```
classdef FDMIMOChan < matlab.System
    % Frequency-domain MIMO multipath channel
    properties
        % Configuration
        carrierConfig;   % Carrier configuration
        waveformConfig;  % Waveform parameters

        % Path parameters
        gain;  % Relative path gain in dB
        dly;   % Delay of each path in seconds
        aodAz, aodEl; % Angle of departure of each path in degrees
        aoaAz, aoaEl; % Angle of arrival of each path in degrees

        % Derived path parameters
        fd;    % Doppler shift for each path
        gainComplex;  % Complex gain of each path
        svTx, svRx;   % Steering vectors for each path
        elemGainTx, elemGainRx;  % Element gains

        % Other parmaters
        fc = 28e9;    % Carrier freq in Hz
        rxVel = [30,0,0]';  % RX velocity vector in m/s
        txVel = [0,0,0]';   % TX velocity vector in m/s
        Enoise = 0;         % Noise energy per sample in dBmJ

        % Symbol times
        symStart;  % symStart(i) = start of symbol i relative to subframe

        % TX and RX array platforms
        txArrPlatform = [];
        rxArrPlatform = [];

    end
    methods
        function obj = FDMIMOChan(carrierConfig, varargin)
            % Constructor

            % Save the carrier configuration
            obj.carrierConfig = carrierConfig;

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

            % Check all the required fields are specified
            fields = {'txArrPlatform', 'rxArrPlatform', 'gain', 'dly', ...
                'aoaAz', 'aodAz', 'aoaEl', 'aoaAz' };
            nfields = length(fields);
            for i = 1:nfields
                fstr = fields{i};
                if isempty(obj.(fstr))
                    e =  MException('FDMIMOChan:missingParam', ...
                        'Parameter %s not speficied', fstr);
                    throw(e);
                end
            end

            % Complex gain for each path using a random initial phase
            % The gains are normalized to an average of one
            npath = length(obj.gain);
            phase = 2*pi*rand(npath, 1);
            obj.gainComplex = db2mag(obj.gain).*exp(1i*phase);

            % Symbol times relative to the start of the subframe
            obj.waveformConfig = nrOFDMInfo(obj.carrierConfig);
            nsym = obj.waveformConfig.SymbolLengths;
            obj.symStart = nsym/obj.waveformConfig.SampleRate;
            obj.symStart = cumsum([0 obj.symStart]');

            % Get Doppler shift for RX
            vc = physconst('Lightspeed');
            [ux, uy, uz] = sph2cart(deg2rad(obj.aoaAz), deg2rad(obj.aoaEl), 1);
            obj.fd = [ux uy uz]*obj.rxVel*obj.fc/vc;

            % Get Doppler shift for TX
            [ux, uy, uz] = sph2cart(deg2rad(obj.aodAz), deg2rad(obj.aodEl), 1);
            obj.fd = obj.fd + [ux uy uz]*obj.txVel*obj.fc/vc;
        end

        function computePathSV(obj)
            % Computes the element gains and steering vectors of each path

            % Call the array platform objects to get the steering vectors
            % and element gains
            [obj.svTx, obj.elemGainTx] = ...
                obj.txArrPlatform.step(obj.aodAz', obj.aodEl',true);
            [obj.svRx, obj.elemGainRx] = ...
                obj.rxArrPlatform.step(obj.aoaAz', obj.aoaEl',true);

        end


    end
    methods (Access = protected)


        function [chanGrid, noiseVar] = stepImpl(obj, frameNum, slotNum)
            % Applies a frequency domain channel and noise
            %
            % Parameters
            % ----------
            % frameNum:  The index of the frame  (1 frame = 10ms)
            % slotNum:  The index of the slot in the frame
            %    This should be 0,...,waveformConfig.SlotsPerFrame
            %
            % Outputs
            % -------
            % chanGrid:  Grid of the channel values
            % noiseVar:  Noise variance

            % Compute the steering vectors and element gains
            obj.computePathSV();

            % Get the number of TX and RX elements
            ntx = obj.txArrPlatform.getNumElements();
            nrx = obj.rxArrPlatform.getNumElements();

            % Get the number of sub-carriers
            nscPerRB = 12;
            nsc = obj.carrierConfig.NSizeGrid * nscPerRB;
            nsym = obj.carrierConfig.SymbolsPerSlot;

            % Compute the frequency of each carrier
            f = (0:nsc-1)'*obj.carrierConfig.SubcarrierSpacing*1e3;

            % Compute slot in sub-frame and sub-frame index
            sfNum = floor(slotNum / obj.waveformConfig.SlotsPerSubframe);
            slotNum1 = mod(slotNum, obj.waveformConfig.SlotsPerSubframe);

            % Compute the time for each symbol
            framePeriod = 0.01;
            sfPeriod = 1e-3;
            t = frameNum*framePeriod + sfPeriod*sfNum + ...
                obj.symStart(slotNum1+1:slotNum1+nsym);

            % Initialize the channel grid to zero
            chanGrid = zeros(nrx, ntx, nsc, nsym);
            npath = length(obj.gain);

            % TODO: Set the channel:
            %
            % chanGrid(j,k,n,t) = MIMO channel matrix from
            %    RX antenna j, TX antenna k, sub-carrier n,
            %    symbol t.
            %
            % This should be a sum of the paths
            %
            % chanGrid(j,k,:,:)
            %   = \sum_i exp(1i*phase)*svRx(j,i)*svTx(k,i)
            %
            % where
            %
            % phase = 2*pi*(f*obj.dly(i) + t'*obj.fd(i));

            for j=1:nrx
                for k=1:ntx
                    for p=1:npath
                        phase = 2*pi*(f*obj.dly(p) + t'*obj.fd(p));
                        chan_path = exp(1i*phase)*obj.svRx(j,p)*obj.svTx(k,p);
                        curr_chan = reshape(chanGrid(j,k,:,:),nsc,nsym);
                        chanGrid(j,k,:,:) = curr_chan + chan_path;
                    end
                end
            end

            % Compute noise variance
            noiseVar = db2pow(obj.Enoise);

        end

    end
end
```

```
Not enough input arguments.

Error in FDMIMOChan (line 39)
            obj.carrierConfig = carrierConfig;
```

*Published with MATLAB® R2020b*