を示す NumData は assignment2.m 上で 10000 回と定めている．これは加算性白色ガウス雑音 (AWGN) の大きさが正規分布にしたがってランダムに増減することから，乱数による影響を軽減するためである．ただし，実行時間が非常に長くなるため，動作チェックのために送信回数を短くしてもよい.（ただし，グラフ取得時は十分な送信回数を確保すること）
※注意点 2: 伝送路品質 SNR は snVec にあるとおり 0〜10dB までとなっている．SNR は値が小さくなればなるほど品質の悪い伝送路であることを示している．グラフ取得時は-10〜20dB など範囲を広げて誤り率を確認してもよい（ただし，範囲を広げるにしたがって実行時間が長くなるため注意すること）.

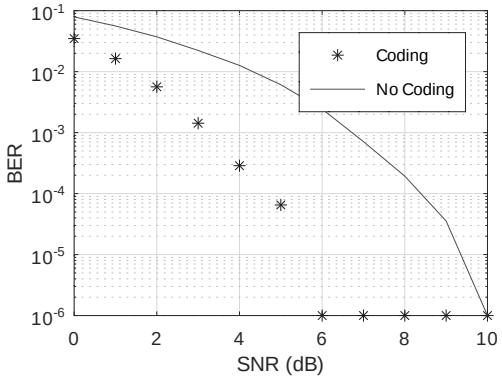

Figura 1: 出力後の片対数グラフの一例

```
clear

# RS coding parameters [3, 7]
# m = 3; % Number of bits per symbol
# n = 2^m-1; % Codeword length
# k = 3; # Message length
# prim_poly=11; %原始多項式 prime poly = x^3+x+1 = [1 0 1 1] = 11,

# Original Data
NumData = 10000; # Represents how many times send m bits
data = randi([0 n], NumData, k); # transmission data (make NumData x m matrix)

# Modulation parameters
M = 2; # modulation order (Binary Phase Shift Keying: BPSK)

# Channel parameters
c_model = 'awgn'; # We support 'awgn', 'fading_posteq', 'fading_preeq', 'error-free'
snVec = (0:10)'; # channel SNR in dB
K = 0; # K-factor in dB: Ratio between the powerin the direct path and the other

# Variable for BER stats
ber_Coding = zeros(1, length(snVec)); # Store BER with RS coding
ber_NoCoding = zeros(1, length(snVec)); # Store BER without RS coding

# RS encoding
encData = rsEncoder(data, m, n, k, prim_poly); # Take RS coding
de_encData = gf2dec(encData,m,prim_poly); # GF(2^3) numbers to decimal numbers

# Modulation
modData = modulator(de_encData,m,M); # modulation with BPSK
for i = 1:length(snVec)
  # Transmission
  rxSig = channel(modData, c_model, snVec(i), K); # Deliver via wireless channels
  # Demodulation
  rxData = demodulator(rxSig,n,M); # demodulate bits from the received symbols
  # RS decoding
  decData = rsDecoder(rxData, m, n, k, prim_poly); # Take RS decoding
  de_decData = gf2dec(decData,m,prim_poly);

  # Calculate BER
  ber_Coding(i) = error_calc(data, de_decData, m); # Calculate BER

### No Coding Results

# Modulate original data
modData = modulator(data,m,M); # modulation for original data with BPSK
for i = 1:length(snVec)
  # Transmission
  rxSig = channel(modData, c_model, snVec(i), K); # Deliver via wireless channels
  # Demodulation
  rxData = demodulator(rxSig,k,M); # demodulate bits from the received symbols

  # Calculate BER without RS coding
  ber_NoCoding(i) = error_calc(data, rxData, m);
end

# Plot the evaluation results (以降を実装してください)
# y軸を対数表記とする片対数グラフを作成する. 横軸はSN, 縦軸は対数表記のBER
# y軸のラベルを設定
# x軸のラベルを設定
# リードソロモン符号を利用する場合, 利用しない場合とで凡例を設定
# グリッド線を表示する
```

ソースコード 5: assignment2.m

```
function modData = modulator(encData, m, M)

[numData, Codewords] = size(encData); # extract number of rows and columns
modData = []; # For results
for i = 1:numData
   r_data = encData(i,:); # extract row vector
   r_mod = [];            # temporal results
   for j = 1:Codewords
        bits = de2bi(r_data(j), m); # decimal to bit representation
        mods = pskmod(bits, M);     # BPSK
        r_mod = [r_mod ; mods];     # store modulalted symbols (to make Codewords x m
    matrix)
   end
   modData = [modData ; r_mod];     # Make matrix of modulated symbols (Codewords*
    numData x m matrix)
end
```

ソースコード 6: modulator.m

```
function demodData = demodulator(transData, n, M)

[numData, Codewords] = size(transData); # extract number of rows and columns
demodData = []; # For results

for i = 1:n:numData
   r_data = transData(i:i+n-1,:);   # Extract n * m matrix to demodulate codeword
   symbols = pskdemod(r_data, M);   # hard decision demodulator
   demods = bi2de(symbols).';       # make row vector with the demodulated symbols
   demodData = [demodData ; demods];  # For making NumData x n matrix
end
```

ソースコード 7: demodulator.m

```
function transData = channel(modData, channel, SN_dB, K)

[row, column] = size(modData); # extract number of rows and columns

# Fading effect according to K-factor
m_channel = sqrt(K/(2*(K+1)));
v_channel = sqrt(1/(2*(K+1)));

Pn = 10.^(-SN_dB/10); # Noise Power
# effective Noise
Noise = complex(randn(row, column).*sqrt(Pn/2), randn(row, column).*sqrt(Pn/2));
# fading effect
amp = abs(v_channel .* complex(randn(row, column), randn(row, column))+m_channel);

# In AWGN channel
if strcmp(channel,'awgn') == 1
    transData = modData + Noise;
# In Fading channel with ZF pre-equalization
elseif strcmp(channel,'fading_preeq') == 1
    transData = amp .* modData + Noise;
# In Fading channel with ZF post-equalization
elseif strcmp(channel,'fading_posteq') == 1
    transData = modData + Noise ./ (amp + 1e-20);
# No noise: for test
elseif strcmp(channel,'error-free') == 1
    transData = modData;
endif
```

ソースコード 8: channel.m

```
function ber = error_calc(data, decData, m)

# vectorize original data
v_data = data(:);
# vectorize decoded data
v_decData = decData(:);

# Obtain the corresponding original bits and decoded bits
orig_bits = de2bi(v_data, m);
dec_bits = de2bi(v_decData, m);

# Calculate number of bit errors
error_bits = abs(orig_bits - dec_bits);

# Obtain bit error rate
ber = sum(error_bits(:)) / length(error_bits(:)) + 1E-6;
```

ソースコード 9: error_calc.m

課題 3: リードソロモン符号導入による画像への影響

下記に示した assignment3.m はリードソロモン符号化を用いて画像を伝送する場合，その誤り率が画像にもたらす影響をシミュレートするプログラムである．より具体的には，送信側では工事員の画像 foreman_mini.jpg から得られるシンボル配列に対してリードソロモン符号化を実行する rsEncoder 関数を用いて符号語を取得するとともに，二位相偏移変調 (BPSK) を用いて変調した無線信号を加算性白色ガウス雑音 (AWGN) が発生する無線伝送路を介して受信側に送信する．受信側では BPSK にしたがって無線信号を復調するとともにリードソロモン復号化を実行する rsDecoder 関数を用いて受信時のシンボル配列を取得するとともに，工事員の画像を復号して image フォルダ内に出力する．

assignment3.m は課題 1 で実装した rsEncoder 関数，rsDecoder 関数，画像を読み書きする imLoader 関数，imDecoder 関数，BPSK による変調および復調を実行する modulator 関数，demodulator 関数，AWGN が発生する無線伝送路を再現した channel 関数，ビット誤り率が計算できる error_calc 関数がすでに実装されている．また，リードソロモン符号を利用する場合，利用しない場合をそれぞれシミュレートするプログラムが実装されている．assignment3.m を利用して以下の課題に答えなさい．

- 出力画像を元にして，リードソロモン符号を利用する場合，リードソロモン符号を利用しない場合とでによるビット誤り率あるいは伝送路品質が伝送する画像にもたらす影響を考察しなさい．
- 自身で用意した画像を用いて assignment3.m を実行しなさい．解像度は時間をかけたくないときは 88×72 画素の解像度，時間をかけられるときは 176×144 画素の解像度の画像を利用するとよい．

```
clear

# RS coding parameters [3, 7]
m = 3; % Number of bits per symbol
n = 2^m-1; % Codeword length
k = 3; # Message length
prim_poly=11; %原始多項式 prime poly = x^3+x+1 = [1 0 1 1] = 11

#load grayscale image
# data: m-bits row vectors for transmission
# h, w: height and width of the image
[data, h, w] = imLoader("foreman_mini.jpg", m, k);

# Modulation parameters
M = 2; # modulation order (Binary Phase Shift Keying: BPSK)

# Channel parameters
c_model = 'awgn'; # We support 'awgn', 'fading_posteq', 'fading_preeq', 'error-free'
snVec = (0:10)'; # channel SNR in dB
K = 0; # K-factor in dB: Ratio between the powerin the direct path and the other

# Variable for BER stats
ber_Coding = zeros(1, length(snVec)); # Store BER with RS coding
ber_NoCoding = zeros(1, length(snVec)); # Store BER without RS coding

# RS encoding
encData = rsEncoder(data, m, n, k, prim_poly); # Take RS coding
de_encData = gf2dec(encData,m,prim_poly); # GF(2^3) numbers to decimal numbers

# Modulation
modData = modulator(de_encData,m,M); # modulation with BPSK
for i = 1:length(snVec)
  # Transmission
  rxSig = channel(modData, c_model, snVec(i), K); # Deliver via wireless channels
  # Demodulation
  rxData = demodulator(rxSig,n,M); # demodulate bits from the received symbols
  # RS decoding
  decData = rsDecoder(rxData, m, n, k, prim_poly); # Take RS decoding
  de_decData = gf2dec(decData,m,prim_poly);
  # Image decoder
  decImage = imDecoder(de_decData,m,h,w);
  imwrite(decImage,strcat("./image/im_coded_sn_",int2str(snVec(i)),".jpg"));
  # Calculate BER
  ber_Coding(i) = error_calc(data, de_decData, m); # Calculate BER
end

### No Coding Results

# Modulate original data
modData = modulator(data,m,M); # modulation for original data with BPSK
for i = 1:length(snVec)
  # Transmission
  rxSig = channel(modData, c_model, snVec(i), K); # Deliver via wireless channels
  # Demodulation
  rxData = demodulator(rxSig,k,M); # demodulate bits from the received symbols
  # Image decoder
  decImage = imDecoder(rxData,m,h,w);
  imwrite(decImage,strcat("./image/im_no_code_sn_",int2str(snVec(i)),".jpg"));
  # Calculate BER without RS coding
  ber_NoCoding(i) = error_calc(data, rxData, m);
end
# Plot the evaluation results
# 課題2と同一の実装でOK
```

ソースコード 10: assignment3.m

```
function [data, h, w] = imLoader(filename, m, k)

[rgb, immap, alpha] = imread(filename); # load color image
im_gray = double(rgb2gray(rgb));  # convert to gray-scale image
[h, w] = size(im_gray);
im_bits = de2bi(im_gray(:), 8);   # obtain 8 bits of the gray-scale image
[N_pixels, bitDepth] = size(im_bits);
N_row = (N_pixels*bitDepth)/m;
im_data = bi2de(reshape(im_bits, [N_row, m]));  # obtain decimal vector with N_row x
    1
data = reshape(im_data, [N_row/k, k]);  # obtain decimal vector with N_row/k x k
```

ソースコード 11: imLoader.m

```
function decImage = imDecoder(decData, m, h, w)

dec_im_data = de2bi(decData(:), m); # convert decimal value into m bits
N_row = (length(decData(:))*m) / 8;
dec_im_bits = reshape(dec_im_data, [N_row, 8]); # convert decoded bits into 8 bits
decImage = reshape(bi2de(dec_im_bits), [h, w]); # convert vector into h x w image
decImage = uint8(decImage); # convert into uint8 value for display
```

ソースコード 12: imDecoder.m