```
classdef SISOMPChan < matlab.System
    % SISOMPChan:  SISO multi-path fading channel
    properties
        fsamp;   % Sample rate in Hz

        % Path properties
        gain;  % path gains in dB
        dly;   % delays in seconds
        dop;   % doppler shift of each path in Hz
        fc;    % the frequency

        % Fractional delay object
        fracDly;

        % Initial set of phases for the next step call
        phaseInit;

    end

    methods
        function obj = SISOMPChan(varargin)
            % Constructor:
            % The syntax allows you to call the constructor with syntax of
            % the form:
            %
            %     chan = SISOMPChan('Prop1', Val1, 'Prop2', val2, ...);
            if nargin >= 1
                obj.set(varargin{:});
            end

        end

    end
    methods (Access = protected)
        function setupImpl(obj)
              % setup:  This is called before the first step.
              % For the SISO MP channel, we will use this point to
              % construct the fractional delay object.

              % TODO:  Create a dsp.VariableFractionalDelay object
              % and store it is fracDly.  Use the parameters
              %  'InterpolationMethod', 'Farrow',
              %  'FilterLength',8
              %  'FarrowSmallDelayAction','Use off-centered kernel',...
              %  'MaximumDelay', 1024
              obj.fracDly = dsp.VariableFractionalDelay(...
                            'InterpolationMethod', 'Farrow','FilterLength',8,...
                            'FarrowSmallDelayAction','Use off-centered kernel',...
                            'MaximumDelay', 1024);
        end

        function resetImpl(obj)
            % reset:  Called on the first step after reset or release.

            % TODO:  Reset the fracDly object
            reset(obj.fracDly);

            % TODO:  Initialize phases, phaseInit, to a row vector of
            % dimension equal to the number of paths with uniform values
```

```
            % from 0 to 2pi
            obj.phaseInit = rand(size(obj.gain))*2*pi;
        end

        function releaseImpl(obj)
            % release:  Called after the release method

            % TODO:  Release the fracDly object
            release(obj.fracDly);
        end

        function y = stepImpl(obj, x)
            % step:  Run a vector of samples through the channel

            % TODO:  Compute the delay in samples
            dlySamp = obj.dly.*obj.fsamp;

            % TODO:  Compute gain of each path in linear scale
            gainLin = db2pow(obj.gain);

            % TODO:  Use the fracDly object to compute delayed versions of
            % the input x.
            xdly = obj.fracDly(x,dlySamp);
            % The resulting xdly should be nsamp x npath.

            % TODO:  Using the Doppler shifts, compute the phase rotations
            % on each path.  Specifically, if nsamp = length(x), create a
            % (nsamp+1) x npath matrix
            %     phase(i,k) = phase rotation on sample i and path k
            nsamp = length(x);
            t = ((0:nsamp)/obj.fsamp)';
            phase =  (t*obj.dop')*2*pi;
            % 2*pi*obj.fc*obj.dly
            % TODO:  Save the final phase, phase(nsamp+1,:)
            % as phaseInit for the next step.
            obj.phaseInit = phase(nsamp+1,:);

            % TODO:  Apply the phases and gain to each path, add the
            % resutls and store in y.
            y = sum(exp(1i*phase(1:end-1,:)).*xdly.*gainLin,2);
        end
    end
end
```

```
ans =

  SISOMPChan with properties:

        fsamp: []
         gain: []
          dly: []
          dop: []
           fc: []
      fracDly: []
    phaseInit: []
```

Published with MATLAB® R2020b