Finite Difference Solution of the 2D Helmholtz Partial Differential Equation

Initialization Code (optional)

```
evaluateForceCommon = Unevaluated[#3 #1^#6 + #4 #2^#7] /. HoldPattern[0.0^0.0] :> 0.0 &;
(*-----------------------------------------*)
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];
(*-----------------------------------------*)
(* returns {number of rows, number of columns} in a list *)
getSolutionDomainDimensions[hx_, hy_, Lx_, Ly_] := {Ly/hy + 1, Lx/hx + 1};
(*-----------------------------------------*)
makeGridCommon[hx_, hy_, Lx_, Ly_, centerGrid_] := Module[{i, j, nx, ny, grid},
   (*ny is number of rows, nx is number of columns*)
   {ny, nx} = getSolutionDomainDimensions[hx, hy, Lx, Ly];

   With[{$icfrom = Floor[ny/2], $icto = -Floor[ny/2],
     $jcfrom = -Floor[nx/2], $jcto = Floor[nx/2], $ifrom = ny - 1, $jto = nx - 1},
    grid = If[centerGrid,
      Table[{j*hx, i*hy}, {i, $icfrom, $icto, -1}, {j, $jcfrom, $jcto}],
      Table[{j*hx, i*hy}, {i, $ifrom, 0, -1}, {j, 0, $jto}]
     ]
   ];

   N@grid
  ];
(*-----------------------------------------*)
forceTermUsedFormatCommon[forceTermSelection_, a_, b_, c_, stdy_, stdx_, x0_, y0_, x_, y_] :=
  Module[{forceTermUsed},
   Which[
    forceTermSelection == 1, forceTermUsed = a ,
    forceTermSelection == 3,
    forceTermUsed = HoldForm[a Exp[- ( (x - x0)^2/(2 (stdx)^2) + (y - y0)^2/(2 (stdy)^2)) ]],
    forceTermSelection == 4, forceTermUsed = HoldForm[a (Cos[b Pi x] + Sin[c Pi y])],
    forceTermSelection == 5, forceTermUsed = HoldForm[a (Cos[b Pi x]*Sin[c Pi y])]
   ];

   forceTermUsed
  ];
(*-----------------------------------------*)
forceTermExpressionCommon[forceTermSelection_, a_, b_, c_, stdy_, stdx_, x0_, y0_, x_, y_] :=
  Module[{forceTermUsed},
   Which[
    forceTermSelection == 1, forceTermUsed = a ,
    forceTermSelection == 3, forceTermUsed = a Exp[- ( (x - x0)^2/(2 (stdx)^2) + (y - y0)^2/(2 (stdy)^2)) ],
    forceTermSelection == 4, forceTermUsed = a (Cos[b Pi x] + Sin[c Pi y]),
    forceTermSelection == 5, forceTermUsed = a (Cos[b Pi x]*Sin[c Pi y])
   ];

   forceTermUsed
  ];
(*-----------------------------------------*)
```

```
setCornerNodeCommon[$u_, i_, j_, nx_, ny_] := Module[{u = $u},
    Which[
     i == 1 && j == 1, u[[1, 1]] = Mean[{u[[2, 1]], u[[1, 2]]}],
     i == 1 && j == nx, u[[1, nx]] = Mean[{u[[1, nx - 1]], u[[2, nx]]}],
     i == ny && j == 1, u[[ny, 1]] = Mean[{u[[ny - 1, 1]], u[[ny, 2]]}],
     i == ny && j == nx, u[[ny, nx]] = Mean[{u[[ny, nx - 1]], u[[ny - 1, nx]]}]
    ];
    u
   ];
(*----------------------------------------*)
makeScrolledPane[mat_?(MatrixQ[#, NumberQ] &),
    nRow_?(IntegerQ[#] && Positive[#] &), nCol_?(IntegerQ[#] && Positive[#] &)] := Module[{t},
    t = Grid[mat, Spacings → {.4, .4}, Alignment → Left, Frame → All];
    t = Text@Style[NumberForm[Chop[N@t], {6, 5}, NumberSigns -> {"-", ""},
        NumberPadding -> {"", ""}, SignPadding -> True], LineBreakWithin → False];
    Pane[t, ImageSize → {nCol, nRow}, Scrollbars → True]
   ];
(*----------------------------------------*)
makeScrolledPane[lst_?(VectorQ[#, NumericQ] &),
    nRow_?(IntegerQ[#] && Positive[#] &), nCol_?(IntegerQ[#] && Positive[#] &)] := Module[{t},
    t = Grid[{lst}, Spacings → {.4, .4}, Alignment → Left, Frame → All];
    t = Text@Style[AccountingForm[Chop[N@t], {6, 5}, NumberSigns -> {"-", ""},
        NumberPadding -> {"", ""}, SignPadding -> True], LineBreakWithin → False];
    Pane[t, ImageSize → {nCol, nRow}, Scrollbars → True]
   ];
(*----------------------------------------*)
process[$u_, $forceGrid_, $grid_, $AA_, $rightHandVector_, $finalDisplayImage_, event_, hx_, hy_,
    Lx_, Ly_, centerGrid_, addFaceGrids_, plotPerformanceGoal_, kValue_, a_, b_, c_, x0_, y0_,
    stdx_, stdy_, forceTermSelection_, plotToShow_, northBCtype_, northbc_, northBCconstantValue_,
    westBCtype_, westbc_, westBCconstantValue_, eastBCtype_, eastbc_, eastBCconstantValue_,
    southBCtype_, southbc_, southBCconstantValue_, zAxisScale_, angle_, gstatusMessage_] :=
  Module[{u = $u, AA = $AA, forceGrid = $forceGrid, grid = $grid,
    rightHandVector = $rightHandVector, finalDisplayImage = $finalDisplayImage},

    If[StringMatchQ[event, {"reset", "run_button"}],
     (
      {grid, forceGrid, u, AA, rightHandVector} =
       initializeSystem[hx, hy, Lx, Ly, centerGrid, forceTermSelection, a, b, c, x0, stdx, y0, stdy,
        northBCtype, northbc, northBCconstantValue, westBCtype, westbc, westBCconstantValue,
        eastBCtype, eastbc, eastBCconstantValue, southBCtype, southbc, southBCconstantValue, kValue, angle
       ]
     )
    ];

    If[event == "run_button" || (StringMatchQ[event, {"reset", "reset", "k_changed"}] &&
        (StringMatchQ[plotToShow, {"system matrix information", "solution data"}])),
     (
      u = solve[u, AA, rightHandVector, northBCtype, westBCtype, eastBCtype, southBCtype]
     )
    ];

    finalDisplayImage =
     makeFinalPlot[u, AA, Lx, Ly, grid, plotToShow, plotPerformanceGoal, addFaceGrids, zAxisScale];
    gstatusMessage = "ready..";
    {finalDisplayImage, u, forceGrid, grid, AA, rightHandVector}
  ];
(*----------------------------------------*)
(* The main function where the matrix A and the RHS vector b are generated *)
makeSystemMatrixAndRightHandSide[u_, h_, kValue_,
    northBCtype_, westBCtype_, eastBCtype_, southBCtype_, forceGrid_, angle_] :=
  Module[{AA, k1, k2, i, j, n = 0, eqs, vars, uu, U, An, b, keepList, nRow, nCol, sin1, sin2, omega, sum, prod},
```

```
sum = 2.0 * h^2;
prod = 1.0 * h^4;
omega = 4.0 BesselJ[0, Sqrt[kValue] h] + (Sqrt[kValue] h)^2;
k1 = Sqrt[kValue] Cos[angle];
k2 = Sqrt[kValue] Cos[angle];
sin1 = Simplify[2. I Sin[k1 h]];
sin2 = Simplify[2. I Sin[k2 h]];
{nRow, nCol} = Dimensions[u];
U = Array[uu[#1, #2] &, {nRow, nCol}];

With[{$nRow = nRow, $nCol = nCol},
 eqs = Table[0., {$nRow * $nCol}];
 vars = eqs
];

For[i = 1, i ≤ nRow, i++,
 For[j = 1, j ≤ nCol, j++,
  (
   n++;
   (*do corner points first *)
   If[(i == 1 && j == 1 || i == nRow && j == 1 || i == 1 && j == nCol || i == nRow && j == nCol),
    (
     {eqs, vars} = processCornersCommon[u, eqs, n,
       U, i, j, vars, northBCtype, westBCtype, southBCtype, eastBCtype, nRow, nCol]
    ), (*completed corner nodes, now check if node on edge *)
    (
     If[(i == 1 || i == nRow || j == 1 || j == nCol),
      (
       Which[i == 1,
        If[northBCtype == "Dirichlet",
         (
          eqs[[n]] = U[[i, j]] == u[[i, j]];
          vars[[n]] = U[[i, j]]
         ),
         (
          eqs[[n]] = U[[i, j]] - sin1 * U[[i + 1, j]] - U[[i + 2, j]] == 0;
          vars[[n]] = U[[i, j]]
         )]
        , i == nRow,
        If[southBCtype == "Dirichlet",
         (
          eqs[[n]] = U[[i, j]] == u[[i, j]];
          vars[[n]] = U[[i, j]]
         ),
         (
          eqs[[n]] = U[[i, j]] - sin1 * U[[i - 1, j]] - U[[i - 2, j]] == 0;
          vars[[n]] = U[[i, j]]
         )
        ], j == 1,
        If[westBCtype == "Dirichlet",
         (
          eqs[[n]] = U[[i, j]] == u[[i, j]];
          vars[[n]] = U[[i, j]]
         ),
         (
          eqs[[n]] = U[[i, j]] - sin2 * U[[i, j + 1]] - U[[i, j + 2]] == 0;
          vars[[n]] = U[[i, j]]
         )], j == nCol,
        If[eastBCtype == "Dirichlet",
         (
          eqs[[n]] = U[[i, j]] == u[[i, j]];
          vars[[n]] = U[[i, j]]
         ),
         (
```

```
              eqs[[n]] = U[[i, j]] - sin2 * U[[i, j - 1]] - U[[i, j - 2]] == 0;
              vars[[n]] = U[[i, j]]
             )
            ]
           ]
          ), (*was not edge, so must be internal*)
          (
           vars[[n]] = U[[i, j]];
           eqs[[n]] = -U[[i + 1, j]] - U[[i - 1, j]] + omega * U[[i, j]] -
              U[[i, j - 1]] - U[[i, j + 1]] - (kValue * h) ^ 2 U[[i, j]] == h ^ 2 * forceGrid[[i, j]]
          )
         ]
        )
       ]
      )
     ]
    ];

    vars = Flatten@U;
    AA = CoefficientArrays[eqs, vars];
    keepList = obtainListOfRowsToKeep[nRow, nCol, northBCtype, southBCtype, westBCtype, eastBCtype];
    An = AA[[2]][[keepList, keepList]];
    b = -AA[[1]][[keepList]];

    {An, b}
   ];
(*-----------------------------------------*)
solve[$u_, AA_, rightHandVector_, northBCtype_, westBCtype_, eastBCtype_, southBCtype_] :=
  Module[{u = $u, nRow, nCol, x, loc, mask},
   {nRow, nCol} = Dimensions[u];
   x = LinearSolve[AA, rightHandVector];
   mask = setUnknownsMask[{nRow, nCol}, northBCtype, westBCtype, eastBCtype, southBCtype];
   loc = Position[mask, 1];
   MapThread[(u = ReplacePart[u, #1 → #2 ]) &, {loc, x}];
   Re@u
  ];
(*-----------------------------------------*)
setUnknownsMask[{nRow_, nCol_}, northBCtype_, westBCtype_, eastBCtype_, southBCtype_] := Module[{mask},
   (*there are 7 cases to check for. Let T=top edge, R=right edge, L=left edge B=bottom edge,
   then we need to check for one of these cases: no Nuemman on any edge, {LTRB},{TL,TR,TB},
   {TLR,TBR,TBL},{LB},{LR} {LBR} {BR}.  mask is used to tell location of unknowns in the
    solution grid. This is needed since now we have Sommerfeld BC and so nodes on the edge
    can be part of the unknowns and we need later to find the location of the unknowns*)

   mask = Table[0, {nRow}, {nCol}];
   Which[westBCtype == "Dirichlet" && northBCtype == "Dirichlet" &&
     eastBCtype == "Dirichlet" && southBCtype == "Dirichlet", mask[[2 ;; -2, 2 ;; -2]] = 1
    ,
    northBCtype == "Sommerfeld" && westBCtype == "Dirichlet" &&
     eastBCtype == "Dirichlet" && southBCtype == "Dirichlet", mask[[1 ;; -2, 2 ;; -2]] = 1
    ,
    northBCtype == "Dirichlet" && westBCtype == "Sommerfeld" &&
     eastBCtype == "Dirichlet" && southBCtype == "Dirichlet", mask[[2 ;; -2, 1 ;; -2]] = 1
    ,
    northBCtype == "Dirichlet" && westBCtype == "Dirichlet" &&
     eastBCtype == "Sommerfeld" && southBCtype == "Dirichlet", mask[[2 ;; -2, 2 ;; -1]] = 1
    ,
    northBCtype == "Dirichlet" && westBCtype == "Dirichlet" &&
     eastBCtype == "Dirichlet" && southBCtype == "Sommerfeld", mask[[2 ;; -1, 2 ;; -2]] = 1
    ,
    (* now do the checks for {TL,TR,TB} case*)
    northBCtype == "Sommerfeld" && westBCtype == "Sommerfeld" &&
     eastBCtype == "Dirichlet" && southBCtype == "Dirichlet", mask[[1 ;; -2, 1 ;; -2]] = 1
    ,
```

```
      northBCtype == "Sommerfeld" && westBCtype == "Dirichlet" &&
       eastBCtype == "Sommerfeld" && southBCtype == "Dirichlet", mask[[1 ;; -2, 2 ;; -1]] = 1
      ,
      northBCtype == "Sommerfeld" && westBCtype == "Dirichlet" &&
       eastBCtype == "Dirichlet" && southBCtype == "Sommerfeld", mask[[1 ;; -1, 2 ;; -2]] = 1
      ,
      northBCtype == "Sommerfeld" && westBCtype == "Sommerfeld" &&
       eastBCtype == "Sommerfeld" && southBCtype == "Dirichlet", mask[[1 ;; -2, 1 ;; -1]] = 1
      ,
      northBCtype == "Sommerfeld" && westBCtype == "Dirichlet" &&
       eastBCtype == "Sommerfeld" && southBCtype == "Sommerfeld", mask[[1 ;; -1, 2 ;; -1]] = 1
      ,
      northBCtype == "Sommerfeld" && westBCtype == "Sommerfeld" &&
       eastBCtype == "Dirichlet" && southBCtype == "Sommerfeld", mask[[1 ;; -1, 1 ;; -2]] = 1
      ,
      northBCtype == "Dirichlet" && westBCtype == "Sommerfeld" &&
       eastBCtype == "Dirichlet" && southBCtype == "Sommerfeld", mask[[2 ;; -1, 1 ;; -2]] = 1
      ,
      northBCtype == "Dirichlet" && westBCtype == "Sommerfeld" &&
       eastBCtype == "Sommerfeld" && southBCtype == "Dirichlet", mask[[2 ;; -2, All]] = 1
      ,
      northBCtype == "Dirichlet" && westBCtype == "Sommerfeld" &&
       eastBCtype == "Sommerfeld" && southBCtype == "Sommerfeld", mask[[2 ;; -1, All]] = 1
      ,
      northBCtype == "Dirichlet" && westBCtype == "Dirichlet" &&
       eastBCtype == "Sommerfeld" && southBCtype == "Sommerfeld", mask[[2 ;; -1, 2 ;; -1]] = 1

     ];
     mask
    ];
(*-----------------------------------------*)
processCornersCommon[u_, $eqs_, n_, U_, i_, j_, $vars_, northBCtype_,
    westBCtype_, southBCtype_, eastBCtype_, nRow_, nCol_] := Module[{vars = $vars, eqs = $eqs},

    vars[[n]] = U[[i, j]];
    Which[i == 1 && j == 1, (*top left-top corner*)
     (
      Which[northBCtype == "Dirichlet" || westBCtype == "Dirichlet",
       If[northBCtype == "Dirichlet" && westBCtype == "Dirichlet",
        eqs[[n]] = U[[i, j]] == Mean[{u[[1, 2]], u[[2, 1]]}]
        ,
        eqs[[n]] = U[[i, j]] == u[[1, 1]]
       ]
       , True, (*both edgs are not Dirichlet*)
       eqs[[n]] = U[[i, j]] - Mean[{U[[1, 2]], U[[2, 1]]}] == 0
      ]
     ),
     i == nRow && j == 1, (*bottom left corner*)
     (
      Which[southBCtype == "Dirichlet" || westBCtype == "Dirichlet",
       If[southBCtype == "Dirichlet" && westBCtype == "Dirichlet",
        eqs[[n]] = U[[i, j]] - Mean[{u[[i, 2]], u[[i - 1, 1]]}] == 0
        ,
        eqs[[n]] = U[[i, j]] == u[[nRow, 1]]
       ]
       , True, (*both edgs are not Dirichlet*)
       eqs[[n]] = U[[i, j]] - Mean[{U[[i, 2]], U[[i - 1, 1]]}] == 0.0
      ]
     ), (*top right corner*)
     i == 1 && j == nCol,
     (
      Which[northBCtype == "Dirichlet" || eastBCtype == "Dirichlet",
       If[northBCtype == "Dirichlet" && eastBCtype == "Dirichlet",
        eqs[[n]] = U[[i, j]] - Mean[{U[[1, j - 1]], U[[2, j]]}] == 0.0,
```

```
        eqs[[n]] = U[[i, j]] == u[[1, nCol]]
       ]
       , True, (*both edgs are not Dirichlet*)
       eqs[[n]] = U[[i, j]] - Mean[{U[[1, j - 1]], U[[2, j]]}] == 0.0;
      ]
     ),
     i == nRow && j == nCol, (*both edgs are not Dirichlet*)
     (
      Which[southBCtype == "Dirichlet" || eastBCtype == "Dirichlet",
       If[southBCtype == "Dirichlet" && eastBCtype == "Dirichlet",
        eqs[[n]] = U[[i, j]] - Mean[{U[[i, j - 1]], U[[i - 1, j]]}] == 0.0,
        eqs[[n]] = U[[i, j]] - u[[nRow, nCol]]
       ]
       , True, (*both edgs are nuemman*)
       eqs[[n]] = U[[i, j]] - Mean[{U[[i, j - 1]], U[[i - 1, j]]}] == 0.0
      ]
     )
    ];
    {eqs, vars}
   ];
(*----------------------------------------*)
initializeSystem[hx_, hy_, Lx_, Ly_, centerGrid_, forceTermSelection_, a_, b_, c_, x0_, stdx_, y0_, stdy_,
    northBCtype_, northbc_, northBCconstantValue_, westBCtype_, westbc_, westBCconstantValue_, eastBCtype_,
    eastbc_, eastBCconstantValue_, southBCtype_, southbc_, southBCconstantValue_, kValue_, angle_] :=
  Module[{nCol, nRow, grid, forceGrid, u, AA, rightHandVector},

    (*grid contains the (x,y) physical coordinates of each grid point*)
    {nRow, nCol} = getSolutionDomainDimensions[hx, hy, Lx, Ly];

    With[{$nRow = nRow, $nCol = nCol}, u = Table[0., {$nRow}, {$nCol}]];

    grid = makeGridCommon[hx, hy, Lx, Ly, centerGrid];
    (*evaluate the source function at each physical coordiate, using the selected term*)
    forceGrid = Which[
      forceTermSelection == 1,
      With[{$nRow = nRow, $nCol = nCol}, Table[a, {$nRow}, {$nCol}]],

      forceTermSelection == 2,
      Map[evaluateForceCommon[#[[1]], #[[2]], a, b, c] & , grid, {2}],

      forceTermSelection == 3,
      Map[(a Exp[ (#[[1]] - x0) ^ 2 / (2 stdx ^ 2) + (#[[2]] - y0) ^ 2 / (2 stdy ^ 2) ]) & , grid, {2}],

      forceTermSelection == 4,
      Map[(a (Cos[b π #[[1]]] + Sin[c π #[[2]]])) & , grid, {2}],

      forceTermSelection == 5,
      Map[(a (Cos[b π #[[1]]] * Sin[c π #[[2]]])) & , grid, {2}]
     ];

    forceGrid = Re[forceGrid];
    u = setBoundaryConditions[u, grid, northBCtype, northbc,
      northBCconstantValue, westBCtype, westbc, westBCconstantValue, eastBCtype,
      eastbc, eastBCconstantValue, southBCtype, southbc, southBCconstantValue];

    {AA, rightHandVector} = makeSystemMatrixAndRightHandSide[u,
      hx, kValue, northBCtype, westBCtype, eastBCtype, southBCtype, forceGrid, angle];

    {grid, forceGrid, u, AA, rightHandVector}
   ];
(*----------------------------------------*)
setBoundaryConditions[$u_, grid_, northBCtype_, northbc_, northBCconstantValue_,
    westBCtype_, westbc_, westBCconstantValue_, eastBCtype_, eastbc_, eastBCconstantValue_,
```

```
    southBCtype_, southbc_, southBCconstantValue_] := Module[{u = $u, nRow, nCol, i, j},

    {nRow, nCol} = Dimensions[u];
    If[northBCtype == "Dirichlet",
     u[[1, 2 ;; nCol - 1]] = northBCconstantValue * Table[northbc[grid[[1, j, 1]]], {j, 2, nCol - 1}]
    ];

    If[westBCtype == "Dirichlet",
     u[[2 ;; nRow - 1, 1]] = westBCconstantValue * Table[westbc[grid[[i, 1, 2]]], {i, 2, nRow - 1}]
    ];

    If[eastBCtype == "Dirichlet",
     u[[2 ;; nRow - 1, nCol]] = eastBCconstantValue * Table[eastbc[grid[[i, nCol, 2]]], {i, 2, nRow - 1}]
    ];

    If[southBCtype == "Dirichlet",
     u[[nRow, 2 ;; nCol - 1]] = southBCconstantValue * Table[southbc[grid[[nRow, j, 1]]], {j, 2, nCol - 1}]
    ];

    (*corner points *)
    u = setCornerNodeCommon[u, 1, 1, nCol, nRow];
    u = setCornerNodeCommon[u, 1, nCol, nCol, nRow];
    u = setCornerNodeCommon[u, nRow, 1, nCol, nRow];
    u = setCornerNodeCommon[u, nRow, nCol, nCol, nRow]
   ];
(*----------------------------------------*)
makeFinalPlot[u_, AA_, Lx_, Ly_, grid_, plotToShow_, plotPerformanceGoal_, addFaceGrids_, zAxisScale_] :=
  Module[{tmp, finalDisplayImage, nRow, nCol, plot, n, dim, cond},
   {nRow, nCol} = Dimensions[u];

   tmp = MapThread[Append[#1, #2] &, {grid, u}, 2];
   tmp = Chop@Flatten[tmp, 1];

   If[plotToShow == "solution and density" || plotToShow == "solution and contour",
    plot = Item@ListPlot3D[tmp,
       PerformanceGoal → plotPerformanceGoal,
       ImagePadding → {{20, 15}, {15, 1}},
       PlotRange → All,
       If[zAxisScale == True, BoxRatios → {Lx, Ly, Min[{Lx, Ly}]}, {}],
       AxesLabel → {Text@Style["x", Italic, 12], Text@Style["y", Italic, 12], None},
       TicksStyle → 9,
       SphericalRegion → True,
       If[addFaceGrids, FaceGrids → All, FaceGrids → None],
       ImageSize → {ContentSizeW, (ContentSizeH - 10) / 2 }
      ]
   ];

   Which[
    plotToShow == "solution and density",
    finalDisplayImage = Grid[{
       {plot},
       {Item@ListDensityPlot[tmp,
          PlotRange → All,
          ImageSize → {ContentSizeW, (ContentSizeH - 10) / 2},
          ImagePadding → {{20, 15}, {20, 1}}, PerformanceGoal → plotPerformanceGoal
         ]
       }
      }, Spacings → {0, 0}
     ]
    ,
    plotToShow == "solution and contour",
    finalDisplayImage = Grid[{
       {Item@plot},
       {
```

```
          Item@ListContourPlot[tmp, Contours → 10,
            ImageSize → {ContentSizeW, (ContentSizeH - 10) / 2},
            ImagePadding → {{20, 15}, {20, 1}}, PerformanceGoal → plotPerformanceGoal]
          }
         }, Spacings → {0, 0}
        ]
       ,
       plotToShow == "solution data",
       finalDisplayImage = makeScrolledPane[Normal@u, ContentSizeH - 40, ContentSizeW]
       ,
       plotToShow == "system matrix information",
       (
        (*if size of A is too large, display part of it only*)
        cond = LUDecomposition[AA][[3]];
        dim = Dimensions[AA];
        n = Min[30, First@dim];

        finalDisplayImage = Grid[{
           {Style[Text@Row[{"condition number = ", cond}], 12]},
           {Style[Text@Row[{"matrix size = ", dim}], 12]},
           {Style[Text["eigenvalues"], 12]},
           {makeScrolledPane[Transpose@Partition[Eigenvalues[Normal@AA, n], 1], 45, ContentSizeW - 20]},
           {Style[Text["A matrix"], 12]},
           {makeScrolledPane[Normal@AA[[1 ;; n, 1 ;; n]], ContentSizeH - 140, ContentSizeW]}
          }]
       )
      ];

      finalDisplayImage
     ];
(*----------------------------------------*)
obtainListOfRowsToKeep[nRow_, nCol_, northBCtype_, southBCtype_, westBCtype_, eastBCtype_] :=
  Module[{rowsToRemove = {}},

   If[northBCtype == "Dirichlet",
    AppendTo[rowsToRemove, Range[1, nCol]];
   ];

   If[southBCtype == "Dirichlet",
    AppendTo[rowsToRemove, Range[(nRow - 1) * nCol + 1, nRow * nCol]];
   ];

   If[westBCtype == "Dirichlet",
    AppendTo[rowsToRemove, Range[1, nRow * nCol, nCol]];
   ];

   If[eastBCtype == "Dirichlet",
    AppendTo[rowsToRemove, Range[nCol, nRow * nCol, nCol]];
   ];

   Complement[Range[nRow * nCol], Flatten[rowsToRemove]]
  ];
(*--------------------------------------------------------------*)
(* Thanks to Heike @SO  for this function for making grid line *)
(*--------------------------------------------------------------*)
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 = 260;
ContentSizeH = 405 ;
```

Manipulate

```
Manipulate[

 gtick;

 {finalDisplayImage, u, forceGrid, grid, systemMatrix, rightHandVector} =
  process[u, forceGrid, grid, systemMatrix, rightHandVector, finalDisplayImage, event, h,
   h, lenX, lenY, centerGrid, addFaceGrids, plotPerformanceGoal, kValue, a, b, c, x0, y0,
   stdx, stdy, forceTermSelection, plotToShow, northBCtype, northbc, northBCconstantValue,
   westBCtype, westbc, westBCconstantValue, eastBCtype, eastbc, eastBCconstantValue,
   southBCtype, southbc, southBCconstantValue, zAxisScale, angle, Unevaluated@gstatusMessage];
 FinishDynamic[];
 Framed[finalDisplayImage , FrameStyle -> Directive[Thickness[.005], Gray], ImageMargins → 0],

 Evaluate@With[{
    plotOptions = Item[Grid[{
         {
          PopupMenu[Dynamic[plotToShow, {plotToShow = #; event = "plot_changed" , gtick += del} &],
           {"solution and contour" → Style["solution + contour", 11],
            "solution and density" → Style["solution + density", 11],
            "solution data" → Style["solution data", 11],
            "system matrix information" → Style["system matrix", 11]
           },
           ImageSize -> All, ContinuousAction -> False]
          ,
          Grid[{{RadioButtonBar[Dynamic[plotPerformanceGoal,
                {plotPerformanceGoal = #; event = "plot_changed" , gtick += del} &],
              {"Speed" -> Text@Style["speed", 11], "Quality" -> Text@Style["quality", 11]},
              Appearance → "Vertical"]
            },
            {Style["plot", 11]}
           }],

          Grid[{{Checkbox[Dynamic[addFaceGrids, {addFaceGrids = #; event = "plot_changed" , gtick += del} &]]
            },
            {Style[Column[{"face", "grids"}], 11]}
           }],

          Grid[{{Checkbox[Dynamic[zAxisScale, {zAxisScale = #; event = "plot_changed" , gtick += del} &]]
            },
            {Style[Column[{"zoom", "scale"}], 11]}
           }]
         }
        },
```

```
    ,
    Alignment → Center, Spacings → {.7, .4}, Frame → {All}, FrameStyle -> Directive[Thickness[.005], Gray]
   ], Alignment → {Center, Top}
  ],
(*------------------------*)
(*--- TOP ROW        -----*)
(*------------------------*)
topRow = Item[Grid[{
     {
      Grid[{
        {
         Button[Text[Style["solve", 12]], {event = "run_button"; gtick += del}, ImageSize -> {50, 28}]
        },
        {
         Button[Text[Style["reset", 12]], {event = "reset"; gtick += del}, ImageSize -> {50, 28}]
        }
       }, Spacings → {0, .2}
      ]
      ,
      Grid[{
        {
         Row[{Style["k", Italic, 11], Spacer[3],
           Manipulator[Dynamic[kValue, {kValue = #; event = "k_changed"; gtick += del} &],
            {0.0, 100., 1.}, ImageSize → Tiny, ContinuousAction -> False],
           Spacer[2],
           Text@Style[Dynamic@padIt2[kValue, {3, 0}], 11]
          }],

         Row[{Style["θ", 11], Spacer[3],
           Manipulator[Dynamic[angle, {angle = #; event = "k_changed"; gtick += del} &], {0,
             2 Pi, Pi / 100}, ImageSize → Tiny, ContinuousAction -> False],
           Spacer[2],
           Text@Style[Row[{Dynamic@padIt2[180. / Pi * angle, {5, 2}], Degree}], 11]
          }]
        },
        {
         Row[{Text@Style["test case", 12], Spacer[2],
           PopupMenu[Dynamic[testCase, {testCase = #;
                Which[testCase == 1,
                 (
                  angle = 0.; zAxisScale = False; centerGrid = True; addFaceGrids = False;
                  plotPerformanceGoal = "Quality"; h = 1 / 4; lenX = 1; lenY = 1; kValue = 5;
                  a = 1.; b = 0.; c = 0.; x0 = 0.; y0 = 0.; stdx = 0.3; stdy = 0.3;
                  forceTermSelection = 1; plotToShow = "solution and contour";
                  northBCtype = "Sommerfeld"; northbc = (1) &; northBCconstantValue = 0.;
                  westBCtype = "Dirichlet"; westbc = (1) &; westBCconstantValue = 0;
                  eastBCtype = "Dirichlet"; eastbc = (1) &; eastBCconstantValue = 0;
                  southBCtype = "Sommerfeld"; southbc = (1) &; southBCconstantValue = 0
                 ), testCase == 2,
                 (
                  angle = 0.; zAxisScale = False; centerGrid = True; addFaceGrids = True;
                  plotPerformanceGoal = "Quality"; h = 1 / 12; lenX = 2; lenY = 2; kValue = 16;
                  a = 1.; y0 = 0.; x0 = 0.; stdx = 0.3; stdy = 0.3; forceTermSelection = 3;
                  plotToShow = "solution and contour"; northBCtype = "Sommerfeld";
                  northbc = (1) &; northBCconstantValue = 0.; westBCtype = "Dirichlet";
                  westbc = (1) &; westBCconstantValue = 0; eastBCtype = "Dirichlet";
                  eastbc = (1) &; eastBCconstantValue = 0; southBCtype = "Sommerfeld";
                  southbc = (1) &; southBCconstantValue = 0
                 ), testCase == 3,
                 (
                  angle = 109 Degree; zAxisScale = False; centerGrid = True;
                  addFaceGrids = True; plotPerformanceGoal = "Quality"; h = 1 / 12; lenX = 2;
```

```
                    lenY = 2; kValue = 52; a = 1.; y0 = 0.; x0 = 0.; stdx = 0.3; stdy = 0.3;
                    forceTermSelection = 3; plotToShow = "solution and contour";
                    northBCtype = "Sommerfeld"; northbc = (1) &; northBCconstantValue = 0.;
                    westBCtype = "Dirichlet"; westbc = (1) &; westBCconstantValue = 0;
                    eastBCtype = "Dirichlet"; eastbc = (1) &; eastBCconstantValue = 0;
                    southBCtype = "Sommerfeld"; southbc = (1) &; southBCconstantValue = 0
                   ), testCase == 4,
                   (
                    angle = 208 Degree; zAxisScale = False;
                    centerGrid = True; addFaceGrids = False; plotPerformanceGoal = "Quality";
                    h = 1 / 16; lenX = 2; lenY = 2; kValue = 7; a = 1.; y0 = 0.; x0 = 0.; stdx = 0.3;
                    stdy = 0.3; forceTermSelection = 3; plotToShow = "solution and contour";
                    northBCtype = "Dirichlet"; northbc = (Cos[2 Pi #]) &;
                    northBCconstantValue = 1.; westBCtype = "Sommerfeld"; eastBCtype = "Sommerfeld";
                    southBCtype = "Dirichlet"; southbc = (Cos[Pi #]) &; southBCconstantValue = 1.
                   ), testCase == 5,
                   (
                    angle = 208 Degree; zAxisScale = False; centerGrid = True;
                    addFaceGrids = False; plotPerformanceGoal = "Quality"; h = 1 / 16;
                    lenX = 2; lenY = 2; kValue = 7; a = 1.; y0 = 0.; x0 = 0.; stdx = 0.3;
                    stdy = 0.3; forceTermSelection = 3; plotToShow = "solution and contour";
                    northBCtype = "Sommerfeld"; eastBCtype = "Sommerfeld";
                    westBCtype = "Dirichlet"; westbc = (Cos[2 Pi #]) &; westBCconstantValue = 1.;
                    southBCtype = "Dirichlet"; southbc = (Cos[Pi #]) &; southBCconstantValue = 1.
                   )
                  ]; event = "reset"; gtick += del} &],
              {1 → Style["1", 11], 2 → Style["2", 11], 3 → Style["3", 11], 4 → Style["4", 11],
               5 → Style["5", 11]}, ImageSize -> All, ContinuousAction -> False]
            }],
           Text@
            Style[Row[{"-", ("∇")^2, Style["u", Italic], " - ", Style["k", Italic]^2, Style["u", Italic],
               " = ", Style["f", Italic], "(", Style["x", Italic], ", ", Style["y", Italic], ")"}], 12]
          }}, Alignment → Center, Spacings → {.5, .4}, Frame → {None, All},
         FrameStyle -> Directive[Thickness[.005], Gray]
        ]
       }}, Alignment → Center, Spacings → {.2, 0}
    ], Alignment → {Center, Top}
   ],

(*--------------------------*)
(*--- geometry            --*)
(*--------------------------*)
geometryTerm = Item[Grid[{
     {
      Grid[{
         {
          Grid[{{
              Text@Style["grid", 12],
              SetterBar[Dynamic[h, {h = #; event = "reset"; gtick += del} &],
               # → Style[#, 11] & /@ {1 / 4, 1 / 6, 1 / 8, 1 / 10, 1 / 12, 1 / 14, 1 / 16}],
              Spacer[1],
              Style["center grid ", 12],
              Checkbox[Dynamic[centerGrid, {centerGrid = #; event = "reset"; gtick += del} &]]
             }}, Spacings → {.2, .5}]
         }
         ,
         {
          Grid[{
             {
              Text@Style[Row[{Style["x", Italic], " length"}], 12],
              SetterBar[Dynamic[lenX, {lenX = #; event = "reset"; gtick += del} &], {1, 2}],
              Spacer[95],
              Text@Style[Row[{Style["y", Italic], " length"}], 12],
```

```
        SetterBar[Dynamic[lenY, {lenY = #; event = "reset"; gtick += del} &], {1, 2}]
       }
      }, Spacings → {.6, .1}
     ]
    }
   }, Spacings → {0, .5}
 ], SpanFromLeft
}
,
{
 Grid[{
   {
    RadioButtonBar[Dynamic[northBCtype,
      {northBCtype = #;
        If[northBCtype == "Sommerfeld" && southBCtype == "Sommerfeld" &&
          westBCtype == "Sommerfeld" && eastBCtype == "Sommerfeld",
         (
          northBCtype = "Dirichlet"
         ),
         (
          event = "reset";
          gtick += del
         )
        ]} &],
     {"Dirichlet" → Text@Style["Dirichlet", 10],
      "Sommerfeld" → Text@Style["Sommerfeld", 10]}, Appearance → "Vertical"
    ], SpanFromLeft
   }
   ,
   {
    PopupMenu[Dynamic[northbc, {northbc = #; event = "reset"; gtick += del} &],
     {
      (1.0) & → Style["a", Italic, 11],
      (#) & → Style["a x", Italic, 11],
      (#^2) & → Style[Row[{Style["a ", Italic], Style["x", Italic]^2}], 11],
      (Cos[Pi #]) & → Style[Row[{Style["a", Italic], " cos(π ", Style["x", Italic], ")"}], 11],
      (Cos[2 Pi #]) & → Style[Row[{Style["a", Italic], " cos(2 π ", Style["x", Italic], ")"}], 11]
     },
     ImageSize -> All, ContinuousAction -> False,
     Enabled → Dynamic[northBCtype == "Dirichlet"]], SpanFromLeft
   },
   {
    Grid[{
      {
       Text@Style["a", Italic, 12],
       Manipulator[Dynamic[northBCconstantValue,
         {northBCconstantValue = #; event = "reset"; gtick += del} &], {-20, 20, 0.1}, ImageSize ->
         Tiny, ContinuousAction -> False, Enabled → Dynamic[northBCtype == "Dirichlet"]],
       Text@Style[Dynamic@padIt1[northBCconstantValue, {3, 1}], 10],
       Button[Text@Style["zero", 11], {northBCconstantValue = 0.0; event = "reset"; gtick += del},
        ImageSize → {45, 20}, Enabled → Dynamic[northBCtype == "Dirichlet"]],
       Button[Text@Style["one", 11], {northBCconstantValue = 1.0; event = "reset"; gtick += del},
        ImageSize → {45, 20}, Enabled → Dynamic[northBCtype == "Dirichlet"]]
      }
     }, Spacings → {.2, 0}, Alignment → Center, Frame → None], SpanFromLeft
   }
  }, Frame → None, Spacings → {0.1, 0}, Alignment → Center
 ], SpanFromLeft
}
,
{
 Grid[{
```

```
  --
 {
  RadioButtonBar[Dynamic[westBCtype, {westBCtype = #;
       If[northBCtype == "Sommerfeld" && southBCtype == "Sommerfeld" &&
         westBCtype == "Sommerfeld" && eastBCtype == "Sommerfeld",
        (
         westBCtype = "Dirichlet"
        ),
        (
         event = "reset";
         gtick += del
        )
       ]} &],
    {"Dirichlet" → Text@Style["Dirichlet", 10],
     "Sommerfeld" → Text@Style["Sommerfeld", 10]}, Appearance → "Vertical"]
 },
 {
  PopupMenu[Dynamic[westbc, {westbc = #; event = "reset"; gtick += del} &],
   {
    (1.0) & → Style["a", Italic, 11],
    (#) & → Style["a y", Italic, 11],
    (#^2) & → Style[Row[{Style["a ", Italic], Style["y", Italic]^2}], 11],
    (Cos[Pi #]) & → Style[Row[{Style["a", Italic], " cos(π ", Style["y", Italic], ")"}], 11],
    (Cos[2 Pi #]) & → Style[Row[{Style["a", Italic], " cos(2 π ", Style["y", Italic], ")"}], 11]
   },
   ImageSize -> All, ContinuousAction -> False, Enabled → Dynamic[westBCtype == "Dirichlet"]]
 },
 {Grid[{
    {
     Grid[{

       {Text@Style["a", Italic, 11],
        Manipulator[Dynamic[westBCconstantValue,
          {westBCconstantValue = #; event = "reset"; gtick += del} &], {-20, 20, 0.1}, ImageSize ->
          Tiny, ContinuousAction -> False, Enabled → Dynamic[westBCtype == "Dirichlet"]
        ],
        Text@Style[Dynamic@padIt1[westBCconstantValue, {3, 1}], 10]
       },
       {
        Row[{
          Button[Text@Style["zero", 11], {westBCconstantValue = 0.0; event = "reset"; gtick += del},
           ImageSize → {45, 20}, Enabled → Dynamic[westBCtype == "Dirichlet"]]
          , Spacer[2],
          Button[Text@Style["one", 11], {westBCconstantValue = 1.0; event = "reset"; gtick += del},
           ImageSize → {45, 20}, Enabled → Dynamic[westBCtype == "Dirichlet"]]
         }], SpanFromLeft
        }
       }, Spacings → {.1, 0}, Alignment → Center]
    }
   }, Alignment → Center, Spacings → {0, 0}
  ]
 }
}
]
,
Grid[{
  {
   RadioButtonBar[Dynamic[eastBCtype, {eastBCtype = #;
       If[northBCtype == "Sommerfeld" && southBCtype == "Sommerfeld" &&
         westBCtype == "Sommerfeld" && eastBCtype == "Sommerfeld",
        (
         eastBCtype = "Dirichlet"
        ),
        (
```

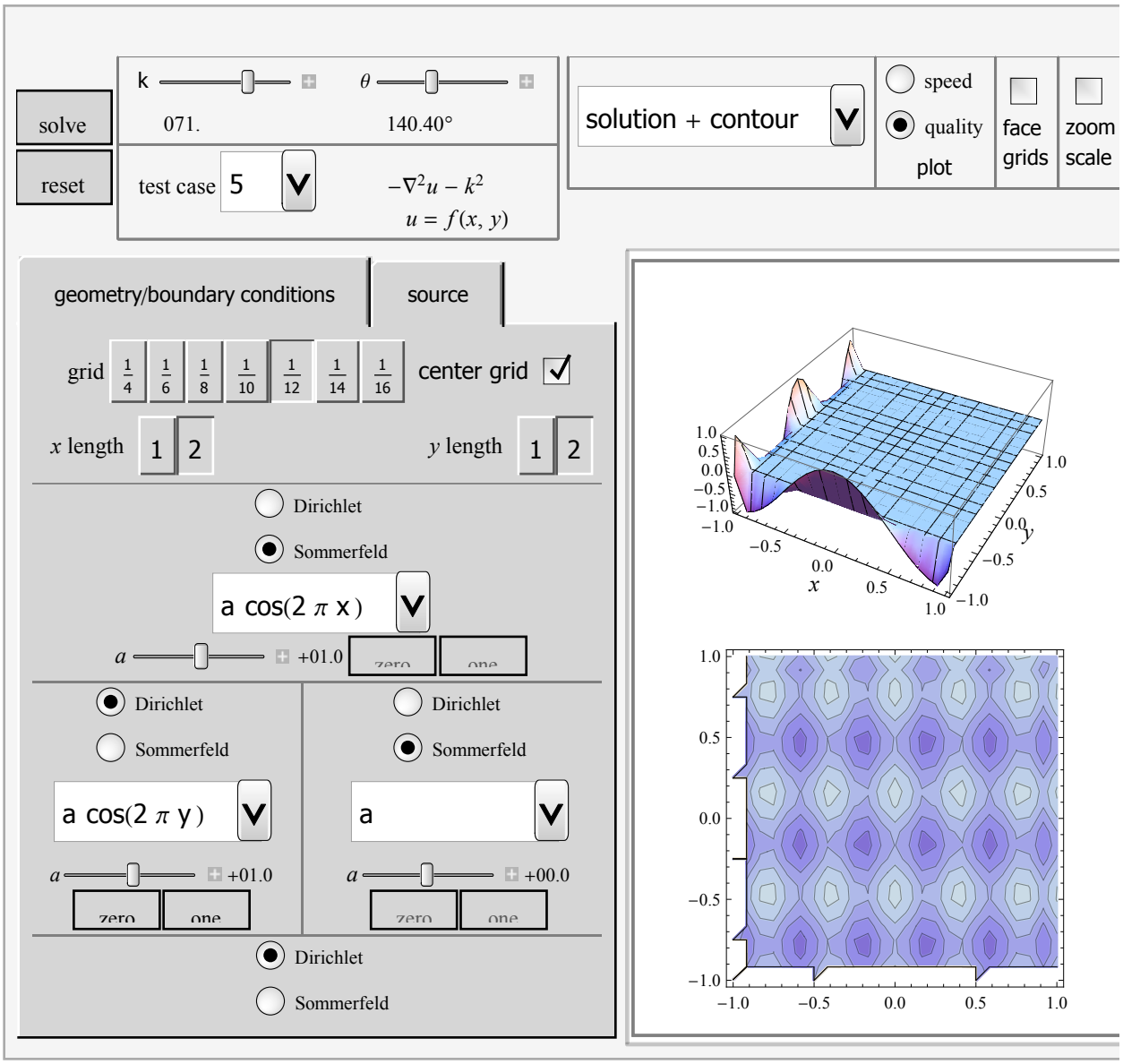
```
          event = "reset";
          gtick += del
         )
        ]} &],
    {"Dirichlet" → Text@Style["Dirichlet", 10],
     "Sommerfeld" → Text@Style["Sommerfeld", 10]}, Appearance → "Vertical"], SpanFromLeft
  },
  {
   PopupMenu[Dynamic[eastbc, {eastbc = #; event = "reset"; gtick += del} &],
    {
     (1.0) & → Style["a", Italic, 11],
     (#) & → Style["a y", Italic, 11],
     (#^2) & → Style[Row[{Style["a ", Italic], Style["y", Italic]^2}], 11],
     (Cos[Pi #]) & → Style[Row[{Style["a", Italic], " cos(π ", Style["y", Italic], ")"}], 11],
     (Cos[2 Pi #]) & → Style[Row[{Style["a", Italic], " cos(2 π ", Style["y", Italic], ")"}], 11]
    },
    ImageSize -> All, ContinuousAction -> False,
    Enabled → Dynamic[eastBCtype == "Dirichlet"]], SpanFromLeft
  },
  {Grid[{
     {
      Grid[{
        {
         Text@Style["a", Italic, 12],
         Manipulator[Dynamic[eastBCconstantValue,
           {eastBCconstantValue = #; event = "reset"; gtick += del} &], {-20, 20, 0.1}, ImageSize ->
           Tiny, ContinuousAction -> False, Enabled → Dynamic[eastBCtype == "Dirichlet"]
         ],
         Text@Style[Dynamic@padIt1[eastBCconstantValue, {3, 1}], 10]
        },
        {
         Row[{Button[Text@Style["zero", 11], {eastBCconstantValue = 0.0; event = "reset";
              gtick += del}, ImageSize → {45, 20}, Enabled → Dynamic[eastBCtype == "Dirichlet"]]
           , Spacer[2],
           Button[Text@Style["one", 11], {eastBCconstantValue = 1.0; event = "reset"; gtick += del},
            ImageSize → {45, 20}, Enabled → Dynamic[eastBCtype == "Dirichlet"]]
          }],
         SpanFromLeft
        }
       }, Spacings → {.1, 0}, Alignment → Center]
     }
    }, Alignment → Center, Spacings → {0, 0}
   ],
   SpanFromLeft
  }
 }]
}
,
{
 Grid[{
   {
    RadioButtonBar[Dynamic[southBCtype, {southBCtype = #;
         If[northBCtype == "Sommerfeld" && southBCtype == "Sommerfeld" &&
           westBCtype == "Sommerfeld" && eastBCtype == "Sommerfeld",
          (
           southBCtype = "Dirichlet"
          ),
          (
           event = "reset";
           gtick += del
          )
         ]} &],
```

```
        {"Dirichlet" → Text@Style["Dirichlet", 10],
         "Sommerfeld" → Text@Style["Sommerfeld", 10]}, Appearance → "Vertical"]
      }
      ,
      {
       PopupMenu[Dynamic[southbc, {southbc = #; event = "reset"; gtick += del} &],
        {
         (1.0) & → Style["a", Italic, 11],
         (#) & → Style["a x", Italic, 11],
         (#^2) & → Style[Row[{Style["a ", Italic], Style["x", Italic]^2}], 11],
         (Cos[Pi #]) & → Style[Row[{Style["a", Italic], " cos(π ", Style["x", Italic], ")"}], 11],
         (Cos[2 Pi #]) & → Style[Row[{Style["a", Italic], " cos(2 π ", Style["x", Italic], ")"}], 11]
        },
        ImageSize -> All, ContinuousAction -> False, Enabled → Dynamic[southBCtype == "Dirichlet"]]
      },
      {Grid[{
         {Text@Style["a", Italic, 12],
          Manipulator[Dynamic[southBCconstantValue,
            {southBCconstantValue = #; event = "reset"; gtick += del} &], {-20, 20, 0.1}, ImageSize ->
            Tiny, ContinuousAction -> False, Enabled → Dynamic[southBCtype == "Dirichlet"]],
          Text@Style[Dynamic@padIt1[southBCconstantValue, {3, 1}], 10],
          Button[Text@Style["zero", 11], {southBCconstantValue = 0.0; event = "reset"; gtick += del},
           ImageSize → {45, 20}, Enabled → Dynamic[southBCtype == "Dirichlet"]],
          Button[Text@Style["one", 11], {southBCconstantValue = 1.0; event = "reset"; gtick += del},
           ImageSize → {45, 20}, Enabled → Dynamic[southBCtype == "Dirichlet"]]
         }
        }, Spacings → {.2, 0}
       ]
      }
     }
    ], SpanFromLeft
   }
  }, Spacings → {2, .3}, Alignment → Center, Frame → All,
  FrameStyle -> Directive[Thickness[.005], Gray]], Alignment → {Center, Top}
 ],
(*----------------*)
(*--- source    --*)
(*----------------*)
sourceTerm = Item[Grid[{
   {
    Item[
     PopupMenu[Dynamic[forceTermSelection, {forceTermSelection = #; event = "reset"; gtick += del} &],
      {1 → Style["a", Italic, 12],
       3 → Style[Row[{Style["a", Italic], " exp (",
            Row[{Style["x", Italic], " - ", (Style["x", Italic]"0")^2}]
            / ("2 σ"Style["x",Italic])^2
            , " + ", Row[{Style["y", Italic],
              " - ", (Style["y", Italic]"0")^2}] / ("2 σ"Style["y",Italic])^2, ")"}], 12],
       4 → Style[Row[{Style["a", Italic], " ( cos( ", Style["b", Italic], " π ", Style["x", Italic],
           " ) + sin( ", Style["c", Italic], " π ", Style["y", Italic], " ) )"}], 12],
       5 → Style[Row[{Style["a", Italic], " cos( ", Style["b", Italic], " π ", Style["x", Italic],
           " ) * sin( ", Style["c", Italic], " π ", Style["y", Italic], " )"}], 12]
      }, ImageSize → {ContentSizeW, ContentSizeH - 365}, ContinuousAction -> False],
     Alignment → {Center}
    ], SpanFromLeft
   },
   {
```

```
 Spacer[2],
 Text@Style["a", Italic, 12],
 Manipulator[Dynamic[a, {a = #; event = "reset"; 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; event = "reset"; gtick += del}, ImageSize → {45, 20}],
 Button[Text@Style["one", 10], {a = 1; event = "reset"; gtick += del}, ImageSize → {45, 20}]
},
{
 Spacer[2],
 Text@Style["b", Italic, 12],
 Manipulator[Dynamic[b, {b = #; event = "reset"; gtick += del} &],
  {-10., 10., 0.1}, ImageSize → Small, ContinuousAction -> False,
  Enabled → Dynamic[forceTermSelection == 2 || forceTermSelection == 4 || forceTermSelection == 5]],
 Text@Style[Dynamic@padIt1[b, {3, 1}], 11],
 Button[Text@Style["zero", 10], {b = 0.0; event = "reset"; gtick += del}, ImageSize → {45, 20}],
 Button[Text@Style["one", 10], {b = 1.0; event = "reset"; gtick += del}, ImageSize → {45, 20}]
},
{
 Spacer[2],
 Text@Style["c", Italic, 12],
 Manipulator[Dynamic[c, {c = #; event = "reset"; gtick += del} &], {-10., 10., 0.1}, ImageSize → Small,
  ContinuousAction -> False, Enabled → Dynamic[forceTermSelection == 4 || forceTermSelection == 5]],
 Text@Style[Dynamic@padIt1[c, {3, 1}], 11],
 Button[Text@Style["zero", 10], {c = 0.0; event = "reset"; gtick += del}, ImageSize → {45, 20}],
 Button[Text@Style["one", 10], {c = 1.0; event = "reset"; gtick += del}, ImageSize → {45, 20}]
},
{
 Spacer[2],
 Text@Style[Row[{Style["x", Italic]"0"}], 12],
 Manipulator[Dynamic[x0, {x0 = #; event = "reset"; 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; event = "reset"; gtick += del}, ImageSize → {45, 20}],
 Button[Text@Style["one", 10], {x0 = 1.0; event = "reset"; gtick += del}, ImageSize → {45, 20}]

},
{
 Spacer[2],
 Text@Style[Row[{Style["y", Italic]"0"}], 12],
 Manipulator[Dynamic[y0, {y0 = #; event = "reset"; gtick += del} &], {-1.5, 1.5, 0.01},
  ImageSize → Small, ContinuousAction -> False, Enabled → Dynamic[forceTermSelection == 3]],
 Text@Style[Dynamic@padIt1[y0, {3, 2}], 11],
 Button[Text@Style["zero", 10], {y0 = 0.0; event = "reset"; gtick += del}, ImageSize → {45, 20}],
 Button[Text@Style["one", 10], {y0 = 1.0; event = "reset"; gtick += del}, ImageSize → {45, 20}]
}
,
{
 Item[
  Row[{
    Text@Style[Row[{Style["σ"]Style["x",Italic]}], 11],
    Manipulator[Dynamic[stdx, {stdx = #; event = "reset"; gtick += del} &], {0.1, 3, 0.05},
     ImageSize → Tiny, ContinuousAction -> False, Enabled → Dynamic[forceTermSelection == 3]],
    Text@Style[Dynamic@padIt1[stdx, {3, 2}], 11],
    Spacer[20],
    Text@Style[Row[{Style["σ"]Style["y",Italic]}], 11],
    Manipulator[Dynamic[stdy, {stdy = #; event = "reset"; gtick += del} &], {0.1, 3, 0.05},
     ImageSize → Tiny, ContinuousAction -> False, Enabled → Dynamic[forceTermSelection == 3]],
    Text@Style[Dynamic@padIt1[stdy, {3, 2}], 11]
   }], Alignment → Center
 ], SpanFromLeft
}
,
```

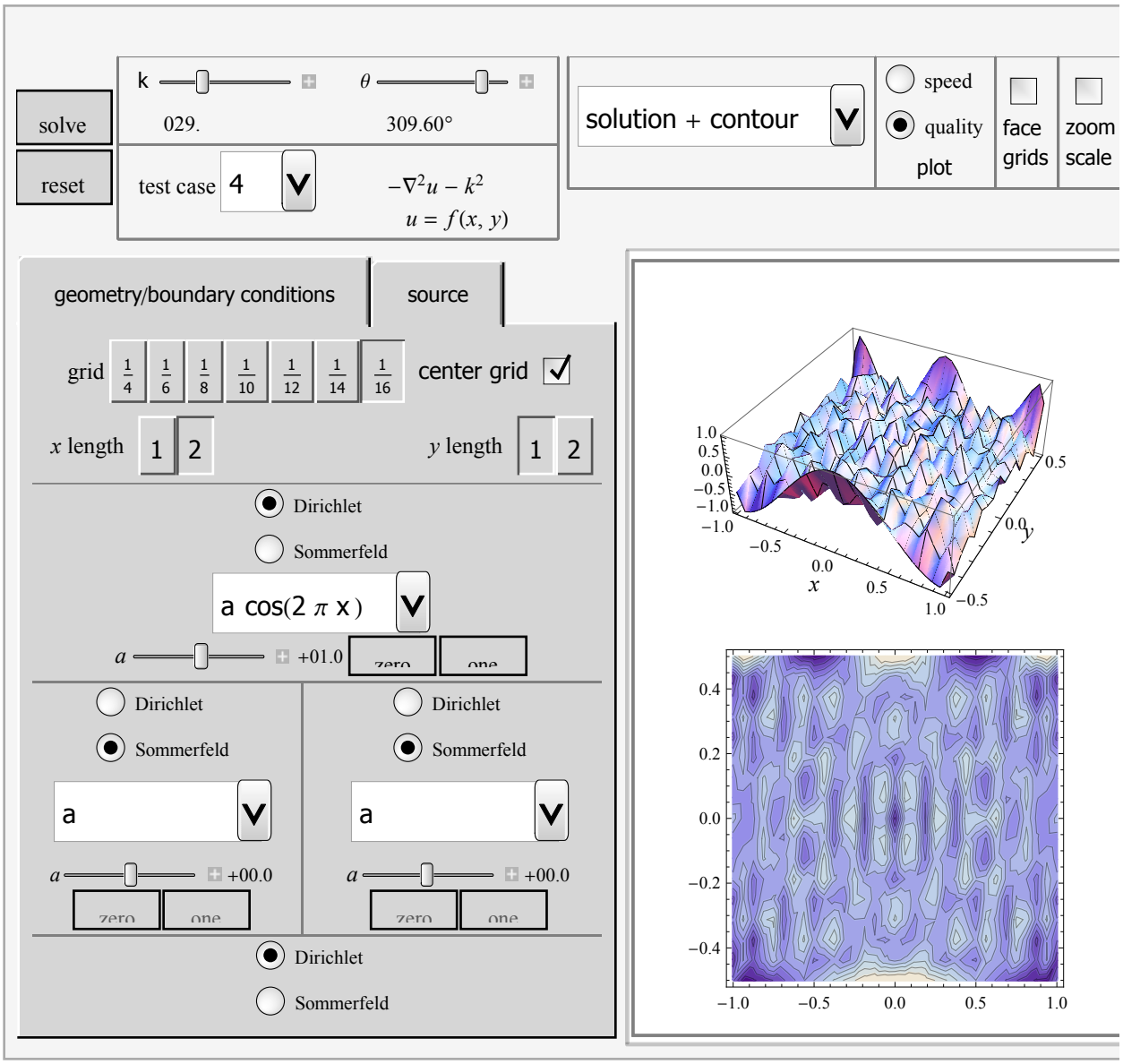
```
        {
         Dynamic@Grid[{
             {Plot3D[
               Evaluate@forceTermExpressionCommon[forceTermSelection, a, b, c, stdy, stdx, x0, y0, x, y],
               Evaluate@{x, grid[[-1, 1, 1]], grid[[-1, -1, 1]]}, {y, grid[[-1, 1, 2]], grid[[1, 1, 2]]},
               PerformanceGoal → plotPerformanceGoal,
               ImagePadding → {{10, 10}, {20, 25}},
               ImageMargins → 1,
               PlotRange → All,
               PlotLabel → Text@Style[Row[{Style["f", Italic], "(", Style["x", Italic],
                     ", ", Style["y", Italic], ") = ", forceTermUsedFormatCommon[
                      forceTermSelection, a, b, c, stdy, stdx, x0, y0, x, y]}], 11],
               AxesLabel → {Text@Style["x", Italic, 11], Text@Style["y", Italic, 11], None},
               ImageSize → {ContentSizeW + 30, ContentSizeH - 240},
               TicksStyle → 9
              ]}, Spacings → {0, 0}, Frame → None], SpanFromLeft
         }
        }, Spacings → {.3, .2}, Alignment → Left,
        Frame → {None, All}, FrameStyle -> Directive[Thickness[.005], Gray]
       ], Alignment → {Center, Top}
      ]
     },
     (*-----------------*)
     (*--- LEVEL 2 -----*)
     (*-----------------*)
     With[{
       pde2 = Grid[{
           {TabView[{
              Style["geometry/boundary conditions", 11] → geometryTerm,
              Style["source", 11] → sourceTerm
             }, ImageSize → {315, 410 }]
           }
         }, Spacings → {0.1, .0}, Alignment → Center
        ]
      },
      (*--- end of level 2 ---*)

      ## &[
       Item[

        Grid[{
          {topRow, plotOptions}
         }, Spacings → {.2, 0}, Alignment → {Center, Top}
        ], ControlPlacement → Top
       ],
       Item[pde2, ControlPlacement → Left]
      ]
     ]
    ],
   (*----------- end of Manipulate controls ----------------------------*)

   {{gstatusMessage, "reseting..."}, None},
   {{gtick, 0}, None},
   {{del, $MachineEpsilon}, None},

   {{testCase, 1}, None},
   {{angle, 0.}, None},
   {{systemMatrix, {}}, None},
   {{rightHandVector, {}}, None},
   {{zAxisScale, False}, None},
   {{centerGrid, True}, None},
```

```
 {{event, "run_button"}, None},
 {{finalDisplayImage, {}}, None},
 {{addFaceGrids, False}, None},
 {{plotPerformanceGoal, "Quality"}, None},
 {{h, 1 / 4}, None},
 {{lenX, 1}, None},
 {{lenY, 1}, None},
 {{kValue, 5.0}, None},
 {{a, 1.0}, None},
 {{b, 0.0}, None},
 {{c, 0.0}, None},
 {{x0, 0.0}, None},
 {{y0, 0.0}, None},
 {{stdx, 0.3}, None},
 {{stdy, 0.3}, None},
 {{forceTermSelection, 1}, None},
 {{plotToShow, "solution and contour"}, None},
 {{northBCtype, "Sommerfeld"}, None},
 {{northbc, (1) &}, None},
 {{northBCconstantValue, 0}, None},
 {{westBCtype, "Dirichlet"}, None},
 {{westbc, (1) &}, None},
 {{westBCconstantValue, 0}, None},
 {{eastBCtype, "Dirichlet"}, None},
 {{eastbc, (1) &}, None},
 {{eastBCconstantValue, 0}, None},
 {{southBCtype, "Sommerfeld"}, None},
 {{southbc, (1) &}, None},
 {{southBCconstantValue, 0}, None},
 {{forceGrid, Table[-1, {5}, {5}]}, None},
 {{grid, makeGridCommon[0.25, 0.25, 1, 1, True]}, None},
 {{u, {}}, None},

 ControlPlacement → Left,
 SynchronousUpdating → False,
 ContinuousAction → False,
 SynchronousInitialization → True,
 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 partial differential equation $-\nabla^2 u - k^2 u = f(x, y)$ on a rectangle with uniform grid spacing. Dirichlet and Sommerfeld boundary conditions are supported. You can specify different source functions $f(x, y)$. You ca nprescribe Sommerfeld boundary conditions on up to three edges of the rectangle at the same time. You can vary the k value and the angle of incident θ. The numerical scheme is converted to standard $A u = b$ system and solved. You can view the generated matrix A and its eigenvalues as well as the solution data 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 u can be viewed using the pull down menu. The discretized scheme is given by $\frac{1}{h^2}\left(-u_{i-1,j} + u_{i+1,j} - \omega\, u_{i,j} + u_{i,j-1} + u_{i,j+1} - k^2\, u_{i,j}\right) = f_{i,j}$ where $\omega = (k\,h)^2 + 4\,J_0(h\,k)$ and $J_0(k\,h) = \frac{1}{\pi}\int_0^{\pi} \cos(k\,h\,\sin(\theta))\,d\theta$.

Click the solve button after making changes to the UI variables to get a new solution. The reset button is used to initialize the system back to the state it was before the solve button was clicked. Different types of plots and options are available to choose from.

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*, **2**(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
- [x] Gamepad Controls
- [x] Automatic Animation
- [ ] Bookmark Animation

Search Terms (optional)

Helmholtz partial differential equation
finite difference method

Related Links (optional)

Heat Conduction Equation

Authoring Information

Contributed by: Nasser M. Abbasi