**Nhân viên: Nguyễn Đức Ngọc Toàn**

**MSNV: 277858**

**BỘ TẠO KÊNH PUSCH**

**I. Giải thuật chi tiết**

**1. Determine Transport Block Size:**

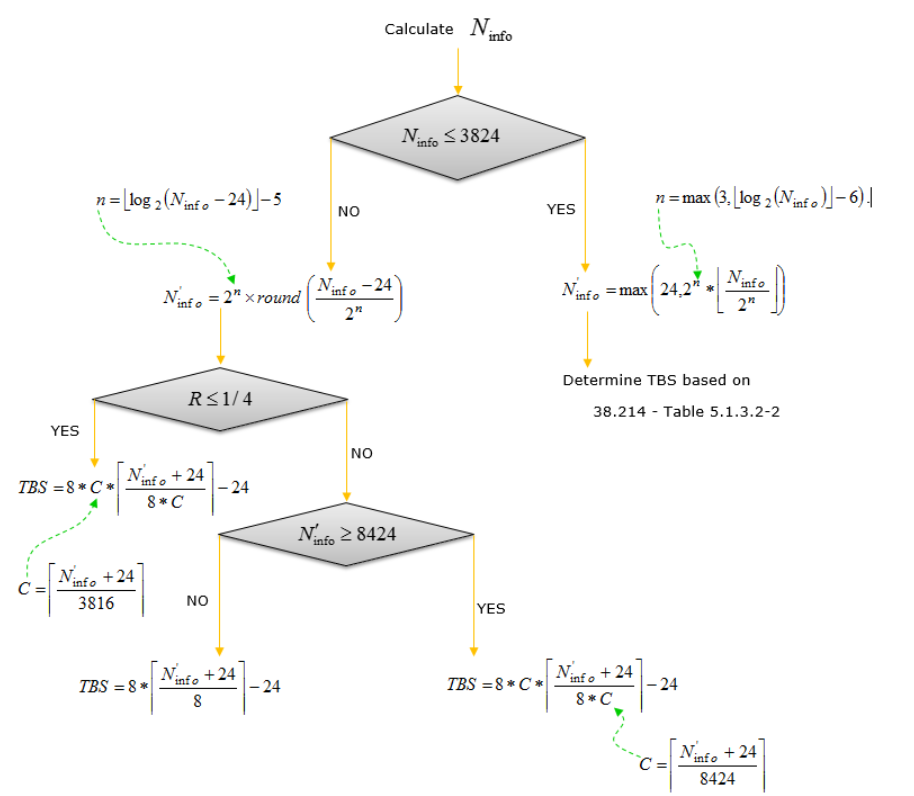
* *Input:*
* *Các tham số: NPrb, NPuschSymbAll, Nlayers, NPrbOh, Qm, R, DmrsConfigurationType, PuschMappingType, DmrsDuration, DmrsAdditionalPosition, NumDmnrCdmGroupWithoutdata*
* *Output:*
* *Kích thước của khối truyền tải TBSsize.*
* *Số RE dành cho PUSCH trong 1RB.*
* *Số RE dành cho DMRS trong 1RB.*
* *Giải thuật:* (Ts 28.214-6.1.4.2)

+ N’RE = NscRB \* Nsymbsh - NDMRSPRB - NohPRB

+ Tổng số RE dành cho PUSCH: NRE = min(156,N’RE)\*nPRB

+ Tổng số bit thông tin (information bits): Ninfo = NRB\*R\*Qm\*v

+ Xác định TBS:



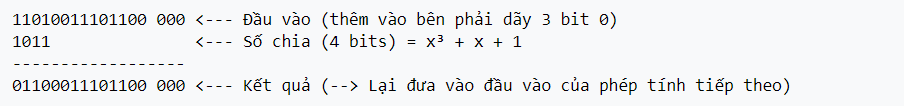
**2. Transport Block CRC Attachment:**

* *Input:*
* *Chuỗi bit đầu vào.*
* *Dạng đa thức sinh: 24A, 24B, 24C, 16.*
* *Output:*
* *Chuỗi bit đã được thêm các bit kiểm tra CRC.*
* *Giải thuật*:

+ Mở rộng chuỗi bit đầu vào với n bit 0 (n: số bit CRC).

+ Thực hiện chia chuỗi bit được mở rộng với đa thức sinh.

*Ví dụ*



Kiểm tra nếu bit ngoài cũng bên trái là bit 0 thì không làm gì cả, dịch số chia sang phải 1 bit. Ngược lại nếu là bit 1 thì lấy dãy đầu vào trừ đi đa thức sinh và dịch số chia sang phải 1 đơn vị. Cứ tiếp tục như vậy cho đến khi không còn chia được nữa.

+ Kết quả phần dư cuối cùng chính là chuỗi các bit CRC được thêm vào.

**3. LDPC base graph selection:**

* *Input:*
* *Kich thước khối truyền tải TBSsize.*
* *Code rate R.*
* *Output:*
* *LDPC graph matrix type 1 hoặc type*
* *Giải thuật*: (Ts 38.212-6.2.2)

If (A <= 292) or (A <= 3824 and R <= 0.67) or (R <= 0.25)

LDPCbasegraph = ‘type2’

else LDPCbasegraph = ‘type1’

**4. Code Block Segmentation and Code Block CRC Attachment**

* *Input:*
* *Chuỗi bit ngõ vào*
* *LDPC graph matrix type 1 hoặc type 2*
* *Output:*
* *Các khỗi mã đã được phân đoạn*
* *setIdx, Zc*
* *Giải thuât*: (Ts 38.212-6.2.3)

+ Xác định kích thước lớn nhất của code block (Kcb)

if LDPC base graph = ‘type 1’

Kcb = 8448

else Kcb = 3840

+ Xác định số lượng code block

if B(Transport block size) < Kcb(Max Codeblock size)

     L = 0; C = 1; B' = B   // Không thực hiện phân đoạn

Else L = 24

     C = Ceiling(B/(Kcb - L))

     B' = B + C \* L

+ Xác định số lượng bits của mỗi code block: K’ = B’/C

+ Xác định Kb

If LDPC base graph = ‘type 1’

Kb = 22

If LDPC base graph = ‘type 2’

if B (Transport blocksize) > 640 Kb = 10

else if B (Transport blocksize) > 560 Kb = 9

else if B (Transport blocksize) > 192 Kb = 8

else Kb = 6

+ Xác định giá trị nhỏ nhất trong bảng LDPC lifting size Zc sao cho Kb\*Zc >= K’

+ Xác định kích thước của chính xác của code block:

If (LDPC base graph == ‘type 1’) K = 22Zc

Else K = 10Zc

+ Thực hiện phân đoạn và chèn bits CRC vào mỗi code block

s = 0  // s = bit position in B (transport block)

for r = 0 to C-1

for k = 0 to K'-L-1

crk = bs; s = s + 1

end for

if C > 1

for k = K'-L to K'-1  // Chèn các bits CRC

crk = pr(k+L-K')

end for

end if

for k = K' to K-1  // Chèn thêm vào các bit NULL (-1) để đảm bảo chiều dài mỗi code block trước khi vào khối LDPC là hợp lệ

crk = < NULL >

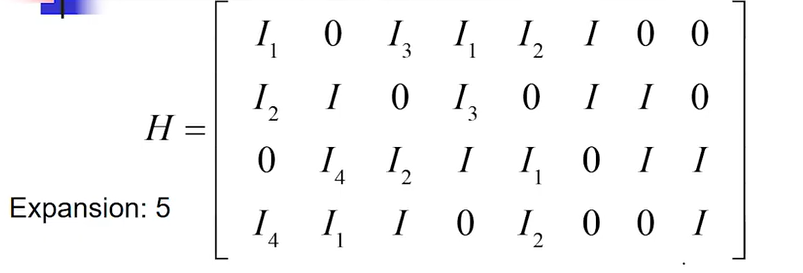
end for

end for

**5. LDPC Channel Coding:**

* *Input:*
* *Chuỗi bit ngõ vào.*
* *LDPC graph matrix type 1 hoặc type 2.*
* *setIdx và Zc.*
* *Output:*
* *Chuỗi bit được mã hóa LDPC*
* *Chuẩn*: Thực hiện tìm các bit parity w sao cho Hx]. Trong đó c là input và ma trận H có được bằng cách thay thế mỗi phần tử của Base Graph Matrix HBG bằng một ma trận ZcxZc
* *Giải thuật*: Sử dụng giải thuật double diagonal

*+ Ví dụ:* Ta có ma trận Base Graph Matrix như sau. Phần tử I là ma trận đơn vị ZcxZc, phân tử 0 là ma trận 0 có kích thước ZcxZc



+ Theo chuẩn LDPC ta tìm các bit parity sao cho H x [m1 m2 m3 m4 p1 p2 p3 p4]T  = 0. Trong đó m1, m2, m3, m4 là các mảng thông tin đầu vào có kích thước là Zc và p1, p2, p3, p4 là các mảng bit parity cũng có kích thước Zc. Thực hiện phép tính:

* Hàng 1: I1m1 + I3m3 + I1m4 + I2p1 = 0
* Hàng 2: I2m1 + Im2 + I3m4 + Ip2 + Ip3 = 0
* Hàng 3: I4m2 + I2m3 + Im4 + I1p1 + Ip3 + Ip4 = 0
* Hàng 4: I4m1 + I1m2 + Im3 + I2p1 + Ip4 = 0

Lấy hàng 1 + hàng 2 + hàng 3 + hàng 4 ta được: I1p1 = I1m1 +I2m3 + I1m4 + I2m1 + Im2 + I3m3 + I4m2 + I2m3 + Im4 + I4m1 + I1m2 + Im3 = 0. Từ đó tìm được p1

+ Sử dụng hàng 1, hàng 2, hàng 3 để tìm lần lượt p1, p2, p3, p4

+ Với Base Graph Matrix của LDPC channel coding ta cũng thực hiện tương tự để tìm các bit parity.

**6. Rate Matching**

* *Input:*
* *Chuỗi bit ngõ vào*
* *Các tham số: Nlayers,ILbrm,rv,Qm,A,LDPCgraph,C,Zc*
* *Output:*
* *Chuỗi bit ngõ ra được phối hợp tốc độ*
* *Giải thuật: (Ts 38.212-6.2.5)*

+ Bit selection:

Xác định chiều dài chuỗi bits đầu ra Er cho mỗi khối.

set j = 0;

for r = 1 to C

if j <= (C-mod(A/(Nl\*Qm),C)-1);

E(i) = Nl\*Qm\*floor(A/(Nl\*Qm\*C));

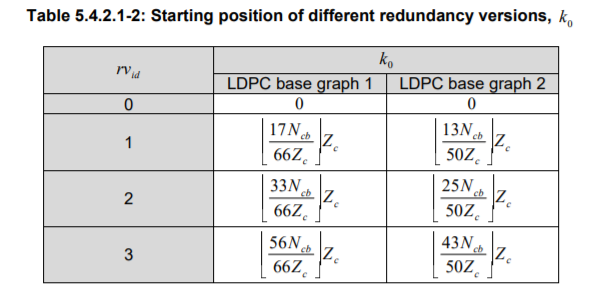
else E(i) = Nl\*Qm\*ceil(A/(Nl\*Qm\*C));

end if

j = j + 1

end for

Xác định vị trí bắt đầu k0 của mỗi rv theo bảng



Lựa chọn chuỗi bit để truyền

k = 1; j = 1;

while k < E

If d­ko+j mod Ncb ≠ NULL

ek = d­ko+j mod Ncb ;

k = k + 1;

end if

j = j + 1;

end while

+ Bit interleaving:

for j = 0 to E/Qm – 1

for i = 0 to Qm – 1

fi + jQm + 1 = ei\*E/Qm + 1 + j

end for

end for

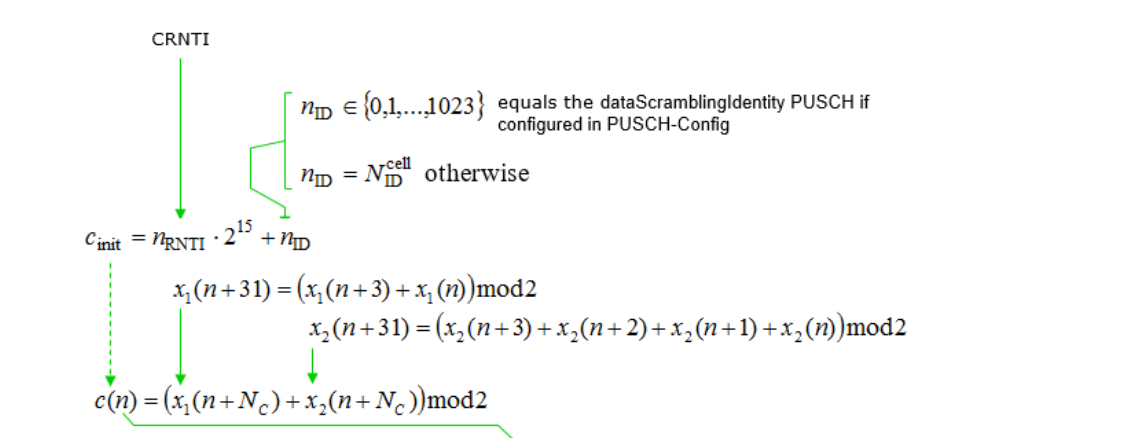
**7. Code Block Concatenation:**

* *Input:*
* *Các khối mã ngõ vào*
* *Output:*
* *Chuỗi bit ngõ ra duy nhất được móc nối*
* *Giải thuật: (Ts 38.212 5.5)*

**8. Scrambing**

* *Input:*
* *Chuỗi bit ngõ vào.*
* *Các tham số nRNTI, nID.*
* *Output:*
* *Chuỗi bit đã được ngẫu nhiên hóa*
* *Giải thuật: (Ts 38.211 6.3.1.1)*

+ Tạo chuỗi giả scrambling c(n) với cinit = nRNTI.215+ nID

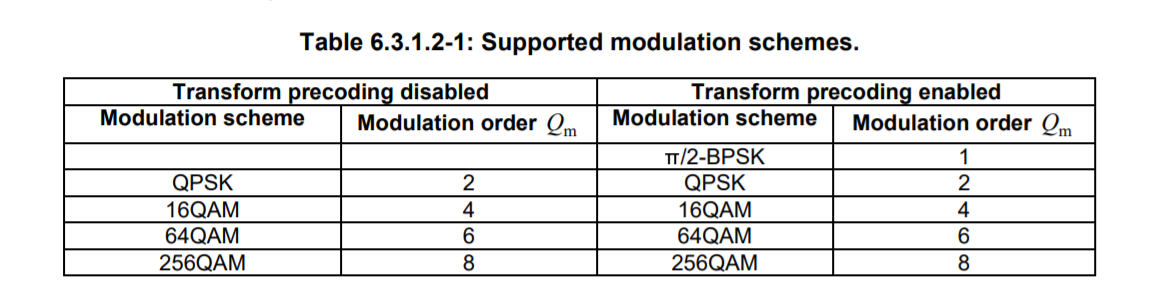


+ Chuỗi ngõ ra: bq = mod(in + c, 2)

9. **Modulation:**

* *Input:*
* *Chuỗi bit ngõ vào*
* *Các tham số Qm, ue.transformprecoding*
* *Output:*
* *Chuỗi kí hiệu đã được điều chế*
* *Giải thuật: (Ts 38.211 6.3.1.2)*

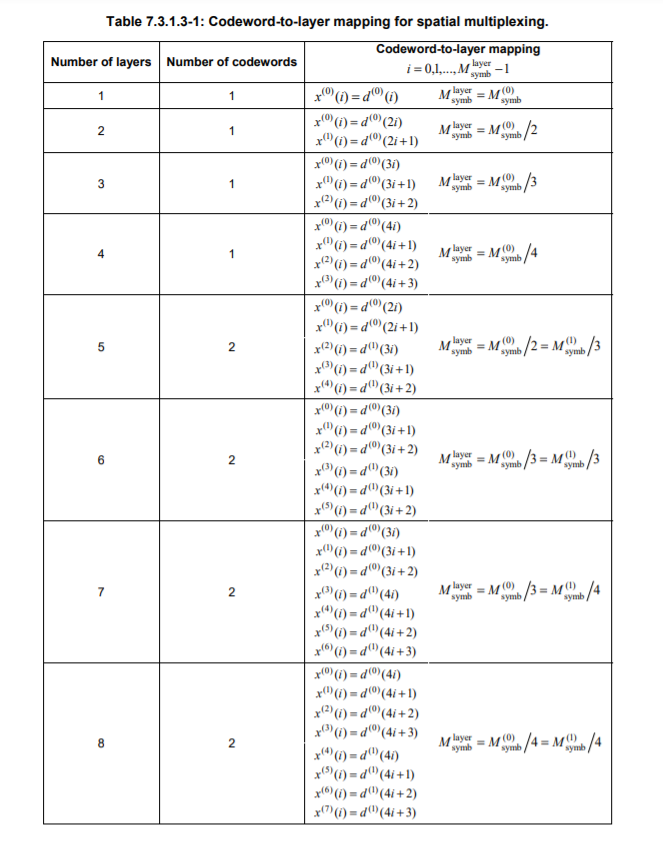
+ Kiểu điều chế ngõ ra dựa vào bảng sau



**10. Layer Mapping**

* *Input:*
* *Chuỗi kí hiệu ngõ vào.*
* *Các tham số: Ncw, Nlayers*
* *Output:*
* *Chuỗi kí hiệu trên các layers*
* *Giải thuật: (Ts 38.211 6.3.1.2)*

+ Chuỗi bit ngõ vào được ánh xạ lớp phụ thuộc vào tham số Ncw và Nlayers theo bảng



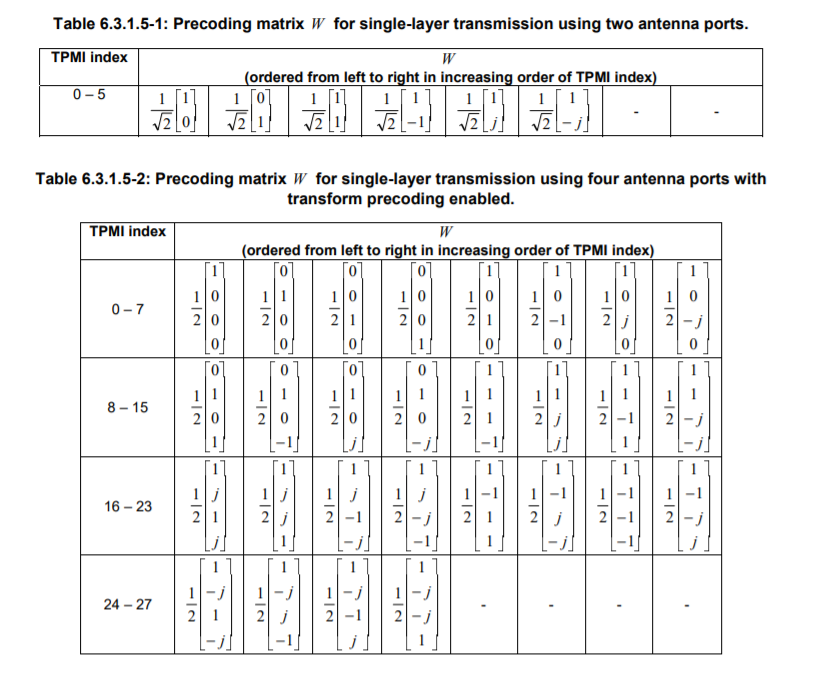
**11. Precoding:**

* *Input:*
* *Chuỗi kí hiệu ngõ vào trên các layers*
* *Các tham số: Nlayers, Nant, Tpmi, ue.transform precoding*
* *Output:*

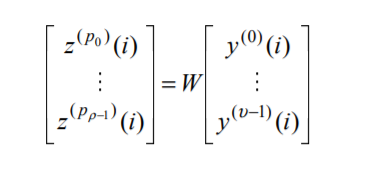
*- Chuỗi kí hiệu trên các antenna sau khi tiền mã hóa*

* *Giải thuật: (Ts 38.211 6.3.1.5)*

+ Từ Nlayers, Nant, Tpmi, transform precoding xác đinh được ma trận precoding theo bảng 6.3.1.5-1, 2, 3, 4, 5, 6, 7



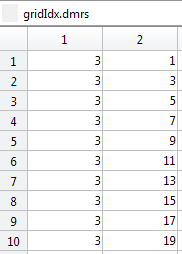
+ Các symbol được ánh xạ lên các antenna port theo công thức: Trong đó p là số port antenna và v là số layers



**12. Tạo lưới tài nguyên**

* *Input:*
* *Vị trí của ký hiệu DMRS*
* *Vị trí của ký hiệu PUSCH*
* *Ký hiệu PUSCH sau khi precoding*
* *Các tham số: NDmrsperPrb, ue.NPrb, sys.Numerology, sys.CpType, ue.NnScIdId, ue.nScId, ue.NumDmnrCdmGroupWithoutdata*
* *Output:*
* *Lưới tài nguyên chứa các ký hiệu PUSCH và ký hiệu tham chuẩn DMRS*
* *Giải thuật:*

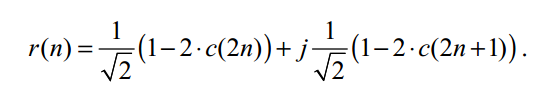
+ Tạo 2 ma trận chứa vị trí của ký hiệu DMRS và PUSCH có form như sau:



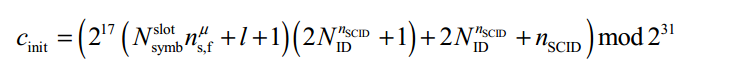
Trong đó các giá trị cột 1 và cột 2 lần lượt là giá trị cột và hàng của ma trận lưới tài nguyên. Việc tạo ma trận chứa các giá trị này khác nhau đối với mỗi cấu hình DMRS, nếu xây hàm tổng quát sẽ khó bao quát hết tất cả các trường hợp cấu hình DMRS khác nhau.

+ Ánh xạ ký hiệu DMRS:

Tạo ký hiệu DMRS r(n)



Trong đó c(n) là chuỗi giả được xây dựng giống trong khối scrambling với cinit được tính như sau:



Sau khi tạo được chuỗi ký hiệu DMRS, lần lượt mapping các ký hiệu lên lưới tài nguyên.

+ Ánh ký hiệu PUSCH: mapping lần lượt các ký hiệu PUSCH theo ma trận vị trí đã được tạo ra ở trên

**13. Điều chế OFDM**

* *Input:*
* *Lưới tài nguyên*
* *Các tham số: sys.Numerology, sys.BwpNrb, sys.CPType.*
* *Output:*
* *Tín hiệu trong miền thời gian.*
* *Giải thuật:*
* Tính các tham số cần thiết:

+ Số điểm fft: Nfft = 2ceil(log2(Nrb\*12/0.85)) => Nfft = max(128,Nfft)

+ Tần số lấy mẫu: Sampling Rate = Nfft\*scs

+ Windowing = max(0, 8-2\*(11-log2(Nfft)))

+ Sỗ mẫu CP cho mỗi symbol trên 1 slot

* Mapping lưới tài nguyên lên lưới fft có kích thước 4096\*14 (đối với 1 slot).
* Tính fft đối với mỗi symbol trong slot vừa tạo ra => Chèn CP => Windowning (Lọc windowing bằng 1 hàm raise\_cosine) để tạo tín hiệu ngõ ra trong miền thời gian.

**II. So sánh tính chính xác của bộ tạo tín hiệu PUSCH với Matlab Toolbox**

**1. So sánh từng khối**

* *Khối CRC:*

+ Code thực hiện :



+ Matlab Toolbox:



+ Thực hiện lệnh isequal đề kiểm tra độ chính xác của ngõ ra



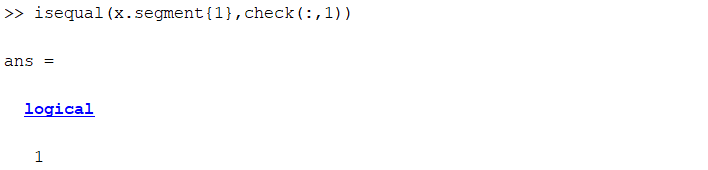
* Kết quả bằng 1 tức là chuỗi ngõ ra khối crc giống với Matlab Toolbox
* *Khối segmentation*

+ Code thực hiện: 

+ Matlab Toolbox



+ Kết quả trường hợp này chia thành 4 khối nhỏ nên lần lượt kiểm tra độ chính xác của từng khối



* Kết quả chính xác với Matlab Toolbox
* *Thực hiện kiểm tra tương tự đối các khối LDPC, Rate Matching (khối rate matching kiểm tra độ chính xác các redundancy version 0 1 2 3), Scrambling, layermapping, precoding (khối precoding chỉ kiểm tra ma trận precoding).*

**2. So sánh từ khối CRC đến khối Concatenation:**

* Toolbox Matlab:

+ Step1: Tạo object nrULSCH và set cấu hình: [encUL = nrULSCH(Name,Value)](https://www.mathworks.com/help/5g/ref/nrulsch-system-object.html#d122e55056)

Với case QPSK R = 193/1024



+ Step2: Load khối TBS vào object: [setTransportBlock(enc,trblk)](https://www.mathworks.com/help/5g/ref/nrulsch.settransportblock.html#d122e48587)

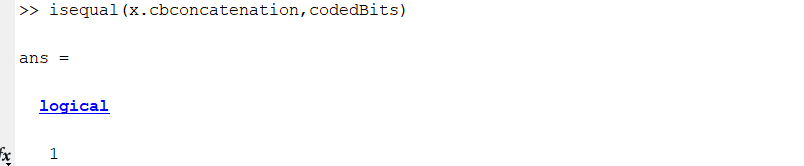


+ Step3: Thực hiện lệnh: [codedBits = encUL(mod,nLayers,outlen,rv)](https://www.mathworks.com/help/5g/ref/nrulsch-system-object.html#d122e55212)

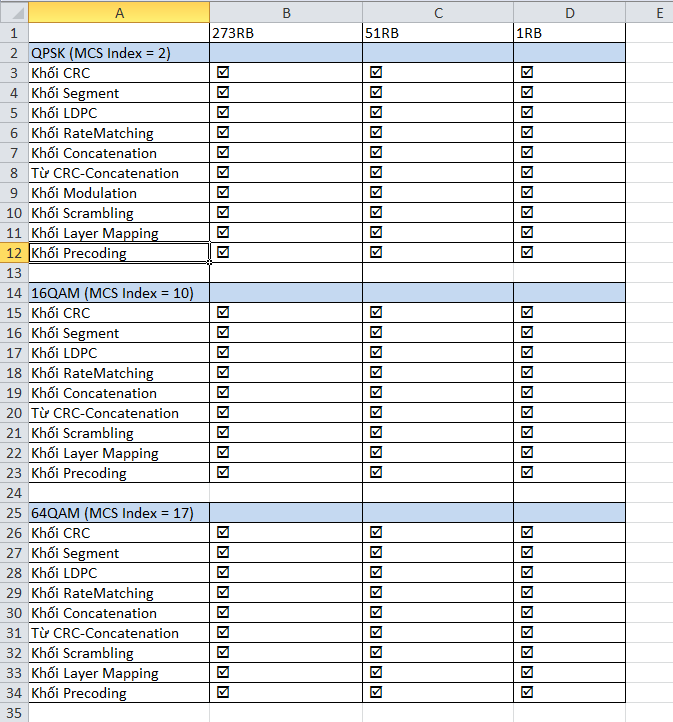
Vd case QPSK, 1 layer, số bit ngõ ra sau khối concatenation là 78624 và rv = 3



+ Kết quả:



* Kết quả giữa bộ tạo kênh từ khối CRC đến concatenation với Matlab Toolbox giống nhau
* Thực hiện kiểm tra tương tự trên toàn bộ các case: 1RB, 51RB, 273RB, QPSK, 16QAM, 64QAM



**3. So sánh toàn bộ kênh PUSCH:**

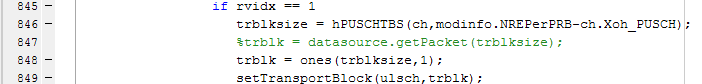
* Step 1:

+ Mở ví dụ 5G NR Uplink Carrier Waveform Generation có trong Toolbox Matlab.

+ Thực hiện động bộ tham số với tham số đầu vào được cho. Ở đây chỉ sử dụng 1 BWP và tín hiệu ULSCH nên các tín hiệu còn lại set disable.

* Step 2: Tạo dữ liệu đầu vào giống nhau.

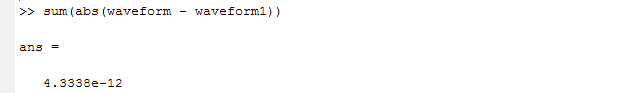
+ Với Toolbox Matlab, điều chỉnh input data ở file hNRUplinkWaveformGenerator.m



* Step 3:

+ Tạo tín hiệu (case: 20 slots, 273RB, 64QAM) ở Toolbox Matlab và code thực hiện.

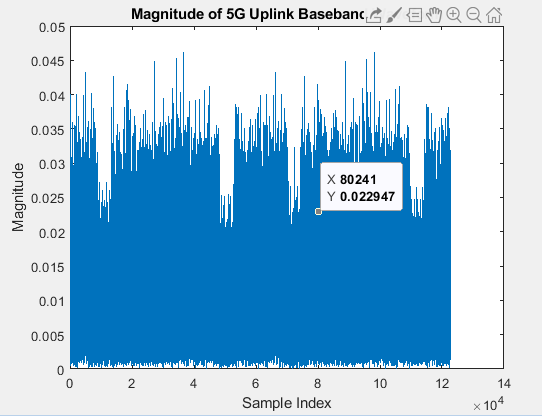
+ Kiểm tra sai số:



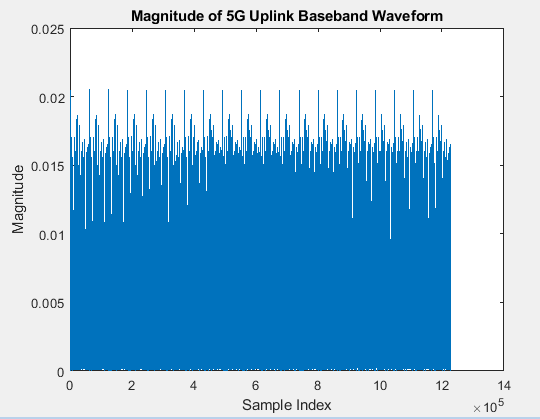
* Sai số gần như bằng không
* Step 4: Thực hiện kiểm tra ở các case 51 RB, 1RB, QPSK, 16QAM, 64 QAM, 1 AntTx, 2 AntTx, 4 AntTx đều cho kết quả đúng.

**III. Một vài kết quả ngõ ra của bộ tạo kênh PUSCH:**

* Case: 273RB, QPSK (MCS = 2)



* Case: 51RB, 16QAM (MCS = 10)



* Case: 1RB, 64QAM (MCS = 17)

