Utilizing analytical transfer functions to gauge the effect of velocity reversals

Frequency range to take into consideration:

In[1]:=
```
fmin = 0.01; fmax = 100;
```

In[2]:=
```
(*Frequency interval*)
flist = Subdivide[fmin, fmax, 2000];
```

The parameters of the simulations

In[3]:=
$$\text{anTF} = \text{Cos}[r_1]\,\text{Cos}[r_2]\left(1 - \frac{\sqrt{\mu_1\,\rho_1}\;\text{Tan}[r_1]\,\text{Tan}[r_2]}{\sqrt{\mu_2\,\rho_2}} - \frac{i\left(-\sqrt{\mu_1\,\rho_1}\;\text{Tan}[r_1] - \sqrt{\mu_2\,\rho_2}\;\text{Tan}[r_2]\right)}{\sqrt{\mu_{\text{half}}\,\rho_{\text{half}}}}\right);$$

Implementing the layer matrices

This list, when evaluated (see that as per definition the evaluation is delayed), generates the matrices for a given set of impedances (“alphas”), layer heights (“hs”) and shear-wave velocities (“Vs”).

The matrices relate the ground displacement to the displacement at the base of the last layer, the last one being the last one being the half-space.

In[4]:=
$$\text{Llist} := \text{Table}\Big[\Big\{\Big\{\frac{1}{2}(1+\text{alphas}[[jj]]) * \text{Exp}\Big[i * \frac{\omega * \text{hs}[[jj]]}{\text{Vs}[[jj]]}\Big],\ \frac{1}{2}(1-\text{alphas}[[jj]]) * \text{Exp}\Big[-i * \frac{\omega * \text{hs}[[jj]]}{\text{Vs}[[jj]]}\Big]\Big\},$$
$$\Big\{\frac{1}{2}(1-\text{alphas}[[jj]]) * \text{Exp}\Big[i * \frac{\omega * \text{hs}[[jj]]}{\text{Vs}[[jj]]}\Big],\ \frac{1}{2}(1+\text{alphas}[[jj]]) * \text{Exp}\Big[-i * \frac{\omega * \text{hs}[[jj]]}{\text{Vs}[[jj]]}\Big]\Big\}\Big\},\ \{jj, 1, \text{Length}[\text{alphas}]\}\Big];$$

The parameters of the simulations

In[5]:=
$$\text{Rho}[V\_] := 1800 + \left(\frac{V - 200}{700}\right) * 300;$$
```
dists = {{1, 4}, {2, 3}, {3, 2}, {4, 1}};
Vup = Range[400, 800, 100];
Vdown = Range[300, 800, 50];
```

```
In[9]:= (*Bedrock Properties*)
Vhalf = 900;
rhohalf = Rho[Vhalf];
muhalf = rhohalf * Vhalf^2;
```

Computations

```
In[12]:= NN = 2;
DE = 0.1;
dataTripletsNew = {};
AuxTableFiguresNew = {};
Do[(*kk: For each type thicknesses' combo*)
 Do[(*jj: For each Vs_L1 value*)
  Do[(*ii: For each Vs_L2 value*)
   (*Compute TF---------------------------------------*)
   (*Layer depth, average depth and thickness*)
   Depths = {dists[[kk]][[1]], dists[[kk]][[2]]} * 10.;
   hs = dists[[kk]] * 10;
   SiteName = "L_1=" <> ToString[10 * dists[[kk]][[1]]] <>
     "m, L_2=" <> ToString[10 * dists[[kk]][[2]]] <> "m, Vs_L1=" <>
     ToString[Vup[[jj]]] <> "m/s, Vs_L2=" <> ToString[Vdown[[ii]]] <> "m/s";
   Htotal = Total[hs]; (*always equal to 100 in this case*)
   (*The list is ordered from free surface to bottom*)
   (*Layer shear-wave velocities, densities, impedances and fundamental
    period estimate --------------------------------------------------*)
   Vs = {Vup[[jj]], Vdown[[ii]]};
   rhos = Rho[#] & /@ Vs;
   mus = Table[rhos[[uu]] * Vs[[uu]]^2, {uu, 1, Length[Vs]}];
   (*Frequency interval*)
   flist = Subdivide[fmin, fmax, 2000];
   alphas = Flatten@{(rhos[[1]] * Vs[[1]])/(rhos[[2]] * Vs[[2]]), (rhos[[2]] * Vs[[2]])/(rhohalf * Vhalf)};
   (*Kramer's*)
   L[ω_] = Llist[[2]].Llist[[1]];
   (*For[uu=2,uu≤Length[alphas],uu++,L[ω_]=Flatten[Llist[[uu]],1].L[ω]];*)
   TF[ω_] = 2/(L[ω][[1, 1]] + L[ω][[1, 2]]);
   Vbase = Vs[[-1]];
   TFKramer =
    Table[{flist[[uu]], Abs[TF[(2 * π)/(Sqrt[1 + i * DE]) flist[[uu]]]]}, {uu, 1, Length@flist}];
   (*plotList=AppendTo[plotList,TFKramer];*)
   (*Compute Transfer Function (Aki and Richard's) --
```

```
 -----------------------------------*)
layerMatrix = hs[[#]] * {{0., 1/(mus[[#]] * (1 + i * DE))}, {-ω^2 * rhos[[#]], 0.}} & /@
  Range[Length@hs];
(*Layer matrices*)
exactExpFreq = {};
Do[
 (*Create the exponential matrices
  for each layer and put them in order to multiply them*)
 expMatList = func[layerMatrix[[#]] /. ω → 2 * π * flist[[ii]]] & /@
    Range[Length@hs] /. func → MatrixExp;
 (*Proceed with the multiplication*)
 exactExp = expMatList[[2]].expMatList[[1]];
 (*Do[exactExp=exactExp.expMatList[[jj]],
   {jj,2,Length@hs}];*)
 (*Add to the list of values*)
 exactExpFreq = AppendTo[exactExpFreq,
   {flist[[ii]], 2/Abs[exactExp[[1, 1]] - i * exactExp[[2,1]]/(2 π*flist[[ii]]*Sqrt[muhalf*rhohalf])]}]
 , {ii, 1, Length@flist}];
(*Compute and Evaluate Analytical ---------*)
(*Prepare to evaluate numerically*)
auxVar = ((anTF /. r_1 → 2 π * f * hs[[1]]/(Vs[[1]] Sqrt[1 + i * DE])) /. μ_1 ρ_1 → (rhos[[1]] * Vs[[1]])^2) /.
  μ_half ρ_half → (rhohalf * Vhalf)^2;
Do[
 auxVar2 = auxVar;
 auxVar = (auxVar2 /. r_i → 2 π * f * hs[[i]]/(Vs[[i]] Sqrt[1 + i * DE])) /. μ_i ρ_i → (rhos[[i]] * Vs[[i]])^2;
 , {i, 2, NN}];
auxVar = Expand[auxVar];
(*Make table evaluating numerically*)
myTF = Table[{flist[[k]], 2/Abs[auxVar /. f → flist[[k]]]}, {k, 1, Length@flist}];
(*Save data*)
peak = Max[myTF[[All, 2]]];
dataTripletsNew = AppendTo[dataTripletsNew, {hs[[1]]/hs[[2]], rhos[[2]] * Vs[[2]]/(rhohalf * Vhalf), peak}];
(*Plot Transfer
  function -------------------------------------------------------------*)
TFPlot = ListLogLinearPlot[{
```

```
       TFKramer,
       exactExpFreq,
       myTF},
      PlotRange → {{fmin, fmax}, {0, 6}},
      Axes → False,
      PlotStyle → {{Automatic, Thickness[0.012]},
        {Red, Dotted, Thickness[0.015/3]}, {Orange, Dashed, Thickness[0.015/3]}},
      Frame → {{True, True}, {True, True}},
      Joined → True,
      FrameLabel → {{"u_top/S_i", None}, {"freq.[Hz]", Style[SiteName, FontSize → 8]}},
      PlotLegends → Placed[
        LineLegend[{"Kramer", "A&R", "Analytical"}, LegendLayout → "Column"], {0.2, Top}],
      RotateLabel -> False,
      ImageSize → Small,
      AspectRatio → 1];
    AuxTableFiguresNew = AppendTo[AuxTableFiguresNew, TFPlot];
    (*AuxTableFigures[[kk,jj]][[ii]]=TFPlot;*)
    , {ii, 1, 1 + 2 * jj}]
   , {jj, 1, Length@Vup}]
  , {kk, 1, Length@dists}]
```

AuxTableFiguresNrew contains all the images of transfer functions.

Verifying the lower bound:

```
In[17]:= Vslone = Table[ConstantArray[Vup[[ii]], 1 + 2 ii], {ii, 1, 5}];
impedanceList = N[Rho[#] * # / Sqrt[rhohalf * muhalf]] & /@ Flatten[Vslone];
listEI = Flatten[Join[impedanceList, impedanceList, impedanceList, impedanceList]];

In[20]:= scalingPoints = Table[{listEI[[ii]], dataTripletsNew[[ii]][[3]]}, {ii, 1, 140}];

In[21]:= listColors = {Blue, Red, Green, Yellow, Gray};
colors = Table[ConstantArray[listColors[[ii]], 1 + 2 ii], {ii, 1, 5}];
colorList = Flatten[Join[colors, colors, colors, colors]];
listPoints2 = Table[{colorList[[ii]], Point[scalingPoints[[ii]]]}, {ii, 1, 140}];
```

Verifying the upper bound:

In[25]:=
```
impedanceList2 = {};
Do[
 Do[
  AppendTo[impedanceList2, N[(Rho[Vdown[[jj]]] * Vdown[[jj]])/(rhohalf * Vhalf)]]
  , {jj, 1, 1 + 2 * ii}]
 , {ii, 1, Length@Vup}]
```

In[27]:=
```
listZs = Flatten[Join[impedanceList2, impedanceList2, impedanceList2, impedanceList2]];
```

In[28]:=
```
scalingPoints2 = Table[{listZs[[ii]], dataTripletsNew[[ii]][[3]]}, {ii, 1, 140}];
```

In[29]:=
```
listPoints3 = Table[{colorList[[ii]], Point[scalingPoints2[[ii]]]}, {ii, 1, 140}];
```

In[30]:=
```
listPoints3 = Table[{colorList[[ii]], Point[scalingPoints2[[ii]]]}, {ii, 1, 140}];
```

Images

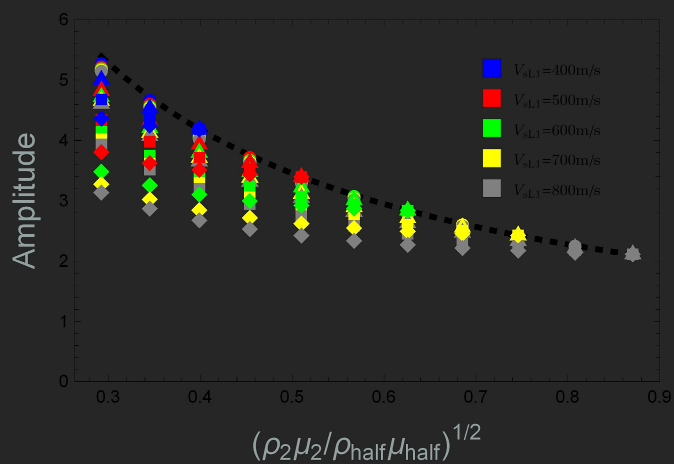
In[31]:=
```
(*Load the package code*)package = Import[
   "http://raw.github.com/AlexeyPopkov/PolygonPlotMarkers/master/PolygonPlotMarkers.m",
   "Text"];

(*Install the package (existing file will be overwritten!)*)
Export[FileNameJoin[{$UserBaseDirectory, "Applications", "PolygonPlotMarkers.m"}],
  package, "Text"];
```

In[33]:=
```
Needs["PolygonPlotMarkers`"]
```

In[34]:=
```
openForms =
  Graphics[{EdgeForm[], PolygonMarker[#, Offset[6]]}, AlignmentPoint → {0, 0}] & /@
   {"Circle", "Triangle", "Square", "Diamond"};
markers = Table[ConstantArray[openForms[[ii]], 35], {ii, 1, 4}];
markerList = Flatten[markers];
```

In[37]:=
```
figUB = Show[
  Plot[2/(x + π/4 DE), {x, Min@scalingPoints2[[All, 1]], Max@scalingPoints2[[All, 1]]},
   PlotRange → {{0.9 * Min@scalingPoints2[[All, 1]], .9}, {0, 6}},
   PlotStyle → {Black, Dashed, Thickness[0.01]},
   Axes → False,
   Frame → True,
   FrameLabel → {{Style["Amplitude", FontSize → 16], ""},
     {Style["(ρ2μ2/ρhalfμhalf)^(1/2)", FontSize → 16], ""}},
   Epilog → Inset[SwatchLegend[{Blue, Red, Green, Yellow, Gray},
      Style["VsL1=" <> #, FontFamily → "LM Roman 8", FontSize → 8] & /@
       {"400m/s", "500m/s", "600m/s", "700m/s", "800m/s"}], Scaled[{0.8, 0.7}]]],
  ListPlot[{#} & /@ scalingPoints2, AspectRatio → 1, PlotStyle → colorList,
   PlotMarkers → markerList]]
```

Out[37]=



In[39]:=
```
figLB = Show[
  Plot[2/(x + π/4 DE), {x, Min@scalingPoints[[All, 1]], Max@scalingPoints[[All, 1]]},
   PlotRange → {{0.9 * Min@scalingPoints[[All, 1]], .9}, {0, 6}},
   PlotStyle → {Black, Dashed, Thickness[0.01]},
   Axes → False,
   Frame → True,
   FrameLabel → {{Style["Amplitude", FontSize → 16], ""},
     {Style["(ρ1μ1/ρhalfμhalf)^1/2", FontSize → 16], ""}},
   Epilog → Inset[SwatchLegend[{Blue, Red, Green, Yellow, Gray},
      Style["VsL1=" <> #, FontFamily → "LM Roman 8", FontSize → 8] & /@ {"400m/s", "500m/s",
        "600m/s", "700m/s", "800m/s"}, LegendLayout → "Row"], Scaled[{0.5, 0.15}]]],
  ListPlot[{#} & /@ scalingPoints, AspectRatio → 1, PlotStyle → colorList,
   PlotMarkers → markerList]]
```

Out[39]=