Not Applicable If : The incident points alternate on the curve and plane mirror, there is no consecutive incidents of light on the either curve or x axis, Number of iterations limited in case of back tracing

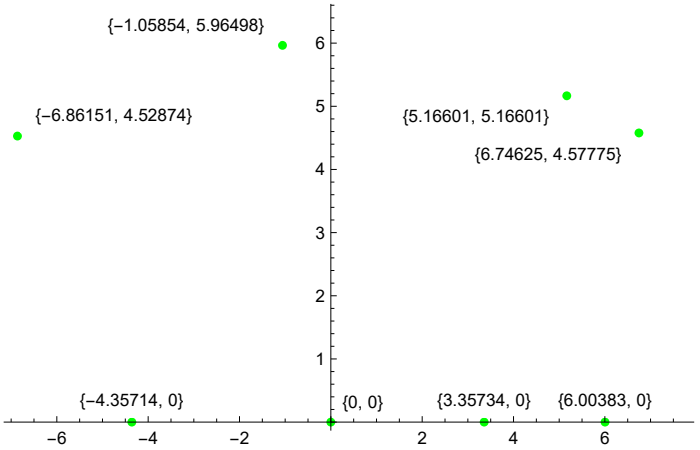
Applicable Cases: No information of source point of light, there can be multiple incidents on the same point, light can retrace its path back

Incident points input and their plot

In[•]:=
```
intersecpts = {{5.166010488516726`,5.166010488516724`},{6.003831069285709`,0},{6.746248565974
intersecplot = ListPlot[Labeled[#,#] &/@ intersecpts, PlotRange→Full,PlotStyle→{Green}]
```

Out[•]=
```
{{5.16601, 5.16601}, {6.00383, 0}, {6.74625, 4.57775}, {3.35734, 0},
 {-1.05854, 5.96498}, {-4.35714, 0}, {-6.86151, 4.52874}, {0, 0}}
```

Out[•]=

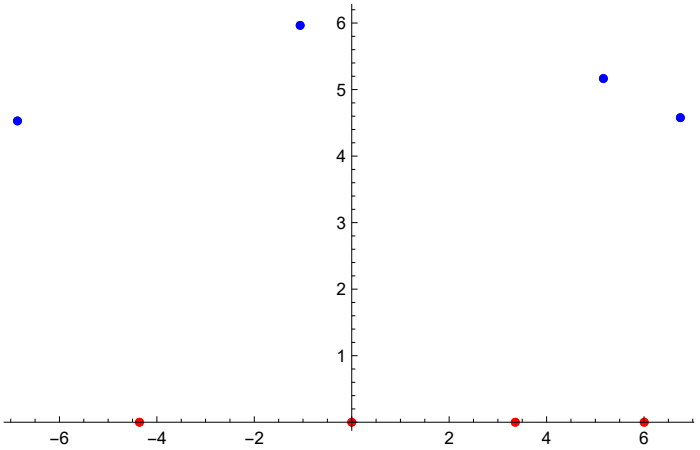


Separate the points into x -axis (plane mirror points) and unknown reflecting curve incident points

In[•]:=
```
For[xaxispts = {}; i=0, i<Length[intersecpts], i++; If[intersecpts[[i, 2]]==0, AppendTo[xaxispts
curvepts = Sort[DeleteCases[intersecpts, Alternatives @@ xaxispts]];
xaxispts = Sort[xaxispts];
Print["curvepts = ", curvepts]
Print["xaxispts = ", xaxispts]
intersecplot2 = ListPlot[{#} & /@ intersecpts, PlotRange→Full,PlotStyle→{Blue, Red, Blue, Red
```

```
curvepts = {{-6.86151, 4.52874}, {-1.05854, 5.96498}, {5.16601, 5.16601}, {6.74625, 4.57775}}
xaxispts = {{-4.35714, 0}, {0, 0}, {3.35734, 0}, {6.00383, 0}}
```

Out[•]=



Function for initial 3 point tuple permutations

In[•]:=

```
pttuplefunc[domainls_, codomainls_]:=
    Module[{tupleset={}},
            For[k=0, k<Length[domainls], k++;
                tupleset = AppendTo[tupleset, {domainls[[2]],codomainls[[3]], If[domainls[[k]]≠{0,0
                ];
    Return[tupleset, Module]];
```

Path permutations till the first plane mirror reflection

In[•]:=

```
tuple3point = pttuplefunc[xaxispts, curvepts]
Print["No of tuples = ", Length[tuple3point]]
```

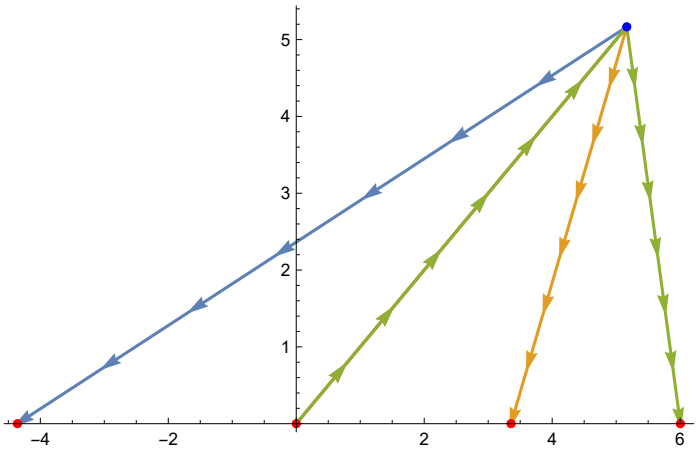
Out[•]=

```
{{{0, 0}, {5.16601, 5.16601}, {-4.35714, 0}},
 {{0, 0}, {5.16601, 5.16601}, {3.35734, 0}}, {{0, 0}, {5.16601, 5.16601}, {6.00383, 0}}}

No of tuples = 3
```

```
For[tupleset1 ={};k =0, k<Length[xaxispts], k++;
    For[i=0, i<Length[curvepts], i++;
        For[slope1=0, slope2 = 0;j=0, j<Length[curvepts], j++;
            If[curvepts[[i]]=curvepts[[j]], Continue[]];
            slope1 = (curvepts[[i]][[2]]-xaxispts[[k]][[2]])/(curvepts[[i]][[1]]-xaxispts[[k]][[1]]);
            slope2 = (curvepts[[j]][[2]]-xaxispts[[k]][[2]])/(curvepts[[j]][[1]]-xaxispts[[k]][[1]]);
            Print[slope1];
            If[slope1 == -slope2, tupleset1 = AppendTo[tupleset1, {curvepts[[i]], xaxispts[[k]], c
            ]]]];

For[tuple1 = {}, k=0, k<Length[xaxispts], k++
```

In[ ]:=
```
point3path = ListLinePlot[tuple3point, Axes→True, AxesOrigin→{0,0}, MeshFunctions → {#2 &}, M
  MeshShading → {Arrowheads[Small]}, DataRange → {0, 4 Pi}] /. Line → Arrow;
Show[point3path, intersecplot2]
```

Out[ ]=



Next we check check reflections and eliminate paths based on it

```
(*Function to check for the curve points lying on the reflected ray *)
reflecdrop[domainls_] :=
    Module[{tupleset = {}},
        For[i=0, i<Length[domainls], i++;
            For[(eqn = y == -(Last[List @@ Reduce[y - InterpolatingPolynomial[{Part[domainls[[
                j=0, j<Length[curvepts], j++;
                If[(eqn /. {x→curvepts[[j,1]], y→curvepts[[j,2]]}) == True,
                    tupleset = AppendTo[tupleset,Join[domainls[[i]], {curvepts[[j]]}]]]]];
    Return[tupleset, Module]];
```

Out of 64 paths, after elimination we have 24 paths after the 1st reflection on plane mirror

In[ ]:=
```
a = reflecdrop[tuple3point];
a
```

Out[ ]=
```
{{{0, 0}, {5.16601, 5.16601}, {6.00383, 0}, {6.74625, 4.57775}}}
```

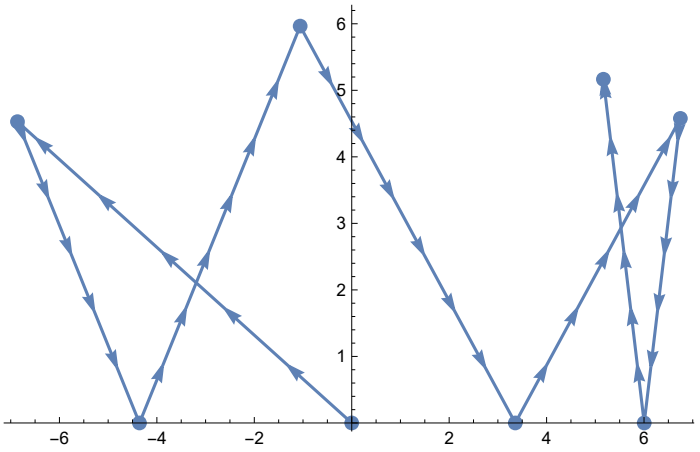
In[ ]:=
```
path2func[path_]:=
    Module[{tupleset={}},
        For[tupleset={};i=0, i<Length[path], i++;
            For[j=0, j<Length[xaxispts], j++;
                tupleset = Append[tupleset, Join[path[[i]], {xaxispts[[j]]}]]]; Print[tupleset]]
        Return[tupleset, Module]];
```

In[ ]:=
```
merge[list_]:=Module[{a= reflecdrop[list]}, Return[path2func[a], Module]];
function[list_]:= Module[{a, new={}},
    If[EvenQ[Length[intersecpts]]==True, a = reflecdrop[Nest[merge, list, (Length[intersecpts]
    For[i=0, i<Length[a], i++;
    If[CountDistinct[a[[i]]]==Length[intersecpts], new = AppendTo[new, a[[i]]]]];
    Return[new, Module]];
```

In[ ]:=
```
finalpath = function[tuple3point]
```

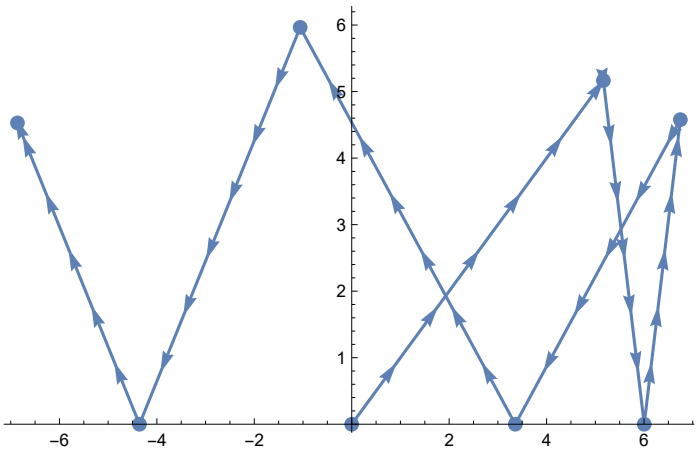
In[ ]:=
```
ListLinePlot[finalpath[[1]], PlotMarkers→{Automatic, 8}, Axes→True, AxesOrigin→{0,0}, MeshFunct
  MeshShading → {Arrowheads[Small]}, DataRange → {0, 4 Pi}] /. Line → Arrow
```

Out[ ]=

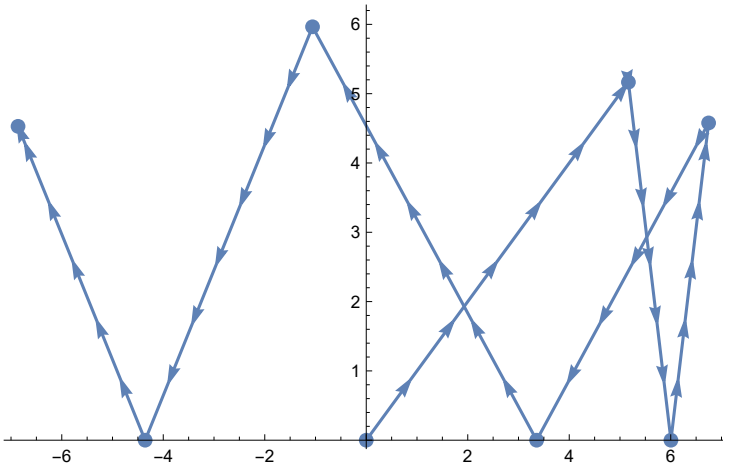


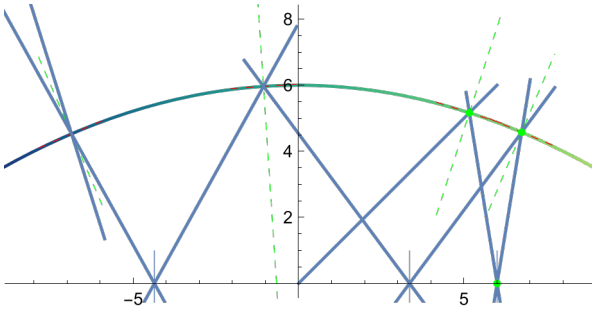
In[ ]:=
```
ListLinePlot[finalpath[[2]], PlotMarkers→{Automatic, 8}, Axes→True, AxesOrigin→{0,0}, MeshFunct
  MeshShading → {Arrowheads[Small]}, DataRange → {0, 4 Pi}] /. Line → Arrow
```

Out[ ]=



Comparison





Trying large number of iterations for back tracing

In[ ]:=
```
(* function2[list_]:= Module[{a, new={}},
    If[EvenQ[Length[intersecpts]]==True, a = reflecdrop[Nest[merge, list, 3*(Length[intersecpt
    For[i=0, i<Length[a], i++;
    If[CountDistinct[a[[i]]==Length[intersecpts], new = AppendTo[new, a[[i]]]]];
    Return[new, Module]]; *)
```

In[ ]:=
```
(* b = function2[tuple3point]*)
```

In[ ]:=
```
(* Length[b]
ListLinePlot[b, PlotMarkers→{Automatic, 8}, Axes→True, AxesOrigin→{0,0}, MeshFunctions → {#2
  MeshShading → {Arrowheads[Small]}, DataRange → {0, 4 Pi}] /. Line → Arrow *)
```