

```
1 >> classdef GUI_V1_2_exported < matlab.apps.AppBase
2
3 % Properties that correspond to app components
4 properties (Access = public)
5     UIFigure
6     H1SwitchLabel
7     H1Switch
8     H2SwitchLabel
9     H2Switch
10    H3SwitchLabel
11    H3Switch
12    H4SwitchLabel
13    H4Switch
14    H5SwitchLabel
15    H5Switch
16    H6SwitchLabel
17    H6Switch
18    H7SwitchLabel
19    H7Switch
20    H8SwitchLabel
21    H8Switch
22    H9SwitchLabel
23    H9Switch
24    IdealPlotButton
25    ActualPlotButton
26    FrequencyHzLabel
27    FrequencyHzEditField
28    AzimuthdegLimits90Label
29    AzimuthdegLimits90EditField
30    ElevationdegLimits90Label
31    ElevationdegLimits90EditField
32    SeparationDistanceEditFieldLabel
33    SeparationDistanceEditField
34    SeparationParameterListBoxLabel
35    SeparationParameterListBox
36 end
37
38 % Callbacks that handle component events
39 methods (Access = private)
40
41     % Button pushed function: IdealPlotButton
        matlab.ui.Figure
        matlab.ui.control.Label
        matlab.ui.control.Switch
        matlab.ui.control.Label
        matlab.ui.control.Switch
        matlab.ui.control.Label
        matlab.ui.control.Switch
        matlab.ui.control.Label
        matlab.ui.control.Switch
        matlab.ui.control.Label
        matlab.ui.control.Switch
        matlab.ui.control.Label
        matlab.ui.control.Switch
        matlab.ui.control.Label
        matlab.ui.control.Switch
        matlab.ui.control.Label
        matlab.ui.control.Switch
        matlab.ui.control.Label
        matlab.ui.control.Button
        matlab.ui.control.Button
        matlab.ui.control.Label
        matlab.ui.control.NumericEditField
        matlab.ui.control.Label
        matlab.ui.control.NumericEditField
        matlab.ui.control.Label
        matlab.ui.control.NumericEditField
        matlab.ui.control.Label
        matlab.ui.control.NumericEditField
        matlab.ui.control.Label
        matlab.ui.control.NumericEditField
        matlab.ui.control.Label
        matlab.ui.control.ListBox
        matlab.ui.control.ListBox
    end
end
```

```

42 function IdealPlotButtonPushed(app, event)
43     % Assign Frequency and Propagation Speed
44     Frequency = app.FrequencyHzEditField.Value;
45     PropagationSpeed = 1500;
46
47     % Assign Azimuth Variation
48     Az = app.AzimuthdegLimits90EditField.Value;
49     % Assign Elevation Variation
50     El = app.ElevationdegLimits90EditField.Value;
51
52     % Assign Steering Angles (Az;El)
53     SteeringAngles = [Az;El];
54
55     % Specify Separation in Wavelengths (1.5) or Metres (1)
56     SP = app.SeparationParameterListBox.Value;
57
58     if SP == 1
59         SPX = 1.5;
60     elseif SP == 2
61         SPX = 1;
62     end
63
64     % Specify Separation Distance (SD)
65     SD = app.SeparationDistanceEditField.Value;
66     a = 1;
67     b = 1 + SD;
68     c = 1 + (2 * SD);
69
70     % Assign Phase shift quantization bits
71     PhaseShiftBits = 0;
72
73     % Create arbitrary geometry array
74     Array = phased.ConformalArray();
75     % The multiplication factor (.) for lambda units to meter conversion
76     Array.ElementPosition = [0 0 0 0 0 0 0;a b c a b c;a a b b c c] .* SPX;
77     Array.ElementNormal = [0 0 0 0 0 0 0;0 0 0 0 0 0 0];
78     Array.Taper = 1;
79
80     % Create an isotropic hydrophone
81     Elem = phased.IsotropicHydrophone ;
82     Elem.VoltageSensitivity = -120;

```

```

83 Elem.BackBaffled = true;
84 Elem.FrequencyRange = [0 Frequency];
85 Array.Element = Elem;
86
87 % Create Ideal Figures
88
89 % Plot Array Geometry
90 figure('Name','Ideal Array Geometry','NumberTitle','off');
91 viewArray(Array,'ShowNormal',true,...
92 'ShowTaper',false,'ShowIndex','None');
93
94 % Calculate Steering Weights
95
96 Freq3D = Frequency;
97 % Find the weights
98 w = zeros(getNumElements(Array), length(Frequency));
99 SteerVector = phased.SteeringVector('SensorArray', Array,...
100 'PropagationSpeed', PropagationSpeed, 'NumPhaseShifterBits', PhaseShiftBits(1));
101 for idx = 1:length(Frequency)
102     w(:, idx) = step(SteerVector, Frequency(idx), SteeringAngles(:, idx));
103 end
104
105 % Plot 3d graph
106 format = 'polar';
107 figure('Name','Ideal Coverage','NumberTitle','off');
108 pattern(Array, Freq3D, 'PropagationSpeed', PropagationSpeed,...
109 'Type','directivity', 'CoordinatesSystem', format,'weights', w(:,1));
110 end
111
112 % Button pushed function: ActualPlotButton
113 function ActualPlotButtonPushed(app, event)
114     Frequency = app.FrequencyHzEditField.Value;
115     PropagationSpeed = 1500;
116
117     Az = app.AzimuthdegLimits90EditField.Value;
118     El = app.ElevationdegLimits90EditField.Value;
119
120     SteeringAngles = [Az;El];
121
122     SP = app.SeparationParameterListBox.Value;
123

```

```
124         SP == 1
125         SPX = 1.5;
126     elseif SP == 2
127         SPX = 1;
128     end
129
130     SD = app.SeparationDistanceEditField.Value;
131
132     a = 1;
133     b = 1 + SD;
134     c = 1 + (2 * SD);
135
136     PhaseShiftBits = 0;
137
138     H1 = app.H1Switch.Value;
139     H2 = app.H2Switch.Value;
140     H3 = app.H3Switch.Value;
141     H4 = app.H4Switch.Value;
142     H5 = app.H5Switch.Value;
143     H6 = app.H6Switch.Value;
144     H7 = app.H7Switch.Value;
145     H8 = app.H8Switch.Value;
146     H9 = app.H9Switch.Value;
147
148     % H1 OFF
149
150     if (strcmp (H1, 'Off')) && (strcmp (H2, 'On')) && (strcmp (H3, 'On')) ...
151         && (strcmp (H4, 'On')) && (strcmp (H5, 'On')) && (strcmp (H6, 'On')) ...
152         && (strcmp (H7, 'On')) && (strcmp (H8, 'On')) && (strcmp (H9, 'On'))
153
154         Array = phased.ConformalArray();
155         Array.ElementPosition = [0 0 0 0 0 0 0 0;b c a b c a b b c c c] .* SPX;
156         Array.ElementNormal = [0 0 0 0 0 0 0 0;0 0 0 0 0 0 0 0];
157         Array.Taper = 1;
158
159         Elem = phased.IsotropicHydrophone ;
160         Elem.VoltageSensitivity = -120;
161         Elem.BackBaffled = true;
162         Elem.FrequencyRange = [0 Frequency];
163         Array.Element = Elem;
164
```

```

165 figure('Name','Actual Array Geometry','NumberTitle','off');
166 viewArray(Array,'ShowNormal',true,...
167     'ShowTaper',false,'ShowIndex','None');
168
169 Freq3D = Frequency;
170 w = zeros(getNumElements(Array), length(Frequency));
171 SteerVector = phased.SteeringVector('SensorArray', Array,...
172     'PropagationSpeed', PropagationSpeed, 'NumPhaseShifterBits', PhaseShiftBits(1));
173 for idx = 1:length(Frequency)
174     w(:, idx) = step(SteerVector, Frequency(idx), SteeringAngles(:, idx));
175 end
176
177 format = 'polar';
178 figure('Name','Actual Coverage','NumberTitle','off');
179 pattern(Array, Freq3D, 'PropagationSpeed', PropagationSpeed,...
180     'Type','directivity', 'CoordinatesSystem', format,'weights', w(:,1));
181
182 % H2 OFF
183
184 elseif (strcmp (H1, 'On')) && (strcmp (H2, 'Off')) && (strcmp (H3, 'On')) ...
185     && (strcmp (H4, 'On')) && (strcmp (H5, 'On')) && (strcmp (H6, 'On')) ...
186     && (strcmp (H7, 'On')) && (strcmp (H8, 'On')) && (strcmp (H9, 'On'))
187
188 Array = phased.ConformalArray();
189 Array.ElementPosition = [0 0 0 0 0 0 0;a c a b c a b b c c c] .* SPX;
190 Array.ElementNormal = [0 0 0 0 0 0 0;0 0 0 0 0 0 0 0 0 0];
191 Array.Taper = 1;
192
193 Elem = phased.IsotropicHydrophone ;
194 Elem.VoltageSensitivity = -120;
195 Elem.BackBaffled = true;
196 Elem.FrequencyRange = [0 Frequency];
197 Array.Element = Elem;
198
199 figure('Name','Actual Array Geometry','NumberTitle','off');
200 viewArray(Array,'ShowNormal',true,...
201     'ShowTaper',false,'ShowIndex','None');
202
203 Freq3D = Frequency;
204 w = zeros(getNumElements(Array), length(Frequency));
205 SteerVector = phased.SteeringVector('SensorArray', Array,...

```

```

206 'PropagationSpeed', PropagationSpeed, 'NumPhaseShifterBits', PhaseShiftBits(1));
207 for idx = 1:length(Frequency)
208     w(:, idx) = step(SteerVector, Frequency(idx), SteeringAngles(:, idx));
209 end
210
211 format = 'polar';
212 figure('Name','Actual Coverage','NumberTitle','off');
213 pattern(Array, Freq3D, 'PropagationSpeed', PropagationSpeed,...
214 'Type','directivity', 'CoordinatesSystem', format,'weights', w(:,1));
215
216 % H3 OFF
217
218 elseif (strcmp (H1, 'On')) && (strcmp (H2, 'On')) && (strcmp (H3, 'Off')) ...
219 && (strcmp (H4, 'On')) && (strcmp (H5, 'On')) && (strcmp (H6, 'On')) ...
220 && (strcmp (H7, 'On')) && (strcmp (H8, 'On')) && (strcmp (H9, 'On'))
221
222 Array = phased.ConformalArray();
223 Array.ElementPosition = [0 0 0 0 0 0 0;a b c a b c a b c c] .* SPX;
224 Array.ElementNormal = [0 0 0 0 0 0 0;0 0 0 0 0 0 0];
225 Array.Taper = 1;
226
227 Elem = phased.IsotropicHydrophone ;
228 Elem.VoltageSensitivity = -120;
229 Elem.BackBaffled = true;
230 Elem.FrequencyRange = [0 Frequency];
231 Array.Element = Elem;
232
233 figure('Name','Actual Array Geometry','NumberTitle','off');
234 viewArray(Array,'ShowNormal',true,...
235 'ShowTaper',false,'ShowIndex','None');
236
237 Freq3D = Frequency;
238 w = zeros(getNumElements(Array), length(Frequency));
239 SteerVector = phased.SteeringVector('SensorArray', Array,...
240 'PropagationSpeed', PropagationSpeed, 'NumPhaseShifterBits', PhaseShiftBits(1));
241 for idx = 1:length(Frequency)
242     w(:, idx) = step(SteerVector, Frequency(idx), SteeringAngles(:, idx));
243 end
244
245 format = 'polar';
246 figure('Name','Actual Coverage','NumberTitle','off');

```

```

247 pattern(Array, Freq3D , 'PropagationSpeed', PropagationSpeed,...
248 'Type','directivity', 'CoordinatesSystem', format,'weights', w(:,1));
249
250 % H4 OFF
251
252 elseif (strcmp (H1, 'On')) && (strcmp (H2, 'On')) && (strcmp (H3, 'On')) ...
253 && (strcmp (H4, 'Off')) && (strcmp (H5, 'On')) && (strcmp (H6, 'On')) ...
254 && (strcmp (H7, 'On')) && (strcmp (H8, 'On')) && (strcmp (H9, 'On'))
255
256 Array = phased.ConformalArray();
257 Array.ElementPosition = [0 0 0 0 0 0 0;a b c b c a b c c c] .* SPX;
258 Array.ElementNormal = [0 0 0 0 0 0 0;0 0 0 0 0 0 0];
259 Array.Taper = 1;
260
261 Elem = phased.IsotropicHydrophone ;
262 Elem.VoltageSensitivity = -120;
263 Elem.BackBaffled = true;
264 Elem.FrequencyRange = [0 Frequency];
265 Array.Element = Elem;
266
267 figure('Name','Actual Array Geometry','NumberTitle','off');
268 viewArray(Array,'ShowNormal',true,...
269 'ShowTaper',false,'ShowIndex','None');
270
271 Freq3D = Frequency;
272 w = zeros(getNumElements(Array), length(Frequency));
273 SteerVector = phased.SteeringVector('SensorArray', Array,...
274 'PropagationSpeed', PropagationSpeed, 'NumPhaseShifterBits', PhaseShiftBits(1));
275 for idx = 1:length(Frequency)
276     w(:, idx) = step(SteerVector, Frequency(idx), SteeringAngles(:, idx));
277 end
278
279 format = 'polar';
280 figure('Name','Actual Coverage','NumberTitle','off');
281 pattern(Array, Freq3D , 'PropagationSpeed', PropagationSpeed,...
282 'Type','directivity', 'CoordinatesSystem', format,'weights', w(:,1));
283
284 % H5 OFF
285
286 elseif (strcmp (H1, 'On')) && (strcmp (H2, 'On')) && (strcmp (H3, 'On')) ...
287 && (strcmp (H4, 'On')) && (strcmp (H5, 'Off')) && (strcmp (H6, 'On')) ...

```

```

288 && (strcmp (H7, 'On')) && (strcmp (H8, 'On')) && (strcmp (H9, 'On'))
289
290 Array = phased.ConformalArray();
291 Array.ElementPosition = [0 0 0 0 0 0 0;a b c a b c c] .* SPX;
292 Array.ElementNormal = [0 0 0 0 0 0 0;0 0 0 0 0 0 0];
293 Array.Taper = 1;
294
295 Elem = phased.IsotropicHydrophone ;
296 Elem.VoltageSensitivity = -120;
297 Elem.BackBaffled = true;
298 Elem.FrequencyRange = [0 Frequency];
299 Array.Element = Elem;
300
301 figure('Name','Actual Array Geometry','NumberTitle','off');
302 viewArray(Array,'ShowNormal',true,...
303 'ShowTaper',false,'ShowIndex','None');
304
305 Freq3D = Frequency;
306 w = zeros(getNumElements(Array), length(Frequency));
307 SteerVector = phased.SteeringVector('SensorArray', Array,...
308 'PropagationSpeed', PropagationSpeed, 'NumPhaseShifterBits', PhaseShiftBits(1));
309 for idx = 1:length(Frequency)
310     w(:, idx) = step(SteerVector, Frequency(idx), SteeringAngles(:, idx));
311 end
312
313 format = 'polar';
314 figure('Name','Actual Coverage','NumberTitle','off');
315 pattern(Array, Freq3D, 'PropagationSpeed', PropagationSpeed,...
316 'Type','directivity', 'CoordinatesSystem', format,'weights', w(:,1));
317
318 % H6 OFF
319
320 elseif (strcmp (H1, 'On')) && (strcmp (H2, 'On')) && (strcmp (H3, 'On')) ...
321 && (strcmp (H4, 'On')) && (strcmp (H5, 'On')) && (strcmp (H6, 'Off')) ...
322 && (strcmp (H7, 'On')) && (strcmp (H8, 'On')) && (strcmp (H9, 'On'))
323
324 Array = phased.ConformalArray();
325 Array.ElementPosition = [0 0 0 0 0 0 0;a b c a b c c] .* SPX;
326 Array.ElementNormal = [0 0 0 0 0 0 0;0 0 0 0 0 0 0];
327 Array.Taper = 1;
328

```



```

329 Elem = phased.IsotropicHydrophone ;
330 Elem.VoltageSensitivity = -120;
331 Elem.BackBaffled = true;
332 Elem.FrequencyRange = [0 Frequency];
333 Array.Element = Elem;
334
335 figure('Name','Actual Array Geometry','NumberTitle','off');
336 viewArray(Array,'ShowNormal',true,...
337     'ShowTaper',false,'ShowIndex','None');
338
339 Freq3D = Frequency;
340 w = zeros(getNumElements(Array), length(Frequency));
341 SteerVector = phased.SteeringVector('SensorArray', Array,...
342     'PropagationSpeed', PropagationSpeed, 'NumPhaseShifterBits', PhaseShiftBits(1));
343 for idx = 1:length(Frequency)
344     w(:, idx) = step(SteerVector, Frequency(idx), SteeringAngles(:, idx));
345 end
346
347 format = 'polar';
348 figure('Name','Actual Coverage','NumberTitle','off');
349 pattern(Array, Freq3D, 'PropagationSpeed', PropagationSpeed,...
350     'Type','directivity', 'CoordinatesSystem', format,'weights', w(:,1));
351
352 % H7 OFF
353
354 elseif (strcmp (H1, 'On')) && (strcmp (H2, 'On')) && (strcmp (H3, 'On')) ...
355     && (strcmp (H4, 'On')) && (strcmp (H5, 'On')) && (strcmp (H6, 'On')) ...
356     && (strcmp (H7, 'Off')) && (strcmp (H8, 'On')) && (strcmp (H9, 'On'))
357
358 Array = phased.ConformalArray();
359 Array.ElementPosition = [0 0 0 0 0 0 0;a b c a b b c c] .* SPX;
360 Array.ElementNormal = [0 0 0 0 0 0 0;0 0 0 0 0 0 0];
361 Array.Taper = 1;
362
363 Elem = phased.IsotropicHydrophone ;
364 Elem.VoltageSensitivity = -120;
365 Elem.BackBaffled = true;
366 Elem.FrequencyRange = [0 Frequency];
367 Array.Element = Elem;
368
369 figure('Name','Actual Array Geometry','NumberTitle','off');

```

```

370 viewArray(Array,'ShowNormal',true,...
371 'ShowTaper',false,'ShowIndex','None');
372
373 Freq3D = Frequency;
374 w = zeros(getNumElements(Array), length(Frequency));
375 SteerVector = phased.SteeringVector('SensorArray', Array,...
376 'PropagationSpeed', PropagationSpeed, 'NumPhaseShifterBits', PhaseShiftBits(1));
377 for idx = 1:length(Frequency)
378     w(:, idx) = step(SteerVector, Frequency(idx), SteeringAngles(:, idx));
379 end
380
381 format = 'polar';
382 figure('Name','Actual Coverage','NumberTitle','off');
383 pattern(Array, Freq3D, 'PropagationSpeed', PropagationSpeed,...
384 'Type','directivity', 'CoordinatesSystem', format,'weights', w(:,1));
385
386 % H8 OFF
387
388 elseif (strcmp (H1, 'On')) && (strcmp (H2, 'On')) && (strcmp (H3, 'On')) ...
389 && (strcmp (H4, 'On')) && (strcmp (H5, 'On')) && (strcmp (H6, 'On')) ...
390 && (strcmp (H7, 'On')) && (strcmp (H8, 'Off')) && (strcmp (H9, 'On'))
391
392 Array = phased.ConformalArray();
393 Array.ElementPosition = [0 0 0 0 0 0 0;a b c a b b c c] .* SPX;
394 Array.ElementNormal = [0 0 0 0 0 0 0;0 0 0 0 0 0 0];
395 Array.Taper = 1;
396
397 Elem = phased.IsotropicHydrophone ;
398 Elem.VoltageSensitivity = -120;
399 Elem.BackBaffled = true;
400 Elem.FrequencyRange = [0 Frequency];
401 Array.Element = Elem;
402
403 figure('Name','Actual Array Geometry','NumberTitle','off');
404 viewArray(Array,'ShowNormal',true,...
405 'ShowTaper',false,'ShowIndex','None');
406
407 Freq3D = Frequency;
408 w = zeros(getNumElements(Array), length(Frequency));
409 SteerVector = phased.SteeringVector('SensorArray', Array,...
410 'PropagationSpeed', PropagationSpeed, 'NumPhaseShifterBits', PhaseShiftBits(1));

```

```

411 for idx = 1:length(Frequency)
412     w(:, idx) = step(SteerVector, Frequency(idx), SteeringAngles(:, idx));
413 end
414
415 format = 'polar';
416 figure('Name','Actual Coverage','NumberTitle','off');
417 pattern(Array, Freq3D , 'PropagationSpeed', PropagationSpeed,...
418         'Type','directivity', 'CoordinatesSystem', format,'weights', w(:,1));
419
420 % H9 OFF
421
422 elseif (strcmp (H1, 'On')) && (strcmp (H2, 'On')) && (strcmp (H3, 'On')) ...
423     && (strcmp (H4, 'On')) && (strcmp (H5, 'On')) && (strcmp (H6, 'On')) ...
424     && (strcmp (H7, 'On')) && (strcmp (H8, 'On')) && (strcmp (H9, 'Off'))
425
426 Array = phased.ConformalArray();
427 Array.ElementPosition = [0 0 0 0 0 0 0;a b c a b c a b b c c] .* SPX;
428 Array.ElementNormal = [0 0 0 0 0 0 0;0 0 0 0 0 0 0 0 0 0];
429 Array.Taper = 1;
430
431 Elem = phased.IsotropicHydrophone ;
432 Elem.VoltageSensitivity = -120;
433 Elem.BackBaffled = true;
434 Elem.FrequencyRange = [0 Frequency];
435 Array.Element = Elem;
436
437 figure('Name','Actual Array Geometry','NumberTitle','off');
438 viewArray(Array,'ShowNormal',true,...
439         'ShowTaper',false,'ShowIndex','None');
440
441 Freq3D = Frequency;
442 w = zeros(getNumElements(Array), length(Frequency));
443 SteerVector = phased.SteeringVector('SensorArray', Array,...
444         'PropagationSpeed', PropagationSpeed, 'NumPhaseShifterBits', PhaseShiftBits(1));
445 for idx = 1:length(Frequency)
446     w(:, idx) = step(SteerVector, Frequency(idx), SteeringAngles(:, idx));
447 end
448
449 format = 'polar';
450 figure('Name','Actual Coverage','NumberTitle','off');
451 pattern(Array, Freq3D , 'PropagationSpeed', PropagationSpeed,...

```

```

452     'Type','directivity', 'CoordinatesSystem', format,'weights', w(:,1));
453
454     % H1, H2, H3 OFF
455
456     elseif (strcmp (H1, 'Off')) && (strcmp (H2, 'Off')) && (strcmp (H3, 'Off')) ...
457     && (strcmp (H4, 'On')) && (strcmp (H5, 'On')) && (strcmp (H6, 'On')) ...
458     && (strcmp (H7, 'On')) && (strcmp (H8, 'On')) && (strcmp (H9, 'On'))
459
460     Array = phased.ConformalArray();
461     Array.ElementPosition = [0 0 0 0 0;a b c a b c;b b c c c] .* SPX;
462     Array.ElementNormal = [0 0 0 0 0;0 0 0 0 0 0];
463     Array.Taper = 1;
464
465     Elem = phased.IsotropicHydrophone ;
466     Elem.VoltageSensitivity = -120;
467     Elem.BackBaffled = true;
468     Elem.FrequencyRange = [0 Frequency];
469     Array.Element = Elem;
470
471     figure('Name','Actual Array Geometry','NumberTitle','off');
472     viewArray(Array,'ShowNormal',true,...
473     'ShowTaper',false,'ShowIndex','None');
474
475     Freq3D = Frequency;
476     w = zeros(getNumElements(Array), length(Frequency));
477     SteerVector = phased.SteeringVector('SensorArray', Array,...
478     'PropagationSpeed', PropagationSpeed, 'NumPhaseShifterBits', PhaseShiftBits(1));
479     for idx = 1:length(Frequency)
480         w(:, idx) = step(SteerVector, Frequency(idx), SteeringAngles(:, idx));
481     end
482
483     format = 'polar';
484     figure('Name','Actual Coverage','NumberTitle','off');
485     pattern(Array, Freq3D, 'PropagationSpeed', PropagationSpeed,...
486     'Type','directivity', 'CoordinatesSystem', format,'weights', w(:,1));
487
488     % H4, H5, H6 OFF
489
490     elseif (strcmp (H1, 'On')) && (strcmp (H2, 'On')) && (strcmp (H3, 'On')) ...
491     && (strcmp (H4, 'Off')) && (strcmp (H5, 'Off')) && (strcmp (H6, 'Off')) ...
492     && (strcmp (H7, 'On')) && (strcmp (H8, 'On')) && (strcmp (H9, 'On'))

```

```

493 Array = phased.ConformalArray();
494 Array.ElementPosition = [0 0 0 0 0;a b c;a b c c] .* SPX;
495 Array.ElementNormal = [0 0 0 0 0;0 0 0 0 0];
496 Array.Taper = 1;
497
498
499 Elem = phased.IsotropicHydrophone ;
500 Elem.VoltageSensitivity = -120;
501 Elem.BackBaffled = true;
502 Elem.FrequencyRange = [0 Frequency];
503 Array.Element = Elem;
504
505 figure('Name','Actual Array Geometry','NumberTitle','off');
506 viewArray(Array,'ShowNormal',true,...
507     'ShowTaper',false,'ShowIndex','None');
508
509
510 Freq3D = Frequency;
511 w = zeros(getNumElements(Array), length(Frequency));
512 SteerVector = phased.SteeringVector('SensorArray', Array,...
513     'PropagationSpeed', PropagationSpeed, 'NumPhaseShifterBits', PhaseShiftBits(1));
514 for idx = 1:length(Frequency)
515     w(:, idx) = step(SteerVector, Frequency(idx), SteeringAngles(:, idx));
516 end
517
518 format = 'polar';
519 figure('Name','Actual Coverage','NumberTitle','off');
520 pattern(Array, Freq3D, 'PropagationSpeed', PropagationSpeed,...
521     'Type','directivity', 'CoordinatesSystem', format,'weights', w(:,1));
522 % H7, H8, H9 OFF
523
524 elseif (strcmp (H1, 'On')) && (strcmp (H2, 'On')) && (strcmp (H3, 'On')) ...
525     && (strcmp (H4, 'On')) && (strcmp (H5, 'On')) && (strcmp (H6, 'On')) ...
526     && (strcmp (H7, 'Off')) && (strcmp (H8, 'Off')) && (strcmp (H9, 'Off'))
527
528 Array = phased.ConformalArray();
529 Array.ElementPosition = [0 0 0 0 0;a b c;a b c;a b b] .* SPX;
530 Array.ElementNormal = [0 0 0 0 0;0 0 0 0 0];
531 Array.Taper = 1;
532
533 Elem = phased.IsotropicHydrophone ;

```

```

534 Elem.VoltageSensitivity = -120;
535 Elem.BackBaffled = true;
536 Elem.FrequencyRange = [0 Frequency];
537 Array.Element = Elem;
538
539 figure('Name','Actual Array Geometry','NumberTitle','off');
540 viewArray(Array,'ShowNormal',true,...
541 'ShowTaper',false,'ShowIndex','None');
542
543 Freq3D = Frequency;
544 w = zeros(getNumElements(Array), length(Frequency));
545 SteerVector = phased.SteeringVector('SensorArray', Array,...
546 'PropagationSpeed', PropagationSpeed, 'NumPhaseShifterBits', PhaseShiftBits(1));
547 for idx = 1:length(Frequency)
548     w(:, idx) = step(SteerVector, Frequency(idx), SteeringAngles(:, idx));
549 end
550
551 format = 'polar';
552 figure('Name','Actual Coverage','NumberTitle','off');
553 pattern(Array, Freq3D, 'PropagationSpeed', PropagationSpeed,...
554 'Type','directivity', 'CoordinatesSystem', format,'weights', w(:,1));
555
556 % H1, H4, H7 OFF
557
558 elseif (strcmp (H1, 'Off')) && (strcmp (H2, 'On')) && (strcmp (H3, 'On')) ...
559 && (strcmp (H4, 'Off')) && (strcmp (H5, 'On')) && (strcmp (H6, 'On')) ...
560 && (strcmp (H7, 'Off')) && (strcmp (H8, 'On')) && (strcmp (H9, 'On'))
561
562 Array = phased.ConformalArray();
563 Array.ElementPosition = [0 0 0 0 0;b c b c b c;a b c c] .* SPX;
564 Array.ElementNormal = [0 0 0 0 0;0 0 0 0 0 0];
565 Array.Taper = 1;
566
567 Elem = phased.IsotropicHydrophone ;
568 Elem.VoltageSensitivity = -120;
569 Elem.BackBaffled = true;
570 Elem.FrequencyRange = [0 Frequency];
571 Array.Element = Elem;
572
573 figure('Name','Actual Array Geometry','NumberTitle','off');
574 viewArray(Array,'ShowNormal',true,...

```

```

575 'ShowTaper',false,'ShowIndex','None');
576
577 Freq3D = Frequency;
578 w = zeros(getNumElements(Array), length(Frequency));
579 SteerVector = phased.SteeringVector('SensorArray', Array,...
580 'PropagationSpeed', PropagationSpeed, 'NumPhaseShifterBits', PhaseShiftBits(1));
581 for idx = 1:length(Frequency)
582     w(:, idx) = step(SteerVector, Frequency(idx), SteeringAngles(:, idx));
583 end
584
585 format = 'polar';
586 figure('Name','Actual Coverage','NumberTitle','off');
587 pattern(Array, Freq3D, 'PropagationSpeed', PropagationSpeed,...
588 'Type','directivity', 'CoordinatesSystem', format,'weights', w(:,1));
589
590 % H2, H5, H8 OFF
591
592 elseif (strcmp (H1, 'On')) && (strcmp (H2, 'Off')) && (strcmp (H3, 'On')) ...
593 && (strcmp (H4, 'On')) && (strcmp (H5, 'Off')) && (strcmp (H6, 'On')) ...
594 && (strcmp (H7, 'On')) && (strcmp (H8, 'Off')) && (strcmp (H9, 'On'))
595
596 Array = phased.ConformalArray();
597 Array.ElementPosition = [0 0 0 0 0;a c a c a b c c] .* SPX;
598 Array.ElementNormal = [0 0 0 0 0;0 0 0 0 0 0];
599 Array.Taper = 1;
600
601 Elem = phased.IsotropicHydrophone ;
602 Elem.VoltageSensitivity = -120;
603 Elem.BackBaffled = true;
604 Elem.FrequencyRange = [0 Frequency];
605 Array.Element = Elem;
606
607 figure('Name','Actual Array Geometry','NumberTitle','off');
608 viewArray(Array,'ShowNormal',true,...
609 'ShowTaper',false,'ShowIndex','None');
610
611 Freq3D = Frequency;
612 w = zeros(getNumElements(Array), length(Frequency));
613 SteerVector = phased.SteeringVector('SensorArray', Array,...
614 'PropagationSpeed', PropagationSpeed, 'NumPhaseShifterBits', PhaseShiftBits(1));
615 for idx = 1:length(Frequency)

```

```

616         w(:, idx) = step(SteerVector, Frequency(idx), SteeringAngles(:, idx));
617     end
618
619     format = 'polar';
620     figure('Name','Actual Coverage','NumberTitle','off');
621     pattern(Array, Freq3D , 'PropagationSpeed', PropagationSpeed,...
622         'Type','directivity', 'CoordinatesSystem', format,'weights', w(:,1));
623
624     % H3, H6, H9 OFF
625
626     elseif (strcmp (H1, 'On')) && (strcmp (H2, 'On')) && (strcmp (H3, 'Off')) ...
627         && (strcmp (H4, 'On')) && (strcmp (H5, 'On')) && (strcmp (H6, 'Off')) ...
628         && (strcmp (H7, 'On')) && (strcmp (H8, 'On')) && (strcmp (H9, 'Off'))
629
630     Array = phased.ConformalArray();
631     Array.ElementPosition = [0 0 0 0 0;a b a b a b c c] .* SPX;
632     Array.ElementNormal = [0 0 0 0 0;0 0 0 0 0 0];
633     Array.Taper = 1;
634
635     Elem = phased.IsotropicHydrophone ;
636     Elem.VoltageSensitivity = -120;
637     Elem.BackBaffled = true;
638     Elem.FrequencyRange = [0 Frequency];
639     Array.Element = Elem;
640
641     figure('Name','Actual Array Geometry','NumberTitle','off');
642     viewArray(Array,'ShowNormal',true,...
643         'ShowTaper',false,'ShowIndex','None');
644
645     Freq3D = Frequency;
646     w = zeros(getNumElements(Array), length(Frequency));
647     SteerVector = phased.SteeringVector('SensorArray', Array,...
648         'PropagationSpeed', PropagationSpeed, 'NumPhaseShifterBits', PhaseShiftBits(1));
649     for idx = 1:length(Frequency)
650         w(:, idx) = step(SteerVector, Frequency(idx), SteeringAngles(:, idx));
651     end
652
653     format = 'polar';
654     figure('Name','Actual Coverage','NumberTitle','off');
655     pattern(Array, Freq3D , 'PropagationSpeed', PropagationSpeed,...
656         'Type','directivity', 'CoordinatesSystem', format,'weights', w(:,1));

```



```

657         end
658     end
659 end
660
661 % Value changed function: AzimuthdegLimits90EditField
662 function AzimuthdegLimits90EditFieldValueChanged(app, event)
663     Az = app.AzimuthdegLimits90EditField.Value;
664 end
665
666 % Value changed function: ElevationdegLimits90EditField
667 function ElevationdegLimits90EditFieldValueChanged(app, event)
668     El = app.ElevationdegLimits90EditField.Value;
669 end
670
671 % Value changed function: FrequencyHzEditField
672 function FrequencyHzEditFieldValueChanged(app, event)
673     Frequency = app.FrequencyHzEditField.Value;
674 end
675
676 % Value changed function: SeparationParameterListBox
677 function SeparationParameterListBoxValueChanged(app, event)
678     SP = app.SeparationParameterListBox.Value;
679 end
680
681 % Value changed function: SeparationDistanceEditField
682 function SeparationDistanceEditFieldValueChanged(app, event)
683     SD = app.SeparationDistanceEditField.Value;
684 end
685
686 end
687
688 % Component initialization
689 methods (Access = private)
690
691     % Create UIFigure and components
692     function createComponents(app)
693
694         % Create UIFigure and hide until all components are created
695         app UIFigure = uifigure('Visible', 'off');
696         app UIFigure.Position = [100 100 421 367];
697         app UIFigure.Name = 'MATLAB App';

```

```
698 % Create H1SwitchLabel
699 app.H1SwitchLabel = uilabel(app.UIFigure);
700 app.H1SwitchLabel.HorizontalAlignment = 'center';
701 app.H1SwitchLabel.Position = [103 58 25 22];
702 app.H1SwitchLabel.Text = 'H1';
703
704 % Create H1Switch
705 app.H1Switch = uiswitch(app.UIFigure, 'slider');
706 app.H1Switch.Position = [97 81 37 16];
707 app.H1Switch.Value = 'On';
708
709 % Create H2SwitchLabel
710 app.H2SwitchLabel = uilabel(app.UIFigure);
711 app.H2SwitchLabel.HorizontalAlignment = 'center';
712 app.H2SwitchLabel.Position = [203 58 25 22];
713 app.H2SwitchLabel.Text = 'H2';
714
715 % Create H2Switch
716 app.H2Switch = uiswitch(app.UIFigure, 'slider');
717 app.H2Switch.Position = [197 81 37 16];
718 app.H2Switch.Value = 'On';
719
720 % Create H3SwitchLabel
721 app.H3SwitchLabel = uilabel(app.UIFigure);
722 app.H3SwitchLabel.HorizontalAlignment = 'center';
723 app.H3SwitchLabel.Position = [305 58 25 22];
724 app.H3SwitchLabel.Text = 'H3';
725
726 % Create H3Switch
727 app.H3Switch = uiswitch(app.UIFigure, 'slider');
728 app.H3Switch.Position = [299 81 37 16];
729 app.H3Switch.Value = 'On';
730
731 % Create H4SwitchLabel
732 app.H4SwitchLabel = uilabel(app.UIFigure);
733 app.H4SwitchLabel.HorizontalAlignment = 'center';
734 app.H4SwitchLabel.Position = [103 106 25 22];
735 app.H4SwitchLabel.Text = 'H4';
736
737 % Create H4Switch
738 app.H4Switch = uiswitch(app.UIFigure, 'slider');
```

```
739 app.H4Switch.Position = [97 129 37 16];
740 app.H4Switch.Value = 'On';
741
742 % Create H5SwitchLabel
743 app.H5SwitchLabel = uilabel(app.UIFigure);
744 app.H5SwitchLabel.HorizontalAlignment = 'center';
745 app.H5SwitchLabel.Position = [203 106 25 22];
746 app.H5SwitchLabel.Text = 'H5';
747
748 % Create H5Switch
749 app.H5Switch = uiswitch(app.UIFigure, 'slider');
750 app.H5Switch.Position = [197 129 37 16];
751 app.H5Switch.Value = 'On';
752
753 % Create H6SwitchLabel
754 app.H6SwitchLabel = uilabel(app.UIFigure);
755 app.H6SwitchLabel.HorizontalAlignment = 'center';
756 app.H6SwitchLabel.Position = [305 106 25 22];
757 app.H6SwitchLabel.Text = 'H6';
758
759 % Create H6Switch
760 app.H6Switch = uiswitch(app.UIFigure, 'slider');
761 app.H6Switch.Position = [299 129 37 16];
762 app.H6Switch.Value = 'On';
763
764 % Create H7SwitchLabel
765 app.H7SwitchLabel = uilabel(app.UIFigure);
766 app.H7SwitchLabel.HorizontalAlignment = 'center';
767 app.H7SwitchLabel.Position = [103 155 25 22];
768 app.H7SwitchLabel.Text = 'H7';
769
770 % Create H7Switch
771 app.H7Switch = uiswitch(app.UIFigure, 'slider');
772 app.H7Switch.Position = [97 178 37 16];
773 app.H7Switch.Value = 'On';
774
775 % Create H8SwitchLabel
776 app.H8SwitchLabel = uilabel(app.UIFigure);
777 app.H8SwitchLabel.HorizontalAlignment = 'center';
778 app.H8SwitchLabel.Position = [203 155 25 22];
779 app.H8SwitchLabel.Text = 'H8';
```

```

780 % Create H8Switch
781 app.H8Switch = uiswitch(app.UIFigure, 'slider');
782 app.H8Switch.Position = [197 178 37 16];
783 app.H8Switch.Value = 'On';
784
785 % Create H9SwitchLabel
786 app.H9SwitchLabel = uilabel(app.UIFigure);
787 app.H9SwitchLabel.HorizontalAlignment = 'center';
788 app.H9SwitchLabel.Position = [305 155 25 22];
789 app.H9SwitchLabel.Text = 'H9';
790
791 % Create H9Switch
792 app.H9Switch = uiswitch(app.UIFigure, 'slider');
793 app.H9Switch.Position = [299 178 37 16];
794 app.H9Switch.Value = 'On';
795
796 % Create IdealPlotButton
797 app.IdealPlotButton = uibutton(app.UIFigure, 'push');
798 app.IdealPlotButton.ButtonPushedFcn = createCallbackFcn(app, @IdealPlotButtonPushed, true);
799 app.IdealPlotButton.FontWeight = 'bold';
800 app.IdealPlotButton.Position = [97 28 100 22];
801 app.IdealPlotButton.Text = 'Ideal Plot';
802
803 % Create ActualPlotButton
804 app.ActualPlotButton = uibutton(app.UIFigure, 'push');
805 app.ActualPlotButton.ButtonPushedFcn = createCallbackFcn(app, @ActualPlotButtonPushed, true);
806 app.ActualPlotButton.FontWeight = 'bold';
807 app.ActualPlotButton.Position = [236 28 100 22];
808 app.ActualPlotButton.Text = 'Actual Plot';
809
810 % Create FrequencyHzLabel
811 app.FrequencyHzLabel = uilabel(app.UIFigure);
812 app.FrequencyHzLabel.HandleVisibility = 'callback';
813 app.FrequencyHzLabel.FontWeight = 'bold';
814 app.FrequencyHzLabel.Position = [19 319 92 22];
815 app.FrequencyHzLabel.Text = 'Frequency (Hz)';
816
817 % Create FrequencyHzEditField
818 app.FrequencyHzEditField = uieditfield(app.UIFigure, 'numeric');
819 app.FrequencyHzEditField.LowerLimitInclusive = 'off';
820

```

```
821 app.FrequencyHzEditField.Limits = [0 Inf];
822 app.FrequencyHzEditField.RoundFractionalValues = 'on';
823 app.FrequencyHzEditField.ValueDisplayFormat = '%.0f';
824 app.FrequencyHzEditField.ValueChangedFcn = createCallbackFcn(app, @FrequencyHzEditFieldValueChanged, true);
825 app.FrequencyHzEditField.HandleVisibility = 'callback';
826 app.FrequencyHzEditField.HorizontalAlignment = 'center';
827 app.FrequencyHzEditField.Position = [119 319 85 22];
828 app.FrequencyHzEditField.Value = 1000;
829
830 % Create AzimuthdegLimits90Label
831 app.AzimuthdegLimits90Label = uilabel(app.UIFigure);
832 app.AzimuthdegLimits90Label.FontWeight = 'bold';
833 app.AzimuthdegLimits90Label.Position = [18 269 87 28];
834 app.AzimuthdegLimits90Label.Text = {'Azimuth (deg)'; '[Limits: +/- 90]'};
835
836 % Create AzimuthdegLimits90EditField
837 app.AzimuthdegLimits90EditField = uieditfield(app.UIFigure, 'numeric');
838 app.AzimuthdegLimits90EditField.Limits = [-90 90];
839 app.AzimuthdegLimits90EditField.ValueDisplayFormat = '%.0f';
840 app.AzimuthdegLimits90EditField.ValueChangedFcn = createCallbackFcn(app, @AzimuthdegLimits90EditFieldValueChanged,
841 app.AzimuthdegLimits90EditField.HandleVisibility = 'callback';
842 app.AzimuthdegLimits90EditField.HorizontalAlignment = 'center';
843 app.AzimuthdegLimits90EditField.Position = [119 275 85 22];
844
845 % Create ElevationdegLimits90Label
846 app.ElevationdegLimits90Label = uilabel(app.UIFigure);
847 app.ElevationdegLimits90Label.FontWeight = 'bold';
848 app.ElevationdegLimits90Label.Position = [19 225 92 28];
849 app.ElevationdegLimits90Label.Text = {'Elevation (deg)'; '[Limits: +/- 90]'};
850
851 % Create ElevationdegLimits90EditField
852 app.ElevationdegLimits90EditField = uieditfield(app.UIFigure, 'numeric');
853 app.ElevationdegLimits90EditField.Limits = [-90 90];
854 app.ElevationdegLimits90EditField.ValueDisplayFormat = '%.0f';
855 app.ElevationdegLimits90EditField.ValueChangedFcn = createCallbackFcn(app, @ElevationdegLimits90EditFieldValueChan
856 app.ElevationdegLimits90EditField.HandleVisibility = 'callback';
857 app.ElevationdegLimits90EditField.HorizontalAlignment = 'center';
858 app.ElevationdegLimits90EditField.Position = [119 231 85 22];
859
860 % Create SeparationDistanceEditFieldLabel
861 app.SeparationDistanceEditFieldLabel = uilabel(app.UIFigure);
```

```

862 app.SeparationDistanceEditFieldLabel.FontWeight = 'bold';
863 app.SeparationDistanceEditFieldLabel.Position = [231 236 68 28];
864 app.SeparationDistanceEditFieldLabel.Text = {'Separation'; 'Distance'};
865
866 % Create SeparationDistanceEditField
867 app.SeparationDistanceEditField = uieditfield(app.UIFigure, 'numeric');
868 app.SeparationDistanceEditField.LowerLimitInclusive = 'off';
869 app.SeparationDistanceEditField.Limits = [0 Inf];
870 app.SeparationDistanceEditField.ValueDisplayFormat = '%.1f';
871 app.SeparationDistanceEditField.ValueChangedFcn = createCallbackFcn(app, @SeparationDistanceEditFieldValueChanged,
872 app.SeparationDistanceEditField.HorizontalAlignment = 'center';
873 app.SeparationDistanceEditField.Position = [315 239 90 22];
874 app.SeparationDistanceEditField.Value = 0.5;
875
876 % Create SeparationParameterListBoxLabel
877 app.SeparationParameterListBoxLabel = uilabel(app.UIFigure);
878 app.SeparationParameterListBoxLabel.FontWeight = 'bold';
879 app.SeparationParameterListBoxLabel.Position = [232 300 68 28];
880 app.SeparationParameterListBoxLabel.Text = {'Separation'; 'Parameter'};
881
882 % Create SeparationParameterListBox
883 app.SeparationParameterListBox = uilistbox(app.UIFigure);
884 app.SeparationParameterListBox.Items = {'Wavelength', 'Metres'};
885 app.SeparationParameterListBox.ItemsData = [1 2];
886 app.SeparationParameterListBox.ValueChangedFcn = createCallbackFcn(app, @SeparationParameterListBoxValueChanged, t)
887 app.SeparationParameterListBox.Position = [315 283 90 47];
888 app.SeparationParameterListBox.Value = 1;
889
890 % Show the figure after all components are created
891 app.UIFigure.Visible = 'on';
892
893 end
894
895 % App creation and deletion
896 methods (Access = public)
897
898 % Construct app
899 function app = GUI_V1_2_exported
900
901 % Create UIFigure and components
902 createComponents(app)

```

```
903 % Register the app with App Designer
904 registerApp(app, app UIFigure)
905
906
907 if nargin == 0
908     clear app
909 end
910
911
912 % Code that executes before app deletion
913 function delete(app)
914
915     % Delete UIFigure when app is deleted
916     delete(app.UIFigure)
917 end
918
919 end
```