```
>> classdef GUI_V1_2_exported < matlab.apps.AppBase

    % Properties that correspond to app components
    properties (Access = public)
        UIFigure                      matlab.ui.Figure
        H1SwitchLabel                 matlab.ui.control.Label
        H1Switch                      matlab.ui.control.Switch
        H2SwitchLabel                 matlab.ui.control.Label
        H2Switch                      matlab.ui.control.Switch
        H3SwitchLabel                 matlab.ui.control.Label
        H3Switch                      matlab.ui.control.Switch
        H4SwitchLabel                 matlab.ui.control.Label
        H4Switch                      matlab.ui.control.Switch
        H5SwitchLabel                 matlab.ui.control.Label
        H5Switch                      matlab.ui.control.Switch
        H6SwitchLabel                 matlab.ui.control.Label
        H6Switch                      matlab.ui.control.Switch
        H7SwitchLabel                 matlab.ui.control.Label
        H7Switch                      matlab.ui.control.Switch
        H8SwitchLabel                 matlab.ui.control.Label
        H8Switch                      matlab.ui.control.Switch
        H9SwitchLabel                 matlab.ui.control.Label
        H9Switch                      matlab.ui.control.Switch
        IdealPlotButton               matlab.ui.control.Button
        ActualPlotButton              matlab.ui.control.Button
        FrequencyHzLabel              matlab.ui.control.Label
        FrequencyHzEditField          matlab.ui.control.NumericEditField
        AzimuthdegLimits90Label       matlab.ui.control.Label
        AzimuthdegLimits90EditField   matlab.ui.control.NumericEditField
        ElevationdegLimits90Label     matlab.ui.control.Label
        ElevationdegLimits90EditField  matlab.ui.control.NumericEditField
        SeparationDistanceEditFieldLabel  matlab.ui.control.Label
        SeparationDistanceEditField   matlab.ui.control.NumericEditField
        SeparationParameterListBoxLabel  matlab.ui.control.Label
        SeparationParameterListBox    matlab.ui.control.ListBox
    end

    % Callbacks that handle component events
    methods (Access = private)

        % Button pushed function: IdealPlotButton
```

```matlab
function IdealPlotButtonPushed(app, event)
    % Assign Frequency and Propagation Speed
    Frequency = app.FrequencyHzEditField.Value;
    PropagationSpeed = 1500;

    % Assign Azimuth Variation
    Az = app.AzimuthdegLimits90EditField.Value;
    % Assign Elevation Variation
    El = app.ElevationdegLimits90EditField.Value;

    % Assign Steering Angles (Az;El)
    SteeringAngles = [Az;El];

    % Specify Separation in Wavelengths (1.5) or Metres (1)
    SP = app.SeparationParameterListBox.Value;

    if      SP == 1
            SPX = 1.5;
    elseif  SP == 2
            SPX = 1;
    end

    % Specify Separation Distance (SD)
    SD = app.SeparationDistanceEditField.Value;
    a = 1;
    b = 1 + SD;
    c = 1 + (2 * SD);

    % Assign Phase shift quantization bits
    PhaseShiftBits = 0;

    % Create arbitrary geometry array
    Array = phased.ConformalArray();
    % The multiplication factor (.*) for lambda units to meter conversion
    Array.ElementPosition = [0 0 0 0 0 0 0 0 0;a b c a b c a b c;a a a b b b c c c] .* SPX;
    Array.ElementNormal = [0 0 0 0 0 0 0 0 0;0 0 0 0 0 0 0 0 0];
    Array.Taper = 1;

    % Create an isotropic hydrophone
    Elem = phased.IsotropicHydrophone ;
    Elem.VoltageSensitivity = -120;
```

```
            Elem.BackBaffled = true;
            Elem.FrequencyRange = [0 Frequency];
            Array.Element = Elem;

            % Create Ideal Figures

            % Plot Array Geometry
            figure('Name','Ideal Array Geometry','NumberTitle','off');
            viewArray(Array,'ShowNormal',true,...
              'ShowTaper',false,'ShowIndex','None');

            % Calculate Steering Weights

            Freq3D = Frequency;
            % Find the weights
            w = zeros(getNumElements(Array), length(Frequency));
            SteerVector = phased.SteeringVector('SensorArray', Array,...
             'PropagationSpeed', PropagationSpeed, 'NumPhaseShifterBits', PhaseShiftBits(1));
            for idx = 1:length(Frequency)
                w(:, idx) = step(SteerVector, Frequency(idx), SteeringAngles(:, idx));
            end

            % Plot 3d graph
            format = 'polar';
            figure('Name','Ideal Coverage','NumberTitle','off');
            pattern(Array, Freq3D , 'PropagationSpeed', PropagationSpeed,...
             'Type','directivity', 'CoordinateSystem', format,'weights', w(:,1));
        end

        % Button pushed function: ActualPlotButton
        function ActualPlotButtonPushed(app, event)
            Frequency = app.FrequencyHzEditField.Value;
            PropagationSpeed = 1500;

            Az = app.AzimuthdegLimits90EditField.Value;
            El = app.ElevationdegLimits90EditField.Value;

            SteeringAngles = [Az;El];

            SP = app.SeparationParameterListBox.Value;

            if      SP == 1
                    SPX = 1.5;
            elseif  SP == 2
                    SPX = 1;
            end

            SD = app.SeparationDistanceEditField.Value;

            a = 1;
            b = 1 + SD;
            c = 1 + (2 * SD);

            PhaseShiftBits = 0;

            H1 = app.H1Switch.Value;
            H2 = app.H2Switch.Value;
            H3 = app.H3Switch.Value;
            H4 = app.H4Switch.Value;
            H5 = app.H5Switch.Value;
            H6 = app.H6Switch.Value;
            H7 = app.H7Switch.Value;
            H8 = app.H8Switch.Value;
            H9 = app.H9Switch.Value;

% H1 OFF

            if (strcmp (H1, 'Off')) && (strcmp (H2, 'On')) && (strcmp (H3, 'On')) ...
            && (strcmp (H4, 'On')) && (strcmp (H5, 'On')) && (strcmp (H6, 'On')) ...
            && (strcmp (H7, 'On')) && (strcmp (H8, 'On')) && (strcmp (H9, 'On'))

            Array = phased.ConformalArray();
            Array.ElementPosition = [0 0 0 0 0 0 0 0;b c a b c a b c;a a b b b c c c] .* SPX;
            Array.ElementNormal = [0 0 0 0 0 0 0 0;0 0 0 0 0 0 0 0];
            Array.Taper = 1;

            Elem = phased.IsotropicHydrophone ;
            Elem.VoltageSensitivity = -120;
            Elem.BackBaffled = true;
            Elem.FrequencyRange = [0 Frequency];
            Array.Element = Elem;

```

```
      figure('Name','Actual Array Geometry','NumberTitle','off');
      viewArray(Array,'ShowNormal',true,...
        'ShowTaper',false,'ShowIndex','None');

      Freq3D = Frequency;
      w = zeros(getNumElements(Array), length(Frequency));
      SteerVector = phased.SteeringVector('SensorArray', Array,...
       'PropagationSpeed', PropagationSpeed, 'NumPhaseShifterBits', PhaseShiftBits(1));
      for idx = 1:length(Frequency)
          w(:, idx) = step(SteerVector, Frequency(idx), SteeringAngles(:, idx));
      end

      format = 'polar';
      figure('Name','Actual Coverage','NumberTitle','off');
      pattern(Array, Freq3D , 'PropagationSpeed', PropagationSpeed,...
       'Type','directivity', 'CoordinateSystem', format,'weights', w(:,1));

% H2 OFF

      elseif (strcmp (H1, 'On')) && (strcmp (H2, 'Off')) && (strcmp (H3, 'On')) ...
      && (strcmp (H4, 'On')) && (strcmp (H5, 'On')) && (strcmp (H6, 'On')) ...
      && (strcmp (H7, 'On')) && (strcmp (H8, 'On')) && (strcmp (H9, 'On'))

      Array = phased.ConformalArray();
      Array.ElementPosition = [0 0 0 0 0 0 0 0;a c a b c a b c;a a b b b c c c] .* SPX;
      Array.ElementNormal = [0 0 0 0 0 0 0 0;0 0 0 0 0 0 0 0];
      Array.Taper = 1;

      Elem = phased.IsotropicHydrophone ;
      Elem.VoltageSensitivity = -120;
      Elem.BackBaffled = true;
      Elem.FrequencyRange = [0 Frequency];
      Array.Element = Elem;

      figure('Name','Actual Array Geometry','NumberTitle','off');
      viewArray(Array,'ShowNormal',true,...
        'ShowTaper',false,'ShowIndex','None');

      Freq3D = Frequency;
      w = zeros(getNumElements(Array), length(Frequency));
      SteerVector = phased.SteeringVector('SensorArray', Array,...
       'PropagationSpeed', PropagationSpeed, 'NumPhaseShifterBits', PhaseShiftBits(1));
      for idx = 1:length(Frequency)
          w(:, idx) = step(SteerVector, Frequency(idx), SteeringAngles(:, idx));
      end

      format = 'polar';
      figure('Name','Actual Coverage','NumberTitle','off');
      pattern(Array, Freq3D , 'PropagationSpeed', PropagationSpeed,...
       'Type','directivity', 'CoordinateSystem', format,'weights', w(:,1));

% H3 OFF

      elseif (strcmp (H1, 'On')) && (strcmp (H2, 'On')) && (strcmp (H3, 'Off')) ...
      && (strcmp (H4, 'On')) && (strcmp (H5, 'On')) && (strcmp (H6, 'On')) ...
      && (strcmp (H7, 'On')) && (strcmp (H8, 'On')) && (strcmp (H9, 'On'))

      Array = phased.ConformalArray();
      Array.ElementPosition = [0 0 0 0 0 0 0 0;a b a b c a b c;a a b b b c c c] .* SPX;
      Array.ElementNormal = [0 0 0 0 0 0 0 0;0 0 0 0 0 0 0 0];
      Array.Taper = 1;

      Elem = phased.IsotropicHydrophone ;
      Elem.VoltageSensitivity = -120;
      Elem.BackBaffled = true;
      Elem.FrequencyRange = [0 Frequency];
      Array.Element = Elem;

      figure('Name','Actual Array Geometry','NumberTitle','off');
      viewArray(Array,'ShowNormal',true,...
        'ShowTaper',false,'ShowIndex','None');

      Freq3D = Frequency;
      w = zeros(getNumElements(Array), length(Frequency));
      SteerVector = phased.SteeringVector('SensorArray', Array,...
       'PropagationSpeed', PropagationSpeed, 'NumPhaseShifterBits', PhaseShiftBits(1));
      for idx = 1:length(Frequency)
          w(:, idx) = step(SteerVector, Frequency(idx), SteeringAngles(:, idx));
      end

      format = 'polar';
      figure('Name','Actual Coverage','NumberTitle','off');
```

```
            pattern(Array, Freq3D , 'PropagationSpeed', PropagationSpeed,...
             'Type','directivity', 'CoordinateSystem', format,'weights', w(:,1));

% H4 OFF

            elseif (strcmp (H1, 'On')) && (strcmp (H2, 'On')) && (strcmp (H3, 'On')) ...
            && (strcmp (H4, 'Off')) && (strcmp (H5, 'On')) && (strcmp (H6, 'On')) ...
            && (strcmp (H7, 'On')) && (strcmp (H8, 'On')) && (strcmp (H9, 'On'))

            Array = phased.ConformalArray();
            Array.ElementPosition = [0 0 0 0 0 0 0 0;a b c b c a b c;a a a b b c c c] .* SPX;
            Array.ElementNormal = [0 0 0 0 0 0 0 0;0 0 0 0 0 0 0 0];
            Array.Taper = 1;

            Elem = phased.IsotropicHydrophone ;
            Elem.VoltageSensitivity = -120;
            Elem.BackBaffled = true;
            Elem.FrequencyRange = [0 Frequency];
            Array.Element = Elem;

            figure('Name','Actual Array Geometry','NumberTitle','off');
            viewArray(Array,'ShowNormal',true,...
              'ShowTaper',false,'ShowIndex','None');

            Freq3D = Frequency;
            w = zeros(getNumElements(Array), length(Frequency));
            SteerVector = phased.SteeringVector('SensorArray', Array,...
             'PropagationSpeed', PropagationSpeed, 'NumPhaseShifterBits', PhaseShiftBits(1));
            for idx = 1:length(Frequency)
                w(:, idx) = step(SteerVector, Frequency(idx), SteeringAngles(:, idx));
            end

            format = 'polar';
            figure('Name','Actual Coverage','NumberTitle','off');
            pattern(Array, Freq3D , 'PropagationSpeed', PropagationSpeed,...
             'Type','directivity', 'CoordinateSystem', format,'weights', w(:,1));

% H5 OFF

            elseif (strcmp (H1, 'On')) && (strcmp (H2, 'On')) && (strcmp (H3, 'On')) ...
            && (strcmp (H4, 'On')) && (strcmp (H5, 'Off')) && (strcmp (H6, 'On')) ...
```

```matlab
            && (strcmp (H7, 'On')) && (strcmp (H8, 'On')) && (strcmp (H9, 'On'))

            Array = phased.ConformalArray();
            Array.ElementPosition = [0 0 0 0 0 0 0;a b c a c a b c;a a a b b c c c] .* SPX;
            Array.ElementNormal = [0 0 0 0 0 0 0;0 0 0 0 0 0 0];
            Array.Taper = 1;

            Elem = phased.IsotropicHydrophone ;
            Elem.VoltageSensitivity = -120;
            Elem.BackBaffled = true;
            Elem.FrequencyRange = [0 Frequency];
            Array.Element = Elem;

            figure('Name','Actual Array Geometry','NumberTitle','off');
            viewArray(Array,'ShowNormal',true,...
              'ShowTaper',false,'ShowIndex','None');

            Freq3D = Frequency;
            w = zeros(getNumElements(Array), length(Frequency));
            SteerVector = phased.SteeringVector('SensorArray', Array,...
             'PropagationSpeed', PropagationSpeed, 'NumPhaseShifterBits', PhaseShiftBits(1));
            for idx = 1:length(Frequency)
                w(:, idx) = step(SteerVector, Frequency(idx), SteeringAngles(:, idx));
            end

            format = 'polar';
            figure('Name','Actual Coverage','NumberTitle','off');
            pattern(Array, Freq3D , 'PropagationSpeed', PropagationSpeed,...
             'Type','directivity', 'CoordinateSystem', format,'weights', w(:,1));

% H6 OFF

            elseif (strcmp (H1, 'On')) && (strcmp (H2, 'On')) && (strcmp (H3, 'On')) ...
            && (strcmp (H4, 'On')) && (strcmp (H5, 'On')) && (strcmp (H6, 'Off')) ...
            && (strcmp (H7, 'On')) && (strcmp (H8, 'On')) && (strcmp (H9, 'On'))

            Array = phased.ConformalArray();
            Array.ElementPosition = [0 0 0 0 0 0 0;a b c a b a b c;a a a b b c c c] .* SPX;
            Array.ElementNormal = [0 0 0 0 0 0 0;0 0 0 0 0 0 0];
            Array.Taper = 1;

            Elem = phased.IsotropicHydrophone ;
            Elem.VoltageSensitivity = -120;
            Elem.BackBaffled = true;
            Elem.FrequencyRange = [0 Frequency];
            Array.Element = Elem;

            figure('Name','Actual Array Geometry','NumberTitle','off');
            viewArray(Array,'ShowNormal',true,...
              'ShowTaper',false,'ShowIndex','None');

            Freq3D = Frequency;
            w = zeros(getNumElements(Array), length(Frequency));
            SteerVector = phased.SteeringVector('SensorArray', Array,...
             'PropagationSpeed', PropagationSpeed, 'NumPhaseShifterBits', PhaseShiftBits(1));
            for idx = 1:length(Frequency)
                w(:, idx) = step(SteerVector, Frequency(idx), SteeringAngles(:, idx));
            end

            format = 'polar';
            figure('Name','Actual Coverage','NumberTitle','off');
            pattern(Array, Freq3D , 'PropagationSpeed', PropagationSpeed,...
             'Type','directivity', 'CoordinateSystem', format,'weights', w(:,1));

% H7 OFF

            elseif (strcmp (H1, 'On')) && (strcmp (H2, 'On')) && (strcmp (H3, 'On')) ...
            && (strcmp (H4, 'On')) && (strcmp (H5, 'On')) && (strcmp (H6, 'On')) ...
            && (strcmp (H7, 'Off')) && (strcmp (H8, 'On')) && (strcmp (H9, 'On'))

            Array = phased.ConformalArray();
            Array.ElementPosition = [0 0 0 0 0 0 0 0;a b c a b c b c;a a a b b b c c] .* SPX;
            Array.ElementNormal = [0 0 0 0 0 0 0 0;0 0 0 0 0 0 0 0];
            Array.Taper = 1;

            Elem = phased.IsotropicHydrophone ;
            Elem.VoltageSensitivity = -120;
            Elem.BackBaffled = true;
            Elem.FrequencyRange = [0 Frequency];
            Array.Element = Elem;

            figure('Name','Actual Array Geometry','NumberTitle','off');
            viewArray(Array,'ShowNormal',true,...
              'ShowTaper',false,'ShowIndex','None');

            Freq3D = Frequency;
            w = zeros(getNumElements(Array), length(Frequency));
            SteerVector = phased.SteeringVector('SensorArray', Array,...
             'PropagationSpeed', PropagationSpeed, 'NumPhaseShifterBits', PhaseShiftBits(1));
            for idx = 1:length(Frequency)
                w(:, idx) = step(SteerVector, Frequency(idx), SteeringAngles(:, idx));
            end

            format = 'polar';
            figure('Name','Actual Coverage','NumberTitle','off');
            pattern(Array, Freq3D , 'PropagationSpeed', PropagationSpeed,...
             'Type','directivity', 'CoordinateSystem', format,'weights', w(:,1));

% H8 OFF

            elseif (strcmp (H1, 'On')) && (strcmp (H2, 'On')) && (strcmp (H3, 'On')) ...
            && (strcmp (H4, 'On')) && (strcmp (H5, 'On')) && (strcmp (H6, 'On')) ...
            && (strcmp (H7, 'On')) && (strcmp (H8, 'Off')) && (strcmp (H9, 'On'))

            Array = phased.ConformalArray();
            Array.ElementPosition = [0 0 0 0 0 0 0 0;a b c a b c a c;a a a b b b c c] .* SPX;
            Array.ElementNormal = [0 0 0 0 0 0 0 0;0 0 0 0 0 0 0 0];
            Array.Taper = 1;

            Elem = phased.IsotropicHydrophone ;
            Elem.VoltageSensitivity = -120;
            Elem.BackBaffled = true;
            Elem.FrequencyRange = [0 Frequency];
            Array.Element = Elem;

            figure('Name','Actual Array Geometry','NumberTitle','off');
            viewArray(Array,'ShowNormal',true,...
              'ShowTaper',false,'ShowIndex','None');

            Freq3D = Frequency;
            w = zeros(getNumElements(Array), length(Frequency));
            SteerVector = phased.SteeringVector('SensorArray', Array,...
             'PropagationSpeed', PropagationSpeed, 'NumPhaseShifterBits', PhaseShiftBits(1));
            for idx = 1:length(Frequency)
                w(:, idx) = step(SteerVector, Frequency(idx), SteeringAngles(:, idx));
            end

            format = 'polar';
            figure('Name','Actual Coverage','NumberTitle','off');
            pattern(Array, Freq3D , 'PropagationSpeed', PropagationSpeed,...
             'Type','directivity', 'CoordinateSystem', format,'weights', w(:,1));

% H9 OFF

            elseif (strcmp (H1, 'On')) && (strcmp (H2, 'On')) && (strcmp (H3, 'On')) ...
            && (strcmp (H4, 'On')) && (strcmp (H5, 'On')) && (strcmp (H6, 'On')) ...
            && (strcmp (H7, 'On')) && (strcmp (H8, 'On')) && (strcmp (H9, 'Off'))

            Array = phased.ConformalArray();
            Array.ElementPosition = [0 0 0 0 0 0 0 0;a b c a b c a b;a a a b b b c c] .* SPX;
            Array.ElementNormal = [0 0 0 0 0 0 0 0;0 0 0 0 0 0 0 0];
            Array.Taper = 1;

            Elem = phased.IsotropicHydrophone ;
            Elem.VoltageSensitivity = -120;
            Elem.BackBaffled = true;
            Elem.FrequencyRange = [0 Frequency];
            Array.Element = Elem;

            figure('Name','Actual Array Geometry','NumberTitle','off');
            viewArray(Array,'ShowNormal',true,...
              'ShowTaper',false,'ShowIndex','None');

            Freq3D = Frequency;
            w = zeros(getNumElements(Array), length(Frequency));
            SteerVector = phased.SteeringVector('SensorArray', Array,...
             'PropagationSpeed', PropagationSpeed, 'NumPhaseShifterBits', PhaseShiftBits(1));
            for idx = 1:length(Frequency)
                w(:, idx) = step(SteerVector, Frequency(idx), SteeringAngles(:, idx));
            end

            format = 'polar';
            figure('Name','Actual Coverage','NumberTitle','off');
            pattern(Array, Freq3D , 'PropagationSpeed', PropagationSpeed,...
             'Type','directivity', 'CoordinateSystem', format,'weights', w(:,1));

% H1, H2, H3 OFF

            elseif (strcmp (H1, 'Off')) && (strcmp (H2, 'Off')) && (strcmp (H3, 'Off')) ...
            && (strcmp (H4, 'On')) && (strcmp (H5, 'On')) && (strcmp (H6, 'On')) ...
            && (strcmp (H7, 'On')) && (strcmp (H8, 'On')) && (strcmp (H9, 'On'))

            Array = phased.ConformalArray();
            Array.ElementPosition = [0 0 0 0 0 0;a b c a b c;b b b c c c] .* SPX;
            Array.ElementNormal = [0 0 0 0 0 0;0 0 0 0 0 0];
            Array.Taper = 1;

            Elem = phased.IsotropicHydrophone ;
            Elem.VoltageSensitivity = -120;
            Elem.BackBaffled = true;
            Elem.FrequencyRange = [0 Frequency];
            Array.Element = Elem;

            figure('Name','Actual Array Geometry','NumberTitle','off');
            viewArray(Array,'ShowNormal',true,...
              'ShowTaper',false,'ShowIndex','None');

            Freq3D = Frequency;
            w = zeros(getNumElements(Array), length(Frequency));
            SteerVector = phased.SteeringVector('SensorArray', Array,...
             'PropagationSpeed', PropagationSpeed, 'NumPhaseShifterBits', PhaseShiftBits(1));
            for idx = 1:length(Frequency)
                w(:, idx) = step(SteerVector, Frequency(idx), SteeringAngles(:, idx));
            end

            format = 'polar';
            figure('Name','Actual Coverage','NumberTitle','off');
            pattern(Array, Freq3D , 'PropagationSpeed', PropagationSpeed,...
             'Type','directivity', 'CoordinateSystem', format,'weights', w(:,1));

% H4, H5, H6 OFF

            elseif (strcmp (H1, 'On')) && (strcmp (H2, 'On')) && (strcmp (H3, 'On')) ...
            && (strcmp (H4, 'Off')) && (strcmp (H5, 'Off')) && (strcmp (H6, 'Off')) ...
            && (strcmp (H7, 'On')) && (strcmp (H8, 'On')) && (strcmp (H9, 'On'))

            Array = phased.ConformalArray();
            Array.ElementPosition = [0 0 0 0 0 0;a b c a b c;a a a c c c] .* SPX;
            Array.ElementNormal = [0 0 0 0 0 0;0 0 0 0 0 0];
            Array.Taper = 1;

            Elem = phased.IsotropicHydrophone ;
            Elem.VoltageSensitivity = -120;
            Elem.BackBaffled = true;
            Elem.FrequencyRange = [0 Frequency];
            Array.Element = Elem;

            figure('Name','Actual Array Geometry','NumberTitle','off');
            viewArray(Array,'ShowNormal',true,...
              'ShowTaper',false,'ShowIndex','None');

            Freq3D = Frequency;
            w = zeros(getNumElements(Array), length(Frequency));
            SteerVector = phased.SteeringVector('SensorArray', Array,...
             'PropagationSpeed', PropagationSpeed, 'NumPhaseShifterBits', PhaseShiftBits(1));
            for idx = 1:length(Frequency)
                w(:, idx) = step(SteerVector, Frequency(idx), SteeringAngles(:, idx));
            end

            format = 'polar';
            figure('Name','Actual Coverage','NumberTitle','off');
            pattern(Array, Freq3D , 'PropagationSpeed', PropagationSpeed,...
             'Type','directivity', 'CoordinateSystem', format,'weights', w(:,1));

% H7, H8, H9 OFF

            elseif (strcmp (H1, 'On')) && (strcmp (H2, 'On')) && (strcmp (H3, 'On')) ...
            && (strcmp (H4, 'On')) && (strcmp (H5, 'On')) && (strcmp (H6, 'On')) ...
            && (strcmp (H7, 'Off')) && (strcmp (H8, 'Off')) && (strcmp (H9, 'Off'))

            Array = phased.ConformalArray();
            Array.ElementPosition = [0 0 0 0 0 0;a b c a b c;a a a b b b] .* SPX;
            Array.ElementNormal = [0 0 0 0 0 0;0 0 0 0 0 0];
            Array.Taper = 1;

            Elem = phased.IsotropicHydrophone ;
```

```matlab
            Elem.VoltageSensitivity = -120;
            Elem.BackBaffled = true;
            Elem.FrequencyRange = [0 Frequency];
            Array.Element = Elem;

            figure('Name','Actual Array Geometry','NumberTitle','off');
            viewArray(Array,'ShowNormal',true,...
              'ShowTaper',false,'ShowIndex','None');

            Freq3D = Frequency;
            w = zeros(getNumElements(Array), length(Frequency));
            SteerVector = phased.SteeringVector('SensorArray', Array,...
             'PropagationSpeed', PropagationSpeed, 'NumPhaseShifterBits', PhaseShiftBits(1));
            for idx = 1:length(Frequency)
                w(:, idx) = step(SteerVector, Frequency(idx), SteeringAngles(:, idx));
            end

            format = 'polar';
            figure('Name','Actual Coverage','NumberTitle','off');
            pattern(Array, Freq3D , 'PropagationSpeed', PropagationSpeed,...
             'Type','directivity', 'CoordinateSystem', format,'weights', w(:,1));

% H1, H4, H7 OFF

            elseif (strcmp (H1, 'Off')) && (strcmp (H2, 'On')) && (strcmp (H3, 'On')) ...
            && (strcmp (H4, 'Off')) && (strcmp (H5, 'On')) && (strcmp (H6, 'On')) ...
            && (strcmp (H7, 'Off')) && (strcmp (H8, 'On')) && (strcmp (H9, 'On'))

            Array = phased.ConformalArray();
            Array.ElementPosition = [0 0 0 0 0 0;b c b c b c;a a b b c c] .* SPX;
            Array.ElementNormal = [0 0 0 0 0 0;0 0 0 0 0 0];
            Array.Taper = 1;

            Elem = phased.IsotropicHydrophone ;
            Elem.VoltageSensitivity = -120;
            Elem.BackBaffled = true;
            Elem.FrequencyRange = [0 Frequency];
            Array.Element = Elem;

            figure('Name','Actual Array Geometry','NumberTitle','off');
            viewArray(Array,'ShowNormal',true,...
              'ShowTaper',false,'ShowIndex','None');

            Freq3D = Frequency;
            w = zeros(getNumElements(Array), length(Frequency));
            SteerVector = phased.SteeringVector('SensorArray', Array,...
             'PropagationSpeed', PropagationSpeed, 'NumPhaseShifterBits', PhaseShiftBits(1));
            for idx = 1:length(Frequency)
                w(:, idx) = step(SteerVector, Frequency(idx), SteeringAngles(:, idx));
            end

            format = 'polar';
            figure('Name','Actual Coverage','NumberTitle','off');
            pattern(Array, Freq3D , 'PropagationSpeed', PropagationSpeed,...
             'Type','directivity', 'CoordinateSystem', format,'weights', w(:,1));

% H2, H5, H8 OFF

            elseif (strcmp (H1, 'On')) && (strcmp (H2, 'Off')) && (strcmp (H3, 'On')) ...
            && (strcmp (H4, 'On')) && (strcmp (H5, 'Off')) && (strcmp (H6, 'On')) ...
            && (strcmp (H7, 'On')) && (strcmp (H8, 'Off')) && (strcmp (H9, 'On'))

            Array = phased.ConformalArray();
            Array.ElementPosition = [0 0 0 0 0 0;a c a c a c;a a b b c c] .* SPX;
            Array.ElementNormal = [0 0 0 0 0 0;0 0 0 0 0 0];
            Array.Taper = 1;

            Elem = phased.IsotropicHydrophone ;
            Elem.VoltageSensitivity = -120;
            Elem.BackBaffled = true;
            Elem.FrequencyRange = [0 Frequency];
            Array.Element = Elem;

            figure('Name','Actual Array Geometry','NumberTitle','off');
            viewArray(Array,'ShowNormal',true,...
              'ShowTaper',false,'ShowIndex','None');

            Freq3D = Frequency;
            w = zeros(getNumElements(Array), length(Frequency));
            SteerVector = phased.SteeringVector('SensorArray', Array,...
             'PropagationSpeed', PropagationSpeed, 'NumPhaseShifterBits', PhaseShiftBits(1));
            for idx = 1:length(Frequency)
```

```matlab
                w(:, idx) = step(SteerVector, Frequency(idx), SteeringAngles(:, idx));
            end

            format = 'polar';
            figure('Name','Actual Coverage','NumberTitle','off');
            pattern(Array, Freq3D , 'PropagationSpeed', PropagationSpeed,...
             'Type','directivity', 'CoordinateSystem', format,'weights', w(:,1));

% H3, H6, H9 OFF

            elseif (strcmp (H1, 'On')) && (strcmp (H2, 'On')) && (strcmp (H3, 'Off')) ...
            && (strcmp (H4, 'On')) && (strcmp (H5, 'On')) && (strcmp (H6, 'Off')) ...
            && (strcmp (H7, 'On')) && (strcmp (H8, 'On')) && (strcmp (H9, 'Off'))

            Array = phased.ConformalArray();
            Array.ElementPosition = [0 0 0 0 0 0;a b a b a b;a a b b c c] .* SPX;
            Array.ElementNormal = [0 0 0 0 0 0;0 0 0 0 0 0];
            Array.Taper = 1;

            Elem = phased.IsotropicHydrophone ;
            Elem.VoltageSensitivity = -120;
            Elem.BackBaffled = true;
            Elem.FrequencyRange = [0 Frequency];
            Array.Element = Elem;

            figure('Name','Actual Array Geometry','NumberTitle','off');
            viewArray(Array,'ShowNormal',true,...
              'ShowTaper',false,'ShowIndex','None');

            Freq3D = Frequency;
            w = zeros(getNumElements(Array), length(Frequency));
            SteerVector = phased.SteeringVector('SensorArray', Array,...
             'PropagationSpeed', PropagationSpeed, 'NumPhaseShifterBits', PhaseShiftBits(1));
            for idx = 1:length(Frequency)
                w(:, idx) = step(SteerVector, Frequency(idx), SteeringAngles(:, idx));
            end

            format = 'polar';
            figure('Name','Actual Coverage','NumberTitle','off');
            pattern(Array, Freq3D , 'PropagationSpeed', PropagationSpeed,...
             'Type','directivity', 'CoordinateSystem', format,'weights', w(:,1));
            end
        end

        % Value changed function: AzimuthdegLimits90EditField
        function AzimuthdegLimits90EditFieldValueChanged(app, event)
            Az = app.AzimuthdegLimits90EditField.Value;
        end

        % Value changed function: ElevationdegLimits90EditField
        function ElevationdegLimits90EditFieldValueChanged(app, event)
            El = app.ElevationdegLimits90EditField.Value;
        end

        % Value changed function: FrequencyHzEditField
        function FrequencyHzEditFieldValueChanged(app, event)
            Frequency = app.FrequencyHzEditField.Value;
        end

        % Value changed function: SeparationParameterListBox
        function SeparationParameterListBoxValueChanged(app, event)
            SP = app.SeparationParameterListBox.Value;
        end

        % Value changed function: SeparationDistanceEditField
        function SeparationDistanceEditFieldValueChanged(app, event)
            SD = app.SeparationDistanceEditField.Value;
        end
    end

    % Component initialization
    methods (Access = private)

        % Create UIFigure and components
        function createComponents(app)

            % Create UIFigure and hide until all components are created
            app.UIFigure = uifigure('Visible', 'off');
            app.UIFigure.Position = [100 100 421 367];
            app.UIFigure.Name = 'MATLAB App';

```

```matlab
            % Create H1SwitchLabel
            app.H1SwitchLabel = uilabel(app.UIFigure);
            app.H1SwitchLabel.HorizontalAlignment = 'center';
            app.H1SwitchLabel.Position = [103 58 25 22];
            app.H1SwitchLabel.Text = 'H1';

            % Create H1Switch
            app.H1Switch = uiswitch(app.UIFigure, 'slider');
            app.H1Switch.Position = [97 81 37 16];
            app.H1Switch.Value = 'On';

            % Create H2SwitchLabel
            app.H2SwitchLabel = uilabel(app.UIFigure);
            app.H2SwitchLabel.HorizontalAlignment = 'center';
            app.H2SwitchLabel.Position = [203 58 25 22];
            app.H2SwitchLabel.Text = 'H2';

            % Create H2Switch
            app.H2Switch = uiswitch(app.UIFigure, 'slider');
            app.H2Switch.Position = [197 81 37 16];
            app.H2Switch.Value = 'On';

            % Create H3SwitchLabel
            app.H3SwitchLabel = uilabel(app.UIFigure);
            app.H3SwitchLabel.HorizontalAlignment = 'center';
            app.H3SwitchLabel.Position = [305 58 25 22];
            app.H3SwitchLabel.Text = 'H3';

            % Create H3Switch
            app.H3Switch = uiswitch(app.UIFigure, 'slider');
            app.H3Switch.Position = [299 81 37 16];
            app.H3Switch.Value = 'On';

            % Create H4SwitchLabel
            app.H4SwitchLabel = uilabel(app.UIFigure);
            app.H4SwitchLabel.HorizontalAlignment = 'center';
            app.H4SwitchLabel.Position = [103 106 25 22];
            app.H4SwitchLabel.Text = 'H4';

            % Create H4Switch
            app.H4Switch = uiswitch(app.UIFigure, 'slider');
            app.H4Switch.Position = [97 129 37 16];
            app.H4Switch.Value = 'On';

            % Create H5SwitchLabel
            app.H5SwitchLabel = uilabel(app.UIFigure);
            app.H5SwitchLabel.HorizontalAlignment = 'center';
            app.H5SwitchLabel.Position = [203 106 25 22];
            app.H5SwitchLabel.Text = 'H5';

            % Create H5Switch
            app.H5Switch = uiswitch(app.UIFigure, 'slider');
            app.H5Switch.Position = [197 129 37 16];
            app.H5Switch.Value = 'On';

            % Create H6SwitchLabel
            app.H6SwitchLabel = uilabel(app.UIFigure);
            app.H6SwitchLabel.HorizontalAlignment = 'center';
            app.H6SwitchLabel.Position = [305 106 25 22];
            app.H6SwitchLabel.Text = 'H6';

            % Create H6Switch
            app.H6Switch = uiswitch(app.UIFigure, 'slider');
            app.H6Switch.Position = [299 129 37 16];
            app.H6Switch.Value = 'On';

            % Create H7SwitchLabel
            app.H7SwitchLabel = uilabel(app.UIFigure);
            app.H7SwitchLabel.HorizontalAlignment = 'center';
            app.H7SwitchLabel.Position = [103 155 25 22];
            app.H7SwitchLabel.Text = 'H7';

            % Create H7Switch
            app.H7Switch = uiswitch(app.UIFigure, 'slider');
            app.H7Switch.Position = [97 178 37 16];
            app.H7Switch.Value = 'On';

            % Create H8SwitchLabel
            app.H8SwitchLabel = uilabel(app.UIFigure);
            app.H8SwitchLabel.HorizontalAlignment = 'center';
            app.H8SwitchLabel.Position = [203 155 25 22];
            app.H8SwitchLabel.Text = 'H8';

            % Create H8Switch
            app.H8Switch = uiswitch(app.UIFigure, 'slider');
            app.H8Switch.Position = [197 178 37 16];
            app.H8Switch.Value = 'On';

            % Create H9SwitchLabel
            app.H9SwitchLabel = uilabel(app.UIFigure);
            app.H9SwitchLabel.HorizontalAlignment = 'center';
            app.H9SwitchLabel.Position = [305 155 25 22];
            app.H9SwitchLabel.Text = 'H9';

            % Create H9Switch
            app.H9Switch = uiswitch(app.UIFigure, 'slider');
            app.H9Switch.Position = [299 178 37 16];
            app.H9Switch.Value = 'On';

            % Create IdealPlotButton
            app.IdealPlotButton = uibutton(app.UIFigure, 'push');
            app.IdealPlotButton.ButtonPushedFcn = createCallbackFcn(app, @IdealPlotButtonPushed, true);
            app.IdealPlotButton.FontWeight = 'bold';
            app.IdealPlotButton.Position = [97 28 100 22];
            app.IdealPlotButton.Text = 'Ideal Plot';

            % Create ActualPlotButton
            app.ActualPlotButton = uibutton(app.UIFigure, 'push');
            app.ActualPlotButton.ButtonPushedFcn = createCallbackFcn(app, @ActualPlotButtonPushed, true);
            app.ActualPlotButton.FontWeight = 'bold';
            app.ActualPlotButton.Position = [236 28 100 22];
            app.ActualPlotButton.Text = 'Actual Plot';

            % Create FrequencyHzLabel
            app.FrequencyHzLabel = uilabel(app.UIFigure);
            app.FrequencyHzLabel.HandleVisibility = 'callback';
            app.FrequencyHzLabel.FontWeight = 'bold';
            app.FrequencyHzLabel.Position = [19 319 92 22];
            app.FrequencyHzLabel.Text = 'Frequency (Hz)';

            % Create FrequencyHzEditField
            app.FrequencyHzEditField = uieditfield(app.UIFigure, 'numeric');
            app.FrequencyHzEditField.LowerLimitInclusive = 'off';
            app.FrequencyHzEditField.Limits = [0 Inf];
            app.FrequencyHzEditField.RoundFractionalValues = 'on';
            app.FrequencyHzEditField.ValueDisplayFormat = '%.0f';
            app.FrequencyHzEditField.ValueChangedFcn = createCallbackFcn(app, @FrequencyHzEditFieldValueChanged, true);
            app.FrequencyHzEditField.HandleVisibility = 'callback';
            app.FrequencyHzEditField.HorizontalAlignment = 'center';
            app.FrequencyHzEditField.Position = [119 319 85 22];
            app.FrequencyHzEditField.Value = 1000;

            % Create AzimuthdegLimits90Label
            app.AzimuthdegLimits90Label = uilabel(app.UIFigure);
            app.AzimuthdegLimits90Label.FontWeight = 'bold';
            app.AzimuthdegLimits90Label.Position = [18 269 87 28];
            app.AzimuthdegLimits90Label.Text = {'Azimuth (deg)'; '[Limits: +/- 90]'};

            % Create AzimuthdegLimits90EditField
            app.AzimuthdegLimits90EditField = uieditfield(app.UIFigure, 'numeric');
            app.AzimuthdegLimits90EditField.Limits = [-90 90];
            app.AzimuthdegLimits90EditField.ValueDisplayFormat = '%.0f';
            app.AzimuthdegLimits90EditField.ValueChangedFcn = createCallbackFcn(app, @AzimuthdegLimits90EditFieldValueChanged,
            app.AzimuthdegLimits90EditField.HandleVisibility = 'callback';
            app.AzimuthdegLimits90EditField.HorizontalAlignment = 'center';
            app.AzimuthdegLimits90EditField.Position = [119 275 85 22];

            % Create ElevationdegLimits90Label
            app.ElevationdegLimits90Label = uilabel(app.UIFigure);
            app.ElevationdegLimits90Label.FontWeight = 'bold';
            app.ElevationdegLimits90Label.Position = [19 225 92 28];
            app.ElevationdegLimits90Label.Text = {'Elevation (deg)'; '[Limits: +/- 90]'};

            % Create ElevationdegLimits90EditField
            app.ElevationdegLimits90EditField = uieditfield(app.UIFigure, 'numeric');
            app.ElevationdegLimits90EditField.Limits = [-90 90];
            app.ElevationdegLimits90EditField.ValueDisplayFormat = '%.0f';
            app.ElevationdegLimits90EditField.ValueChangedFcn = createCallbackFcn(app, @ElevationdegLimits90EditFieldValueChan
            app.ElevationdegLimits90EditField.HandleVisibility = 'callback';
            app.ElevationdegLimits90EditField.HorizontalAlignment = 'center';
            app.ElevationdegLimits90EditField.Position = [119 231 85 22];

            % Create SeparationDistanceEditFieldLabel
            app.SeparationDistanceEditFieldLabel = uilabel(app.UIFigure);
            app.SeparationDistanceEditFieldLabel.FontWeight = 'bold';
            app.SeparationDistanceEditFieldLabel.Position = [231 236 68 28];
            app.SeparationDistanceEditFieldLabel.Text = {'Separation'; 'Distance'};

            % Create SeparationDistanceEditField
            app.SeparationDistanceEditField = uieditfield(app.UIFigure, 'numeric');
            app.SeparationDistanceEditField.LowerLimitInclusive = 'off';
            app.SeparationDistanceEditField.Limits = [0 Inf];
            app.SeparationDistanceEditField.ValueDisplayFormat = '%.1f';
            app.SeparationDistanceEditField.ValueChangedFcn = createCallbackFcn(app, @SeparationDistanceEditFieldValueChanged,
            app.SeparationDistanceEditField.HorizontalAlignment = 'center';
            app.SeparationDistanceEditField.Position = [315 239 90 22];
            app.SeparationDistanceEditField.Value = 0.5;

            % Create SeparationParameterListBoxLabel
            app.SeparationParameterListBoxLabel = uilabel(app.UIFigure);
            app.SeparationParameterListBoxLabel.FontWeight = 'bold';
            app.SeparationParameterListBoxLabel.Position = [232 300 68 28];
            app.SeparationParameterListBoxLabel.Text = {'Separation'; 'Parameter'};

            % Create SeparationParameterListBox
            app.SeparationParameterListBox = uilistbox(app.UIFigure);
            app.SeparationParameterListBox.Items = {'Wavelength', 'Metres'};
            app.SeparationParameterListBox.ItemsData = [1 2];
            app.SeparationParameterListBox.ValueChangedFcn = createCallbackFcn(app, @SeparationParameterListBoxValueChanged, t
            app.SeparationParameterListBox.Position = [315 283 90 47];
            app.SeparationParameterListBox.Value = 1;

            % Show the figure after all components are created
            app.UIFigure.Visible = 'on';
        end
    end

    % App creation and deletion
    methods (Access = public)

        % Construct app
        function app = GUI_V1_2_exported

            % Create UIFigure and components
            createComponents(app)
```

```matlab

            % Register the app with App Designer
            registerApp(app, app.UIFigure)

            if nargout == 0
                clear app
            end
        end

        % Code that executes before app deletion
        function delete(app)

            % Delete UIFigure when app is deleted
            delete(app.UIFigure)
        end
    end
end
```