Finite Difference Solution of the 1D Helmholtz Differential Equation

Initialization Code (optional)

```
checkTerm[t_?(NumberQ[#] &)] := If[Abs[t - 1] < $MachineEpsilon, 1, If[Abs[t] < $MachineEpsilon, 0, t]];
(*-----------------------------------------------------------*)
generatePhysicalCoordinates1D[h_?(Element[#, Reals] && Positive[#] &),
    len_?(Element[#, Reals] && Positive[#] &), centerGrid_?(Element[#, Booleans] &)] :=
  Module[{i, nodes, intervals}, intervals = Floor[len/h];
    nodes = intervals + 1;
    Which[centerGrid == True, If[OddQ[nodes], Table[h*i, {i, -(intervals/2), intervals/2, 1}],
      Table[h*i, {i, -(nodes/2) + 1, nodes/2, 1}]], centerGrid == False, Table[h*i, {i, 0, intervals, 1}]]];
(*-----------------------------------------------------------*)
forceTermUsedFormat1D[forceTermSelection_, aa_, bb_, sstdx_, xx0_, nn0_, x_] := Module[
    {a = checkTerm[aa], b = checkTerm[bb], stdx = checkTerm[sstdx], x0 = checkTerm[xx0], n0 = checkTerm[nn0]},
    Which[
     forceTermSelection == 1, a ,
     forceTermSelection == 2, a*x^(n0),
     forceTermSelection == 3, a* 1/(stdx HoldForm[Sqrt[2 Pi]]) Exp[-(x - x0)^2/(2 stdx^2)],
     forceTermSelection == 4, a*Cos[b HoldForm[Pi x]]
    ]
  ];
(*-----------------------------------------------------------*)
makeScrolledPane[mat_?(MatrixQ[#, NumberQ] &),
    nRow_?(IntegerQ[#] && Positive[#] &), nCol_?(IntegerQ[#] && Positive[#] &)] :=
  Module[{t}, t = Grid[mat, Spacings → {.4, .4}, Alignment → Left, Frame → All];
    t = Text@Style[NumberForm[Chop[N@t], {6, 5}, NumberSigns → {"-", ""},
        NumberPadding → {"", ""}, SignPadding → True], LineBreakWithin → False];
    Pane[t, ImageSize → {nCol, nRow}, Scrollbars → True]];

(*-----------------------------------------------------------*)
makeScrolledPane[lst_?(VectorQ[#, NumericQ] &),
    nRow_?(IntegerQ[#] && Positive[#] &), nCol_?(IntegerQ[#] && Positive[#] &)] :=
  Module[{t}, t = Grid[{lst}, Spacings → {.4, .4}, Alignment → Left, Frame → All];
    t = Text@Style[AccountingForm[Chop[N@t], {6, 5}, NumberSigns → {"-", ""},
        NumberPadding → {"", ""}, SignPadding → True], LineBreakWithin → False];
    Pane[t, ImageSize → {nCol, nRow}, Scrollbars → True]];
(*-----------------------------------------------------------*)
process[h_, centerGrid_, kValue_, n0_, a_, b_, x0_, stdx_, forceTermSelection_, plotToShow_,
    westBCtype_, westbc_, westBCconstantValue_, eastBCtype_, eastbc_, eastBCconstantValue_,
    gstatusMessage_, showGridLines_] := Module[{Lx = 1, forceVector, u, forceGrid, grid, AA},

    {grid, forceGrid, u, AA, forceVector} = initializeSystem[h, Lx, centerGrid, forceTermSelection, a, b, n0,
      x0, stdx, westBCtype, westbc, westBCconstantValue, eastBCtype, eastbc, eastBCconstantValue, kValue];
    gstatusMessage = "ready..";

    makeFinalPlot[solve[u, AA, forceVector, westBCtype, eastBCtype], AA, grid, plotToShow, showGridLines]
  ];
 (*-----------------------------------------------------------*)
```

```
solve[$u_, AA_, forceVector_, westBCtype_, eastBCtype_] := Module[{u = $u},

   Which[westBCtype == "Sommerfeld" && eastBCtype == "Dirichlet",
    (
     u[[1 ;; -2]] = Re[LinearSolve[AA, forceVector]]
    ),
    westBCtype == "Dirichlet" && eastBCtype == "Sommerfeld",
    (
     u[[2 ;; -1]] = Re[LinearSolve[AA, forceVector]]
    ),
    westBCtype == "Dirichlet" && eastBCtype == "Dirichlet",
    (
     u[[2 ;; -2]] = LinearSolve[AA, forceVector]
    )
   ];
   u
  ];
 (*-----------------------------------------------------------*)
initializeSystem[h_, length_, centerGrid_, forceTermSelection_, a_, b_, n0_, x0_, stdx_, westBCtype_,
   westbc_, westBCconstantValue_, eastBCtype_, eastbc_, eastBCconstantValue_, kValue_] :=
  Module[{n, grid, forceGrid, u, AA, forceVector},

   (*grid contains the x physical coordinates of each grid point*)
   grid = N[generatePhysicalCoordinates1D[h, length, centerGrid]];
   n = Length[grid];
   u = Table[0, {n}];
   forceGrid = makeForceGrid[a, b, n0, x0, stdx, n, forceTermSelection, grid];

   u = setBoundaryConditions[u, grid, westBCtype,
     westbc, westBCconstantValue, eastBCtype, eastbc, eastBCconstantValue];
   {AA, forceVector} = makeSystemAndRightHandSideVector[n, westBCtype, eastBCtype, kValue, h, forceGrid, u];
   {grid, forceGrid, u, AA, forceVector}
  ];
 (*-----------------------------------------------------------*)
makeForceGrid[a_, b_, n0$_, x0_, stdx_, n_, forceTermSelection_, grid_] := Module[{n0 = checkTerm[n0$], i},

   Which[
    forceTermSelection == 1, Table[a, {n}],

    forceTermSelection == 2,
    If[n0 == 0,
     Table[a, {i, n}],
     Table[a * (grid[[i]]) ^ n0, {i, n}]
    ],

    forceTermSelection == 3, Table[ a / (stdx * Sqrt[2 * Pi]) * Exp[- (grid[[i]] - x0) ^ 2 / (2 stdx ^ 2 ) ], {i, n}],
    forceTermSelection == 4, Table[a Cos[b Pi grid[[i]]] , {i, n}]
   ]
  ];
(*-----------------------------------------------------------*)
makeSystemAndRightHandSideVector[n_, westBCtype_, eastBCtype_, kValue_, h_, forceGrid_, u_] :=
  Module[{forceVector, AA, kh = kValue * h, omega},

   omega = 2.0 Cos[kh] + (kh) ^ 2;
   Which[westBCtype == "Sommerfeld" && eastBCtype == "Dirichlet",
    (
     AA = SparseArray[{
         Band[{1, 1}] → omega - (kh) ^ 2,
         Band[{2, 1}] → -1.,
         Band[{1, 2}] → -1.
        }, {n - 1, n - 1}
      ];
     AA[[1, 1]] = 1.0;
```

```
      AA[[1, 2]] = -2.0 I Sin[kh];
      AA[[1, 3]] = -1.0;

      forceVector = Table[0, {n - 1}];
      forceVector[[1]] = 0;
      forceVector[[2 ;; -2]] = h^2 * forceGrid[[2 ;; -3]];
      forceVector[[-1]] = h^2 * forceGrid[[n - 1]] + u[[-1]]
     ),
     westBCtype == "Dirichlet" && eastBCtype == "Sommerfeld",
     (
      AA = SparseArray[{
          Band[{1, 1}] → omega - (kh)^2,
          Band[{2, 1}] → -1.,
          Band[{1, 2}] → -1.
         }, {n - 1, n - 1}
        ];
      AA[[-1, -1]] = 1.0;
      AA[[-1, -2]] = -2.0 I Sin[kh];
      AA[[-1, -3]] = -1.0;

      forceVector = Table[0, {n - 1}];
      forceVector[[1]] = h^2 * forceGrid[[2]] + u[[1]];
      forceVector[[2 ;; -2]] = h^2 * forceGrid[[3 ;; -2]];
      forceVector[[-1]] = 0
     ),
     westBCtype == "Dirichlet" && eastBCtype == "Dirichlet",
     (
      AA = SparseArray[{
          Band[{1, 1}] → omega - (kh)^2,
          Band[{2, 1}] → -1.,
          Band[{1, 2}] → -1.
         }, {n - 2, n - 2}
        ];
      forceVector = Table[0, {n - 2}];
      forceVector[[1]] = h^2 * forceGrid[[2]] + u[[1]];
      forceVector[[-1]] = h^2 * forceGrid[[n - 1]] + u[[-1]];
      forceVector[[2 ;; -2]] = h^2 * forceGrid[[3 ;; -3]]
     )
    ];

    {AA, forceVector}
   ];
 (*-------------------------------------------------------*)
setBoundaryConditions[$u_, grid_, westBCtype_, westbc_,
    westBCconstantValue_, eastBCtype_, eastbc_, eastBCconstantValue_] := Module[{u = $u},

    If[westBCtype == "Dirichlet",
     (
      u[[1]] = westBCconstantValue * westbc[grid[[1]]]
     )
    ];

    If[eastBCtype == "Dirichlet",
     (
      u[[-1]] = eastBCconstantValue * eastbc[grid[[-1]]]
     )
    ];
    u
   ];
 (*-------------------------------------------------------*)
getNDsolveResult[k_, westBCtype_, eastBCtype_, westBCconstantValue_, eastBCconstantValue_,
    forceTermSelection_, a_, b_, n0_, x0_, stdx_, centerGrid_, showGridLines_] :=
  Module[{f, x, eq, y, sol, boundaryConditions, from, to, plotOptions},
```

```
    plotOptions = {PlotRange → All,
       AxesOrigin → {0, 0},
       ImagePadding → {{40, 20}, {25, 30}},
       PlotLabel → Text@Style[Row[{Style["NDSolve", "MR"], " solution"}], 12],
       AxesLabel → {Text@Style["x", Italic, 11],
         Text@Style[Row[{Style["u", Italic], "(", Style["x", Italic], ")"}], 11]},
       ImageSize → {ContentSizeW - 20, ContentSizeH - 250},
       AspectRatio → 0.5,
       PlotStyle → Red,
       ImageMargins → 1};

    If[centerGrid,
     (from = -0.5; to = 0.5),
     (from = 0; to = 1.0)
    ];

    f = Which[
       forceTermSelection == 1, a,
       forceTermSelection == 2, a * x^n0,
       forceTermSelection == 3, a / (stdx * Sqrt[2 * Pi]) Exp[- (x - x0)^2 / (2 stdx^2 )],
       forceTermSelection == 4, a * Cos[b Pi x]
     ];

    eq = -y''[x] - k^2 * y[x] == f;
    Which[westBCtype == "Sommerfeld" && eastBCtype == "Dirichlet",
     (
      boundaryConditions = {Derivative[1][y][from] - I k y[from] == 0, y[to] == eastBCconstantValue}
     ),
     westBCtype == "Dirichlet" && eastBCtype == "Sommerfeld",
     (
      boundaryConditions = {y[from] == westBCconstantValue, Derivative[1][y][to] - I k y[to] == 0}
     ),
     westBCtype == "Dirichlet" && eastBCtype == "Dirichlet",
     (
      boundaryConditions = {y[from] == westBCconstantValue, y[to] == eastBCconstantValue}
     )
    ];

    sol = y /. First@Quiet@NDSolve[Flatten@{eq, boundaryConditions}, y, {x, from, to}, MaxSteps → Infinity];

    If[showGridLines,
     Framed[Plot[Re[sol[x]], {x, from, to}, Evaluate@plotOptions, GridLines → Automatic],
      FrameStyle -> Directive[Thickness[.005], Gray]
     ]
     ,
     Framed[Plot[Re[sol[x]], {x, from, to}, Evaluate@plotOptions],
      FrameStyle -> Directive[Thickness[.005], Gray]
     ]
    ]
   ];
 (*-----------------------------------------------------------*)
makeFinalPlot[u_, AA_, grid_, plotToShow_, showGridLines_] := Module[{finalDisplayImage},

    Which[
     plotToShow == "solution",
     (
      finalDisplayImage = Grid[{
          {ListPlot[Thread[{grid, u}],
            ImagePadding → {{45, 25}, {25, 20}},
            PlotRange → All,
            Joined → True,
            Mesh → All,
            AxesLabel → {Text@Style["x", Italic, 11],
```

```
            Text@Style[Row[{Style["u", Italic], "(", Style["x", Italic], ")"}], 11]},
           PlotLabel → Text@Style["finite difference solution"],
           ImageSize → {ContentSizeW - 20, ContentSizeH - 20},
           AspectRatio → 1,
           TicksStyle → 9,
           If[showGridLines, GridLines → Automatic, GridLines → None],
           AxesOrigin → {0, 0}
          ]
         }
        }, Spacings → {0, .5}, Alignment → Center, Frame → None, FrameStyle -> Directive[Thickness[.005], Gray]
       ]
     ),
     plotToShow == "solution data",
     (
      finalDisplayImage = makeScrolledPane[Normal@u, ContentSizeH - 350, ContentSizeW - 20]
     ),
     plotToShow == "system matrix information",
     (
      Block[{tmp, dim, m, cond},
       cond = LUDecomposition[Normal@AA][[3]];
       dim = Dimensions[Normal@AA];
       m = Min[20, First@dim];

       finalDisplayImage = Grid[{
          {Style[Text@Row[{"condition number = ", cond}], 12]},
          {Style[Text@Row[{"matrix size = ", dim}], 12]},
          {Style[Text["eigenvalues"], 12]},
          {makeScrolledPane[
            Re@Transpose@Partition[Eigenvalues[Normal@AA, m], 1], 45, ContentSizeW - 20]},
          {Style[Text["A matrix"], 12]},
          {makeScrolledPane[AA[[1 ;; m, 1 ;; m]], ContentSizeH - 150, ContentSizeW - 20]}
         }]
      ]
     )
    ];

    finalDisplayImage
   ];
 (*-------------------------------------------------------*)
myGrid[tab_, opts___] :=
  Module[{divlocal, divglobal, pos}, (*extract option value of Dividers from opts to divglobal*)
   (*default value is {False,False}*)divglobal = (Dividers /. {opts}) /. Dividers → {False, False};
   (*transform divglobal so that it is in the form {colspecs,rowspecs}*)
   If[Head[divglobal] =!= List, divglobal = {divglobal, divglobal}];
   If[Length[divglobal] == 1, AppendTo[divglobal, False]];
   (*Extract positions of dividers between rows from tab*)pos = Position[tab, Dividers → _, 1];
   (*Build list of rules for divider specifications between rows*)
   divlocal = MapIndexed[# - #2[[1]] + 1 → Dividers /. tab[[#]] &, Flatten[pos]];
   (*Final settings for dividers are {colspecs,{rowspecs,divlocal}}*)
   divglobal[[2]] = {divglobal[[2]], divlocal};
   Grid[Delete[tab, pos], Dividers → divglobal, opts]];
(*-------------------------------------------------------*)
MakeBoxes[Derivative[indices__][f_][vars__], TraditionalForm] := SubscriptBox[MakeBoxes[f, TraditionalForm],
   RowBox[Map[ToString, Flatten[Thread[dummyhead[{vars}, Partition[{indices}, 1]]] /. dummyhead → Table]]]];
(*-------------------------------------------------------*)
ContentSizeW = 295;
ContentSizeH = 415;
(*-------------------------------------------------------*)
padIt1[v_?(NumericQ[#] && Im[#] == 0 &), f_List] :=
  AccountingForm[Chop[N@v] , f, NumberSigns -> {"-", "+"}, NumberPadding -> {"0", "0"}, SignPadding -> True];
 (*-------------------------------------------------------*)
padIt2[v_?(NumericQ[#] && Im[#] == 0 &), f_List] :=
  AccountingForm[Chop[N@v] , f, NumberSigns -> {"", ""}, NumberPadding -> {"0", "0"}, SignPadding -> True];
```

```
(*-------------------------------------------------------*)
padIt3[v_?(NumericQ[#] && Im[#] == 0 &), f_List] := AccountingForm[v , f, NumberSigns -> {"", ""},
   NumberPadding -> {"0", "0"}, SignPadding -> True, NumberPoint → If[f[[2]] == 0, "", "."]];
(*-------------------------------------------------------*)
```

Manipulate

```
Manipulate[
 gtick;
 finalDisplayImage = process[h, centerGrid, kValue, n0, a, b, x0,
   stdx, forceTermSelection, plotToShow, westBCtype, westbc, westBCconstantValue,
   eastBCtype, eastbc, eastBCconstantValue, Unevaluated@gstatusMessage, showGridLines];
 FinishDynamic[];
 Framed[finalDisplayImage , FrameStyle -> Directive[Thickness[.005], Gray]],

 Evaluate@With[{
    (*------------------------*)
    (*--- plotOptions macro --*)
    (*------------------------*)
    plotOptions = Grid[{
       {
        Grid[{
          {
           PopupMenu[Dynamic[plotToShow, {plotToShow = #; gtick += del} &],
            {"solution" → Style["solution", 12],
             "solution data" → Style["solution data", 12],
             "system matrix information" → Style["system matrix", 12]
            },
            ImageSize -> All]
          }
         }, Alignment → Center,
         Spacings → {.7, .2}, Frame → None, FrameStyle -> Directive[Thickness[.005], Gray]
        ]
       }
      }],
    (*------------------------*)
    (*--- top row macro  -----*)
    (*------------------------*)
    topRow = Grid[{
       {
        Row[{Text@Style["k", Italic, 12],
          Spacer[2],
          Manipulator[Dynamic[kValue, {kValue = #; gtick += del} &],
           {0.0, 200, 1.0}, ImageSize → Tiny, ContinuousAction → False],
          Spacer[2],
          Text@Style[Dynamic@padIt3[kValue, {5, 0}], 12]
         }]
        ,
        Row[{
          Text@Style["k h", Italic, 12], " = ",
          Text@Style[Dynamic@padIt2[kValue*h, {5, 3}], 12]}
        ]
        ,
        Row[{
          Text@Style["PPW", 12], " = ",
          Dynamic[If[kValue == 0.0, Text@Style[Row[{"N/A", Spacer[19]}], 11],
            Text@Style[padIt2[2.0*Pi/(kValue*h), {6, 3}], 11]]]
         }]
       }
     }, Alignment → Left, Spacings → {0.6, 1}, Frame → All, FrameStyle -> Directive[Thickness[.005], Gray]
```

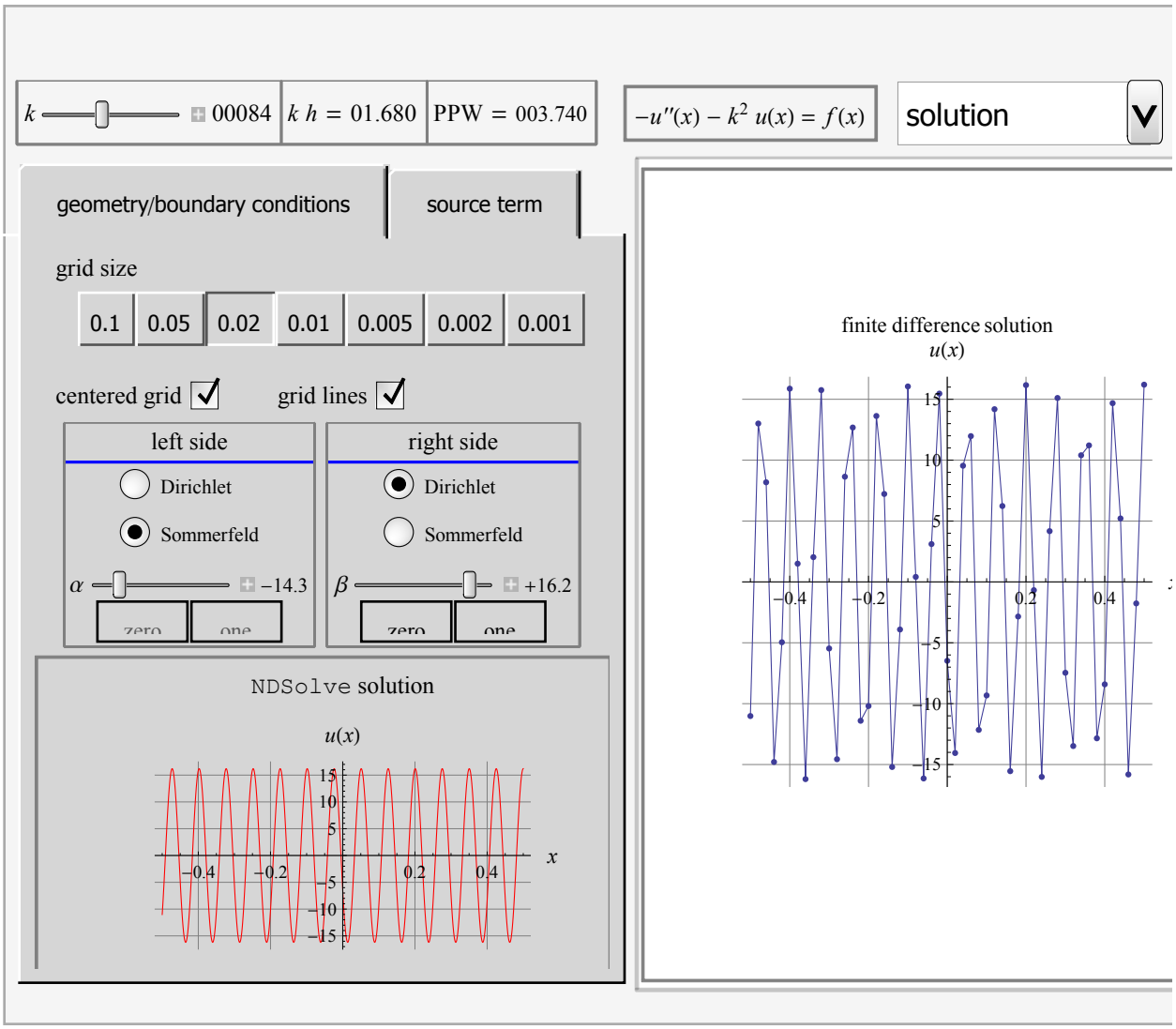
```
 ],
(*-------------------------*)
(*--- geometry macro      --*)
(*-------------------------*)
geometry = Item[Grid[{
     {Grid[{
        {
         Row[{Text@Style["grid size", 12], Spacer[2],
           SetterBar[Dynamic[h, {h = #; gtick += del} &],
            # → Style[#, 11] & /@ {0.1, 0.05, 0.02, 0.01, 0.005, 0.002, 0.001}]}]
        },
        {
         Grid[{
           {Row[{Text@Style["centered grid ", 12],
              Checkbox[Dynamic[centerGrid, {centerGrid = #; gtick += del} &]
              ]}]
            ,
            Row[{Text@Style["grid lines ", 12],
              Checkbox[Dynamic[showGridLines, {showGridLines = #; gtick += del} &]
              ]}]
           }
          }, Alignment → Left,
          Spacings → {2, 0}, Frame → None, FrameStyle -> Directive[Thickness[.005], Gray]
         ]

        }
       }, Spacings → {0, 1}, Alignment → Left]
     },
     {Grid[{
        {
         myGrid[
          {
           {
            Text@Style["left side", 12]
           },
           Dividers → {Thin, Blue},
           {
            RadioButtonBar[Dynamic[westBCtype, {
                 westBCtype = #;
                 If[westBCtype == "Sommerfeld" && eastBCtype == "Sommerfeld", westBCtype = "Dirichlet",
                  gtick += del]} &], {"Dirichlet" → Text@Style["Dirichlet", 10],
               "Sommerfeld" → Text@Style["Sommerfeld", 10]}, Appearance → "Vertical"]
           }
           ,
           {
            Grid[{
              {
               Spacer[2],
               Text@Style[α, 12],
               Spacer[2],
               Manipulator[Dynamic[westBCconstantValue,
                 {westBCconstantValue = #; gtick += del} &], {-20, 20, 0.1}, ImageSize -> Tiny,
                ContinuousAction -> False, Enabled → Dynamic[westBCtype == "Dirichlet"]],
               Spacer[1],
               Text@Style[Dynamic@padIt1[westBCconstantValue, {3, 1}], 10],
               Spacer[2]
              }
              ,
              {
               Row[{Button[Text@Style["zero", 11], {westBCconstantValue = 0.0; gtick += del},
                   ImageSize → {45, 20}, Enabled → Dynamic[westBCtype == "Dirichlet"]],
                  Spacer[2],
                  Button[Text@Style["one", 11], {westBCconstantValue = 1.0; gtick += del},
                   ImageSize → {45, 20}, Enabled → Dynamic[westBCtype == "Dirichlet"]]
```

```
                }],
               SpanFromLeft
              }
             }, Alignment → Center, Spacings → {0, 0}
            ]
           }
          }, Spacings → {0, .4}, Alignment → Center,
          Dividers → True, FrameStyle -> Directive[Thickness[.005], Gray]],
         Spacer[5],
         myGrid[{
           {Text@Style["right side", 12]},
           Dividers → {Thin, Blue},
           {
            RadioButtonBar[Dynamic[eastBCtype, {eastBCtype = #;
                 If[eastBCtype == "Sommerfeld" && westBCtype == "Sommerfeld", eastBCtype = "Dirichlet",
                  gtick += del]} &], {"Dirichlet" → Text@Style["Dirichlet", 10],
              "Sommerfeld" → Text@Style["Sommerfeld", 10]}, Appearance → "Vertical"]
           },
           {Grid[{
              {Spacer[2],
               Text@Style[β, 12],
               Spacer[2],
               Manipulator[Dynamic[eastBCconstantValue,
                 {eastBCconstantValue = #; gtick += del} &], {-20, 20, 0.1}, ImageSize -> Tiny,
                ContinuousAction -> False, Enabled → Dynamic[eastBCtype == "Dirichlet"]],
               Spacer[1],
               Text@Style[Dynamic@padIt1[eastBCconstantValue, {3, 1}], 10],
               Spacer[2]
              },
              {
               Row[{Button[Text@Style["zero", 11], {eastBCconstantValue = 0.0; gtick += del},
                   ImageSize → {45, 20}, Enabled → Dynamic[eastBCtype == "Dirichlet"]],
                  Spacer[2],
                  Button[Text@Style["one", 11], {eastBCconstantValue = 1.0; gtick += del},
                   ImageSize → {45, 20}, Enabled → Dynamic[eastBCtype == "Dirichlet"]]
                }],
               SpanFromLeft
              }
             }, Alignment → Center, Spacings → {0, 0}]
           }
          }, Spacings → {.1, .4}, Alignment → Center,
          Dividers → True, FrameStyle -> Directive[Thickness[.005], Gray]]
        }
       }, Alignment → Center, Spacings → {0, 0.15}]
     },
     {
      Grid[{
        {Dynamic[getNDsolveResult[kValue,
           westBCtype, eastBCtype, westBCconstantValue, eastBCconstantValue, forceTermSelection,
           a, b, n0, x0, stdx, centerGrid, showGridLines]
         ]}
       }]
     }

    }, Alignment → Center, Spacings → {0, .3}
   ], Alignment → {Center, Top}],
(*-----------------------------------*)
(*-- source macro                 --*)
(*-----------------------------------*)
source = Item[Grid[
    {
     {PopupMenu[Dynamic[forceTermSelection, {forceTermSelection = #; gtick += del} &],
```

```
   {1 → Style["a", Italic, 12],
    2 → Style[Row[{Style["a", Italic], Style["x", Italic]^Style["n",Italic]0}], 12],
    3 → Style[Row[{ Style["a", Italic] / (σ √(2 Pi)), "exp (", 1 / (2 σ^2)
          Row[{"( ", Style["x", Italic], " - ", Style["x", Italic]0, " )"^2}], " )"}], 12],
    4 → Style[Row[{Style["a", Italic], " ", "cos ( ", Style["b", Italic],
        " ", π, " ", Style["x", Italic], " )"}] , 12]
   }, ImageSize → {260, 45}, ContinuousAction -> False]
}
,
{
 Grid[{
   {
    Text@Style["a", Italic, 12],
    Manipulator[Dynamic[a, {a = #; gtick += del} &],
     {-10, 10, 0.1}, ImageSize → Small, ContinuousAction -> False],
    Text@Style[Dynamic@padIt1[a, {3, 1}], 11],
    Button[Text@Style["zero", 10], {a = 0; gtick += del}, ImageSize → {45, 20}, Alignment → Center]
   }
   ,
   {
    Text@Style["b", Italic, 12],
    Manipulator[Dynamic[b, {b = #; gtick += del} &], {-10, 10, 0.1}, ImageSize → Small,
     ContinuousAction -> False, Enabled → Dynamic[forceTermSelection == 4]],
    Text@Style[Dynamic@padIt1[b, {3, 1}], 11],
    Button[Text@Style["zero", 10], {b = 0; gtick += del}, ImageSize → {45, 20}, Alignment → Bottom]
   }
   ,
   {
    Text@Style[Style["n", Italic]0, 12],
    Manipulator[Dynamic[n0, {n0 = #; gtick += del} &], {0., 10., .1}, ImageSize → Small,
     ContinuousAction -> False, Enabled → Dynamic[forceTermSelection == 2]],
    Text@Style[Dynamic@padIt2[n0, {3, 1}], 11],
    Button[Text@Style["zero", 10], {n0 = 0.; gtick += del},
     ImageSize → {45, 20}, Alignment → Bottom, BaselinePosition → Center]
   }
   ,
   {
    Text@Style[Style["x", Italic]0, 12],
    Manipulator[Dynamic[x0, {x0 = #; gtick += del} &], {-1.5, 1.5, 0.01}, ImageSize → Small,
     ContinuousAction -> False, Enabled → Dynamic[forceTermSelection == 3]],
    Text@Style[Dynamic@padIt1[x0, {3, 2}], 11],
    Button[Text@Style["zero", 10], {x0 = 0.0; gtick += del},
     ImageSize → {45, 20}, Alignment → Bottom, BaselinePosition → Center]
   }
   ,
   {
    Text@Style[σ, 12],
    Manipulator[Dynamic[stdx, {stdx = #; gtick += del} &], {0.01, 1, 0.01}, ImageSize → Small,
     ContinuousAction -> False, Enabled → Dynamic[forceTermSelection == 3]],
    Text@Style[Dynamic@padIt2[stdx, {3, 2}], 11],
    ""
   }
  }, Spacings → {.4, .1}, Alignment → Center, FrameStyle -> Directive[Thickness[.005], Gray]
 ]
}
,
{
 Dynamic[
  Block[{grid, forceGrid},
   grid = N[generatePhysicalCoordinates1D[h, 1, centerGrid]];
```

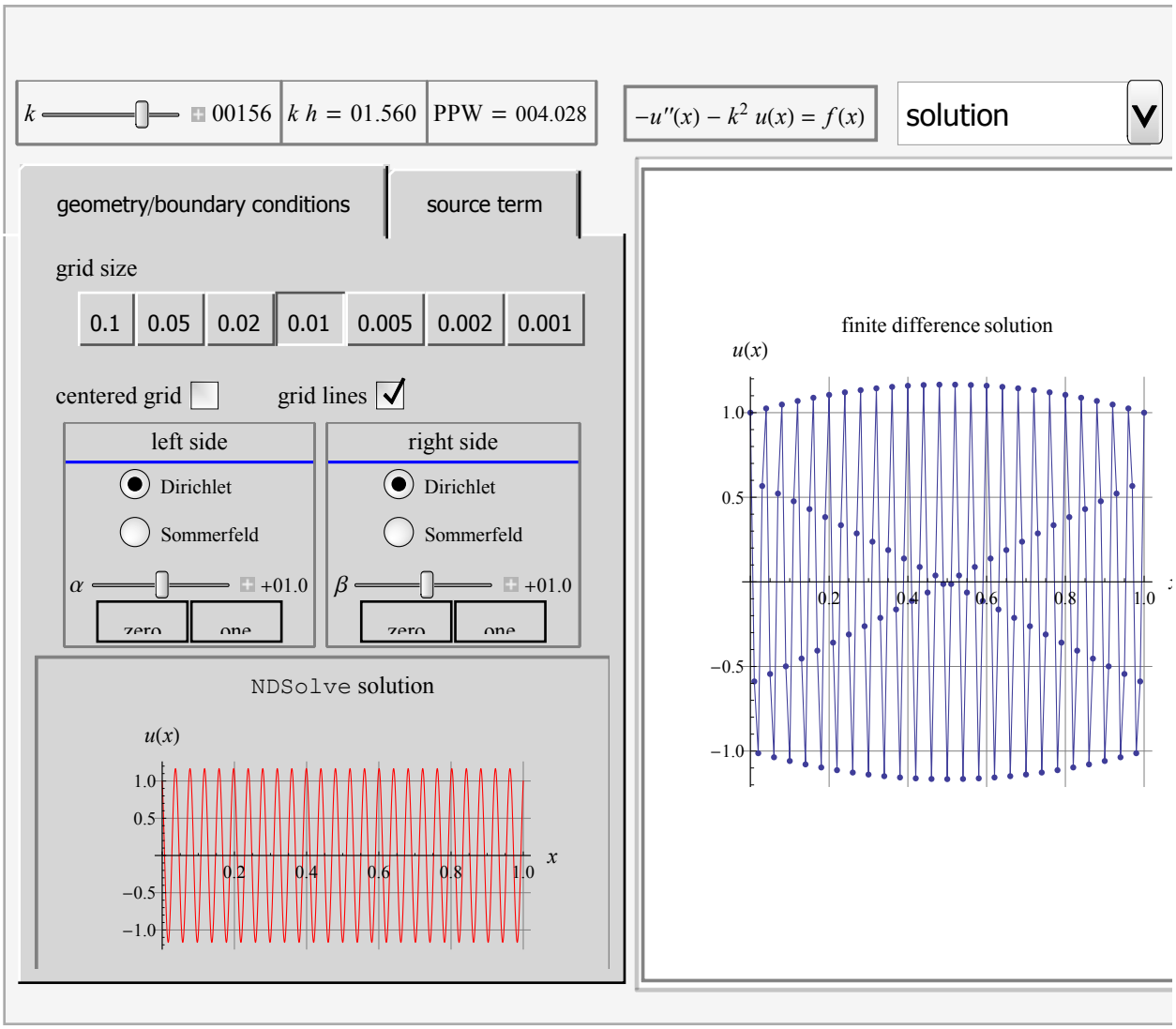
```
          forceGrid = makeForceGrid[a, b, n0, x0, stdx, Length[grid], forceTermSelection, grid];

          ListPlot[Thread[{grid, forceGrid}],
           ImagePadding → {{40, 15}, {40, 65}},
           ImageMargins → 1,
           PlotRange → All,
           Mesh → All,
           Axes → None,
           If[showGridLines, GridLines → Automatic, GridLines → None],
           PlotStyle → Red,
           Joined → True,
           Frame → True,
           FrameLabel → {{None, None},
             {Text@Style["x", Italic, 11],
              Text@Style[Row[
                  {f[x], " = ", forceTermUsedFormat1D[forceTermSelection, a, b, stdx, x0, n0, x]}], 12]}
            },
           ImageSize → {ContentSizeW - 10, ContentSizeH - 240},
           AspectRatio → 0.3,
           TicksStyle → 9
          ]
         ]
        ]
       }

     }, Spacings → {0, .4}, Alignment → Center, Frame → All, FrameStyle -> Directive[Thickness[.005], Gray]
    ], Alignment → {Center, Top}]
},
(*------------------------------*)
(*--- LEVEL 2             -----*)
(*------------------------------*)
With[{
  pde = Grid[{
      {TabView[{
         Style["geometry/boundary conditions", 11] → geometry,
         Style["source term", 11] → source
        }, ImageSize → {305 , 410 }]
      }
     }, Spacings → {0.2, .9}
    ]
 },
 (*--- end of level 2 ---*)
 ## &[
  Item[

   Grid[{
     {
      Grid[{{topRow}}],
      Grid[{{
         Framed[Text@Style[Row[{"-", Style["u''", Italic], "(", Style["x", Italic], ") - ", Style["k",
                 Italic]^2, " ", Style["u", Italic], "(", Style["x", Italic], ") = ", Style[ "f", Italic],
               "(", Style["x", Italic], ")"}], 12], FrameStyle -> Directive[Thickness[.005], Gray]],
         plotOptions
        }}]}
    }, Alignment → Center, Spacings → {1, 0}
   ],
   ControlPlacement → Top
  ],

  Item[pde, ControlPlacement → Left]
 ]
]
```

```
 ],
(*----------- end of Manipulate controls --------------------------*)
{{gstatusMessage, "reseting..."}, None},
{{gtick, 0}, None},
{{del, $MachineEpsilon}, None},

{{centerGrid, False}, None},
{{finalDisplayImage, {}}, None},
{{h, 0.02}, None},
{{kValue, 10.0}, None},
{{n0, 2.}, None},
{{a, 1.0}, None},
{{b, 0.0}, None},
{{x0, 0.0}, None},
{{stdx, 0.3}, None},
{{forceTermSelection, 3}, None},
{{plotToShow, "solution"}, None},
{{westBCtype, "Dirichlet"}, None},
{{westbc, (1) &}, None},
{{westBCconstantValue, 0}, None},
{{eastBCtype, "Dirichlet"}, None},
{{eastbc, (1) &}, None},
{{eastBCconstantValue, 0}, None},
{{showGridLines, True}, None},

ControlPlacement → Left,
SynchronousInitialization → True,
SynchronousUpdating → False,
ContinuousAction → False,
Alignment -> Center,
ImageMargins → 0,
FrameMargins → 0,
TrackedSymbols :→ {gtick},
Paneled → True,
Frame → False,
SaveDefinitions → True
]
```



Caption

This Demonstration implements a recently published algorithm of an improved finite difference scheme for solving the Helmholtz differential equation $-U_{xx} - k^2 U = f(x)$ in one dimension. Dirichlet and Sommerfeld boundary conditions are supported. Different source function $f(x)$ can be specified. Sommerfeld boundary conditions can be specified at either end of the domain but not at both ends at the same time. The current value of points per wavelength (PPW) is shown at the top of the display. You can vary the k value. You can view the matrix A and its eigenvalues using the pull down menu in the top row.



Details (optional)

Details of the algorithm are described in [1]. This implementation converts the finite difference scheme to the standard $A\,u = b$ form and uses the built-in *Mathematica* function `LinearSolve` to obtain the solution. Sparse matrices are used. The matrix A and its eigenvalues and the numerical solution vector u can be viewed using the pull down menu. The table below summarizes the boundary conditions where α and β are the values of left and right boundaries, L is the length of the domain, and U_0 and U_N are the solution at left and right edge of the domain.

edge	Dirichlet	Sommerfeld
west	$U_0 = \alpha$	$\alpha = \left(\frac{dU}{dx}\right)_{x=0} = i\,k\,U_0$
east	$U_N = \beta$	$\beta = \left(\frac{dU}{dx}\right)_{x=L} = i\,k\,U_N$

The PDE $-U_{xx} - k^2\,U = f(x)$ is discretized using $-\frac{U_{i-1} - \omega\,U_i + U_{i+1}}{h^2} - k^2\,U_i = f_i$, where $\omega = 2\cos(k\,h) + (k\,h)^2$. The points per wavelength (PPW) number is calculated using $\frac{\lambda}{h} = \frac{2\pi}{k\,h}$.

References:

[1] Y. S. Wong and G. Li, exact finite difference schemes for solving Helmholtz equation at any wavenumber. International journal of numerical analysis and modeling, series b, Computing and Information Volume 2, Number 1, 2010. pp. 91–108.

Control Suggestions (optional)

- [x] Resize Images
- [ ] Rotate and Zoom in 3D
- [ ] Drag Locators
- [ ] Create and Delete Locators
- [x] Slider Zoom
- [ ] Gamepad Controls
- [ ] Automatic Animation
- [ ] Bookmark Animation

Search Terms (optional)

Helmholtz
finite difference method

Related Links (optional)

Helmholtz differential equation

Authoring Information

Contributed by: Nasser M. Abbasi