Encode PUCCH format 0

PUCCH format 0

Cấu hình carrier

```
carrier = nrCarrierConfig;
carrier.SubcarrierSpacing = 60;
carrier.CyclicPrefix = 'normal';
carrier.NSlot = 57;
```

Cấu hình PUCCH

```
pucch = nrPUCCH0Config;
pucch.SymbolAllocation = [13 1];
pucch.FrequencyHopping = 'neither';
pucch.GroupHopping = 'disable';
pucch.HoppingID = 512;
pucch.InitialCyclicShift = 6;
```

Data

```
ack = [0; 0];
sr = [1];
uci = {ack,sr};
```

Bắt đầu

```
lenACK = length(uci{1});
lenSR = length(uci{2});
symStart = pucch.SymbolAllocation(1);
nPUCCHSym = pucch.SymbolAllocation(2);
```

Độ dài chuỗi....

```
m = 1;
nRBSC = 12;
mzc = m*12;
```

Tính chuỗi $\overline{r}_{u,v}(n) = e^{j\phi(n)\pi/4}$

- Tính u

fss

```
if isempty(pucch.HoppingID)
    nid = carrier.NCellID;
else
    nid = pucch.HoppingID;
```

```
end
nid
```

```
nid = 512
```

```
fss = mod(nid,30) ;
fss %print
```

```
fss = 2
```

fgh,v

```
fgh = zeros(1,2);
% fgh có 2 giá trị vì có thể sử dụng đến 2 symbol
v = zeros(1,2);
% slot number
nslot = mod(double(carrier.NSlot),carrier.SlotsPerFrame);
switch(pucch.GroupHopping)
    case 'neither'
        % fgh = 0; v = 0;
    case 'enable'
        cinit = floor(nid/30);
        fgh(1,:) = mod((2.^(0:7))*reshape(PRBSGen(cinit,[8*2*nslot 16]),8,[]),30);
        v = 0;
    case 'disable'
        cinit = 2.^5*floor(nid/30) + mod(nid,30);
        v(1,:) = double(PRBSGen(cinit,[2*nslot 2])');
end
fgh %print
```

```
fgh = 1×2
     0     0
```

```
v
```

```
v = 1×2
     0     1
```

u

```
u = mod(fgh+fss,30)
```

```
u = 1×2
     2     2
```

Chuỗi r

```
phi = returnPhiValue(u(1),mzc)
```

```
phi = 12×1
    -3
     3
     3
     1
    -3
```

```
     3
    -1
     1
     3
    -3
     ⋮
```

```
rSeq = exp(1i*phi*pi/4)
```

```
rSeq = 12×1 complex
  -0.7071 - 0.7071i
  -0.7071 + 0.7071i
  -0.7071 + 0.7071i
   0.7071 + 0.7071i
  -0.7071 - 0.7071i
  -0.7071 + 0.7071i
   0.7071 - 0.7071i
   0.7071 + 0.7071i
  -0.7071 + 0.7071i
  -0.7071 - 0.7071i
     ⋮
```

Tính alpha $\alpha = \frac{2\pi}{N_{\text{sc}}^{\text{RB}}} \cdot ((m_0 + m_{\text{cs}} + m_{\text{int}} + n_{\text{cs}}(n_{s,f}^{\mu},\ l + l')) \bmod N_{\text{RB}}^{\text{sc}})$

- Sử dụng cyclic Prefix

```
if strcmpi(carrier.CyclicPrefix,'extended')
    nSlotSymb = 12;
else
    nSlotSymb = 14;
end
nSlotSymb
```

```
nSlotSymb = 14
```

```
nslot
```

```
nslot = 17
```

- Tìm ncs

```
% Giá trị ncs cho tất cả symbol
ncs = (2.^(0:7))*reshape(PRBSGen(nid,nSlotSymb*8*[nslot 1]),8,[])
```

```
ncs = 1×14
   187   136   142   110   130   124   103   254   185     2   198    63   246 ···
```

- Tìm mCS với giá trị của uci

```
cstable = returnMcs(lenACK);
```

```
if lenACK == 0
    mcs = cstable(1,1);
elseif (lenSR == 0) || (sr == 0)
    Uci = comm.internal.utilities.convertBit2Int(ack,lenACK);%% Chuyển chuỗi bit
ACK thành giá trị số nguyên có độ dài lenACK
    mcs = cstable(1,Uci+1);
else
    Uci = comm.internal.utilities.convertBit2Int(ack,lenACK);% same
    mcs = cstable(2,Uci+1);
end

mcs %print
```

```
mcs = 1
```

```
m0 = pucch.InitialCyclicShift;
mint = 0; % default = 0 với interlace = false;
alpha = 2*pi*mod(mint + mcs + m0 + ncs,nRBSC)/nRBSC
```

```
alpha = 1×14
    1.0472    5.7596    2.6180    4.7124    2.6180    5.7596    1.0472    4.7124 ···
```

Chỉ lấy giá trị alpha ứng với Symbol được dùng:

```
realAlpha = alpha(:,symStart+(1:nPUCCHSym))
```

```
realAlpha = 5.2360
```

Output r

```
nIndex = (0:mzc-1)';
lps = exp(1j*nIndex*realAlpha).*repmat(rSeq,1,length(realAlpha))
```

```
lps = 12×1 complex
  -0.7071 - 0.7071i
   0.2588 + 0.9659i
   0.9659 + 0.2588i
  -0.7071 - 0.7071i
   0.9659 - 0.2588i
  -0.9659 - 0.2588i
   0.7071 - 0.7071i
   0.9659 - 0.2588i
   0.9659 + 0.2588i
   0.7071 + 0.7071i
      ⋮
```

Nếu có 2 symbols và có freq hopping

```
if strcmp(pucch.FrequencyHopping,'intraSlot') && (nPUCCHSym ==  2)
    %lấy giá trị u(2) để tìm chuỗi r
    phi2 = returnPhiValue(u(2),mzc);
    rSeq2 =  exp(1i*phi2*pi/4);
    lps2 = exp(1j*nIndex*realAlpha(nPUCCHSym)).*repmat(rSeq2,1,1)
    seq = [lps(:,1);lps2]
else
```

```
    seq = lps(:);
end
seqf = encodePUCCH0dx(carrier,pucch,uci);
% so sánh với hàm có sẵn của MATLAB, có thể bỏ
seqm = nrPUCCH(carrier,pucch,uci);
a = (seqf == seqm)
```

```
a = 12×1 logical array
   1
   1
   1
   1
   1
   1
   1
   1
   1
   1
     ⋮
```

```
figure
% Create stem of seq
stem(seq,"DisplayName","seq");
```

```
Warning: Using only the real component of complex data.
```

```
hold on
% Create stem of seqf
stem(seqf,"DisplayName","seqf");
```

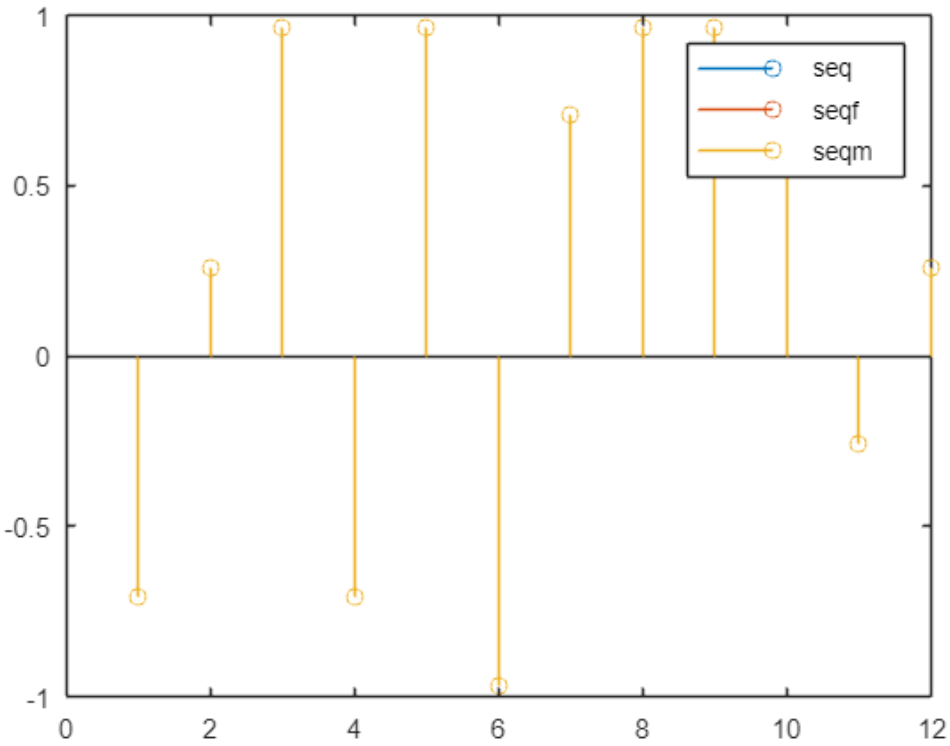
```
Warning: Using only the real component of complex data.
```

```
% Create stem of seqm
stem(seqm,"DisplayName","seqm");
```

```
Warning: Using only the real component of complex data.
```

```
hold off

legend
```



Transmission process....

```
%Đường truyền và giải điều chế OFDM, FFT
```

Demodulation process...

```
% Bao gồm giải điều chế OFDM và hàm nrExtractResources với các chỉ số PUCCH0
% (vị trí trong resource grid) để lấy giá trị của chuỗi.
% Giả sử rằng chúng ta có 5 anten để thu.

numAntenna = 5;
rx_sym = repmat(seq,1,numAntenna);
```

Decoding

Inputs

```
% thông tin về carrier và pucch biết trước
dc_carrier = carrier;
dc_pucch = pucch;
% số bit ack và sr
dc_numOfack = numel(ack)
```

```
dc_numOfack = 2
```

```
dc_numOfsr = numel(sr)
```

```
dc_numOfsr = 1
```

```
% tín hiệu nhận được sau demod
dc_sym = rx_sym;
%tính giá trị ngưỡng, được khuyến nghị bởi MATLAB
thres = 0.49 - 0.07*(pucch.SymbolAllocation(2)==2);
```

Kiểm tra giá trị numofSR

```
nPUCCHsym = pucch.SymbolAllocation(2);
seqlen = 12*nPUCCHsym; % 12 subcarriers.
```

num of sr

```
srOnly = 0;
if (dc_numOfack == 0) && dc_numOfsr
    % SR only transmission
    srOnly = 1; % Flag to indicate if there is only SR transmission
end
```

Main

khởi tạo các tổ hợp

```
allAcks = 2.^dc_numOfack;
allSR = 2.^dc_numOfsr;
c =zeros(allSR,allAcks);
```

năng lượng của sym

```
esym = sum(abs(dc_sym).^2);
```

tạo bảng để so sánh với tất cả trường hợp của uci

```
if ~srOnly
    ackreftable = [0 0 1 1; 0 1 0 1];
else
    ackreftable = false(1,0);
end
srreftable = [0; 1];
```

Kiểm tra các trường hợp

$$R = \frac{\left| \sum_{n=0}^{len} sym[n] * ref[n]^* \right|}{\sqrt{E_{sym} * E_{ref}}}$$

$$E = \sum_{n=0}^{len} |sym[n]^2|$$

```
for srIdx = 1:allSR
    for ackIdx = 1:allAcks
        % sử dụng encode để tìm chuỗi với gía trị ack và sr đang xét
        refsym = encodePUCCH0dx(carrier,pucch,
{ackreftable(:,ackIdx),srreftable(srIdx)});
        if ~isempty(ackreftable)
            mulrefsym = repmat(refsym,1,size(dc_sym,2));
            %power
            emulrefsym = sum(abs(mulrefsym).^2);
            norm = sqrt(emulrefsym.*esym);
            % lưu vào lưới giá trị
            c(srIdx,ackIdx) = mean(abs(sum(dc_sym.*conj(mulrefsym))))./(norm +
eps));% eps for 0 value case
        end
    end
end
```

Lấy giá trị UCI dựa trên max(c) > thres

```
maxval = max(c,[],'all');
if (maxval >= thres)
    [ridx, cidx] = find(c == repmat(maxval,allSR,allAcks));
    resAck = ackreftable(:,cidx);
    % Có truyền SR
    if dc_numOfsr
        resSr = (ridx == 2);
    else
        resSr = false(1,0);
    end
```

Không giá trị nào vượt ngưỡng

```
else
    resAck = false(0,1);
    resSr = false (srOnly,1);
end
ucival = {resAck resSr}
```

ucival = 1×2 cell

	1	2
1	[0;0]	1

```
uci
```

uci = 1×2 cell

	1	2
1	[0;0]	1

```
% thêm để so sánh với hàm của MATLAB
fun = decodePUCCH0dx(carrier,pucch,[lenACK lenSR],seq)
```

fun = 1×2 cell

	1	2
1	[0;0]	1