```
classdef NRgNBTxFD < matlab.System
    % 5G NR gNB transmitter class implemented in frequency domain
    properties
        % Configuration
        carrierConfig;   % Carrier configuration
        pdschConfig;     % PDSCH configuration

        % Transport block data for last transmission
        targetCodeRate = 490/1024;  % Target code rate
        trBlkSizes;                 % Transport block size

        % DLSCH encoder
        encDLSCH;

        % HARQ Process
        nharq = 8;      % number of HARQ processes

        % RV sequence.  This is the sequence that the TX will cycle
        % through in the RVs
        rvSeq = [0,3,2,1]';

        % TX parameters per HARQ process
        rvInd;  % Index of the RV for the current transmission
        newDataAvail;   % If HARQ process can take new data
        txBits;         % Cell array of TX bits

    end
    methods
        function obj = NRgNBTxFD(carrierConfig, pdschConfig, ...
                varargin)
            % Constructor

            % Save the carrier and PDSCH configuration
            obj.carrierConfig = carrierConfig;
            obj.pdschConfig = pdschConfig;

            % Set parameters from constructor arguments
            if nargin >= 1
                obj.set(varargin{:});
            end

            % Create DLSCH encoder system object
            obj.encDLSCH = nrDLSCH('MultipleHARQProcesses', true, ...
                'TargetCodeRate', obj.targetCodeRate);

            % Initialize the HARQ process parameters
            obj.rvInd = zeros(obj.nharq, 1);
            obj.newDataAvail = ones(obj.nharq,1);

            % TX bits for each HARQ process
            obj.txBits = cell(obj.nharq,1);


        end

         function setAck(obj, iharq)
            % Set that the HARQ transmission was received correctly
            obj.newDataAvail(iharq) = 1;
```

```
    end
end
methods (Access = protected)

    function [txGrid, rv, newDat] = stepImpl(obj, iharq)
        % step implementation. Creates one slot of samples for each
        % component carrier
        %
        % Parameters
        % ----------
        % iharq:  HARQ process index to use
        %
        % Returns:
        % --------
        % txGrid:  OFDM grid of transmitted symbols
        % rv:  Redundancy version for the data
        % newDat:  If new data was transmitted in this slot


        % Create the OFDM grid representing the array of modulation
        % symbols to be transmitted
        txGrid = nrResourceGrid(obj.carrierConfig, ...
            obj.pdschConfig.NumLayers);


        % Get indices on where the PDSCH is allocated
        [pdschInd,pdschInfo] = nrPDSCHIndices(obj.carrierConfig, obj.pdschConfig);


        if obj.newDataAvail(iharq)
            % If new data can be transmitted in the HARQ process

            % Compute the extra overhead from the PT-RS
            Xoh_PDSCH = 6*obj.pdschConfig.EnablePTRS;

            % Calculate the transport block size based on the PDSCH
            % allocation and target code rate
            obj.trBlkSizes = nrTBS(obj.pdschConfig.Modulation,obj.pdschConfig.NumLayers,...
                numel(obj.pdschConfig.PRBSet),pdschInfo.NREPerPRB,...
                obj.targetCodeRate,Xoh_PDSCH);

            % Generate random bits for each codeword and set the transport
            % block
            obj.txBits{iharq} = cell(obj.pdschConfig.NumCodewords, 1);
            for icw = 1:obj.pdschConfig.NumCodewords

                % TODO:  Create random bits for each codeword
                obj.txBits{iharq}{icw} = randi([0 1], obj.trBlkSizes(icw), 1);

                % TODO:  Set the transport block to be encoded
                obj.encDLSCH.setTransportBlock(obj.txBits{iharq}{icw},icw-1,iharq-1)
                % You will need to pass the codeword index, icw-1,
                % and HARQ process ID, iharq-1
            end

            % Set the RV index to zero on the first transmission
            obj.rvInd(iharq) = 0;

            % Clear new data flag
            obj.newDataAvail(iharq) = 0;
```

```
                % Mark that the data
                newDat = true;
            else
                % Mark that this data is a re-transmission
                newDat = false;
            end

            % TODO: Get the redundancy version from the current redundancy
            % version index, obj.rvInd(iharq).  The rv should be
            irv = mod(obj.rvInd(iharq),4);
            rv = obj.rvSeq(irv+1);
            % where irv cycles, 0,1,2,3,0,1,2,3,...

            % Encode the DL-SCH transport block
            codedTrBlock = obj.encDLSCH(obj.pdschConfig.Modulation, ...
                obj.pdschConfig.NumLayers, pdschInfo.G, rv, iharq-1);

            % Increment the RV sequence
            obj.rvInd(iharq) = obj.rvInd(iharq) + 1;

            % Modulate the PDSCH modulation
            pdschSymbols = nrPDSCH(obj.carrierConfig, obj.pdschConfig, ...
                codedTrBlock);

            % Map the modulated symbols to the OFDM grid
            txGrid(pdschInd) = pdschSymbols;

        end



    end
end
```

```
Not enough input arguments.

Error in NRgNBTxFD (line 34)
            obj.carrierConfig = carrierConfig;
```

---

*Published with MATLAB® R2020b*