```
classdef NRUERxFD < matlab.System
    % 5G NR UR receiver class implemented in frequency domain
    properties
        % Configuration
        carrierConfig;   % Carrier configuration
        pdschConfig;     % Default PDSCH config
        waveformConfig;  % Waveform config

        % OFDM grid
        rxGrid;

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

        % Received data in last slots
        pdschEq;       % Equalized PDSCH symbols
        rxBits;        % RX bits

        % DLSCH decoder
        decDLSCH;


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

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

            % Create the waveform configuration from the carrier
            % configuration
            obj.waveformConfig = nrOFDMInfo(obj.carrierConfig);

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

            % Create DLSCH decoder
            obj.decDLSCH = nrDLSCHDecoder('MultipleHARQProcesses', false, ...
                'TargetCodeRate', obj.targetCodeRate, ...
                'LDPCDecodingAlgorithm', 'Layered belief propagation');

        end
    end
    methods (Access = protected)


        function stepImpl(obj, rxGrid, chanGrid, noiseVar)
            % Demodulates and decodes one slot of data

            % Get PDSCH received symbols and channel estimates
            % from received grid
            [pdschInd,pdschInfo] = nrPDSCHIndices(obj.carrierConfig, obj.pdschConfig);
            [pdschRx, pdschHest] = nrExtractResources(pdschInd, rxGrid, chanGrid);
            % TODO:  Perform the MMSE equalization using the
            % nrEqualizeMMSE() function.
            % Use the PDSCH Rx symbols, PDSCH channel estimate and noise
            % variance as the input.  Store the equalized symbols in
            % obj.pdschEq and  channel state information in a structure,
            % csi.
            [obj.pdschEq,csi] = nrEqualizeMMSE(pdschRx,pdschHest,noiseVar);

            % TODO:  Get the LLRs with the nrPDSCHDecode() function.
            % Use carrier and PDSCH configuration, the equalized symbols,
            % and the noise variance, noiseVar.
            [dlschLLRs,rxSym] = nrPDSCHDecode(obj.carrierConfig,obj.pdschConfig,obj.pdschEq, noiseVar);

            % Scale LLRs by EbN0.
            % The csi value computed in the nrEqualizeMMSE()
            % function is csi = |pdschHest|^2 + noiseVar.
            % Also, the Eb/N0 = snrEq/Qm where Qm is the number of bits
            % per symbol and snrEq is the SNR after equalization,
            %
            %   snrEq = (|pdschHest|^2 + noiseVar)/noiseVar = csi/noiseVar
            %
            % Hence, Eb/N0 = csi/(noiseVar*Qm).
            % Since the LLRs from the nrPDSCHDecode function are
            % already scaled by 1/noiseVar, we multiply them by  csi/Qm.

            csi = nrLayerDemap(csi); % CSI layer demapping
            numCW = length(csi);
            for cwIdx = 1:numCW
                Qm = length(dlschLLRs{cwIdx})/length(rxSym{cwIdx}); % bits per symbol
                csi{cwIdx} = repmat(csi{cwIdx}.',Qm,1);   % expand by each bit per symbol
                dlschLLRs{cwIdx} = dlschLLRs{cwIdx} .* csi{cwIdx}(:);   % scale
            end

            % 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);
            obj.decDLSCH.TransportBlockLength = obj.trBlkSizes;

            % Reset the soft buffer
            harqId = 0;
            obj.decDLSCH.resetSoftBuffer(harqId);

            % TODO:  Decode the bits with the obj.decDLSCH() method.
            % Use the scaled LLRs from above. Use a redundancy version,
            % rv = 0,  since we are not using HARQ in this lab.
            rv = 0;
            obj.rxBits = obj.decDLSCH(dlschLLRs,obj.pdschConfig.Modulation,obj.pdschConfig.NumLayers,rv);

        end

    end
end
```

```
Not enough input arguments.

Error in NRUERxFD (line 31)
            obj.carrierConfig = carrierConfig;
```

Published with MATLAB® R2020b