Finite Difference Solution of the Diffusion-Advection-Reaction Equation in 1D

Initialization Code (optional)

```
(*------------------------------------------------------*)
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}]]];

(*------------------------------------------------------*)
makeScrolledPane[mat_?(MatrixQ[#, NumberQ] &),
    nRow_?(IntegerQ[#] && Positive[#] &), nCol_?(IntegerQ[#] && Positive[#] &)] :=
  Module[{t}, t = Grid[mat, Spacings → {.4, .4}, Alignment → Left, Frame → All];
    t = 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 = Style[AccountingForm[Chop[N@t], {6, 5}, NumberSigns → {"-", ""},
        NumberPadding → {"", ""}, SignPadding → True], LineBreakWithin → False];
    Pane[t, ImageSize → {nCol, nRow}, Scrollbars → True]];

(*------------------------------------------------------*)
checkTerm[t_?(NumberQ[#] &)] := If[Abs[t - 1] < $MachineEpsilon, 1, If[Abs[t] < $MachineEpsilon, 0, t]];

(*--use simple adaptive method to reduce memory use*)
findOptimalTimeScale[stepNumber_, maxTime_, timeStepDuration_] :=
  Which[stepNumber ≤ 10, Min[maxTime, 10 * timeStepDuration], stepNumber > 10 && stepNumber ≤ 20,
    Min[maxTime, 20 * timeStepDuration], stepNumber > 20 && stepNumber ≤ 50, Min[maxTime, 50 * timeStepDuration],
    stepNumber > 50 && stepNumber ≤ 100, Min[maxTime, 100 * timeStepDuration],
    stepNumber > 100 && stepNumber ≤ 200, Min[maxTime, 200 * timeStepDuration],
    stepNumber > 200 && stepNumber ≤ 300, Min[maxTime, 300 * timeStepDuration],
    stepNumber > 300 && stepNumber ≤ 500, Min[maxTime, 500 * timeStepDuration],
    stepNumber > 500 && stepNumber ≤ 1000, Min[maxTime, 1000 * timeStepDuration],
    stepNumber > 1000 && stepNumber ≤ 2000, Min[maxTime, 2000 * timeStepDuration], True, maxTime];
(*------------------------------------------------*)
updateAnimationBuffer[stepNumber_, animation3dBuffer_, grid_, currentTime_, u_] := Module[{n, m, i},
    Which[stepNumber == 0,
     animation3dBuffer[[1]] = 1;
     animation3dBuffer[[2]][[1]] = Table[{grid[[i]], currentTime, u[[i]]}, {i, Length[grid]}];
     n = 1,

     stepNumber ≤ 10,
     animation3dBuffer[[1]] = animation3dBuffer[[1]] + 1;
     n = animation3dBuffer[[1]];
     animation3dBuffer[[2]][[n]] = Table[{grid[[i]], currentTime, u[[i]]}, {i, Length[grid]}],
```

```
     stepNumber > 10 && stepNumber ≤ 20,
     If[Mod[stepNumber, 2] == 0,
      animation3dBuffer[[1]] = animation3dBuffer[[1]] + 1;
      n = animation3dBuffer[[1]];
      animation3dBuffer[[2]][[n]] = Table[{grid[[i]], currentTime, u[[i]]}, {i, Length[grid]}]
      ,
      n = animation3dBuffer[[1]]
     ],

     stepNumber > 20 && stepNumber ≤ 30,
     If[Mod[stepNumber, 5] == 0,
      animation3dBuffer[[1]] = animation3dBuffer[[1]] + 1;
      n = animation3dBuffer[[1]];
      animation3dBuffer[[2]][[n]] = Table[{grid[[i]], currentTime, u[[i]]}, {i, Length[grid]}]
      ,
      n = animation3dBuffer[[1]]
     ],

     stepNumber > 30,
     If[Mod[stepNumber, 10] == 0,
      animation3dBuffer[[1]] = animation3dBuffer[[1]] + 1;
      n = animation3dBuffer[[1]];
      animation3dBuffer[[2]][[n]] = Table[{grid[[i]], currentTime, u[[i]]}, {i, Length[grid]}]
      ,
      n = animation3dBuffer[[1]]]
    ];

    If[n == 1,
     animation3dBuffer[[2]][[2]] = animation3dBuffer[[2]][[1]];
     m = 2
     ,
     m = n
    ];

    m
   ];
(*-----------------------------------------------------------*)
makeTopTitleForPlot[currentTime_, timeStepDuration_] := Grid[{{Text@Style["time", 11],
      Spacer[5], Style[padIt2[currentTime, {9, 8}], 11], Spacer[1], Text@Style["  sec", 11]},
     {Text@Row[{"Δ", Style["t", Italic]}], Spacer[5], Style[padIt2[timeStepDuration, {9, 8}], 11],
      Spacer[1], Text@Style["  sec", 11]}}, Spacings → {.3, 0}, Alignment → Left];
(*----------------------------------------*)
(*Thanks to Heike@SO for this function*)
(*----------------------------------------*)
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]]]];
(*-------------------------------------------------------------------------------*)
processPDE4[$u_, $grid_, $Aleft_, $Aright_, $stepNumber_, $cpuTimeUsed_, $currentTime_,
```

```
 $state_, $u0_, animation3dBuffer_, initialConditionFunction_, event_, h_, centerGrid_,
 length_, k_, aReactionTerm_, dAdvectionTerm_, cDiffusionTerm_, maxTime_, isPeriodicBC_,
 westBCtype_, eastBCtype_, westBoundaryFunction_, eastBoundaryFunction_, sourceFunction_,
 addGrid_, showIC_, joinedType_, sourceDurationType_, sourceDuration_, yscaleAuto_,
 yscaleAmount_, threeDView_, threeDviewSpeed_, gtick_, delta_, gstatusMessage_] :=
Module[{u = $u, u0 = $u0, grid = $grid, Aleft = $Aleft, Aright = $Aright,
  stepNumber = $stepNumber, cpuTimeUsed = $cpuTimeUsed,
  currentTime = $currentTime, state = $state, finalDisplayImage, pde},

 pde = makePDE4[cDiffusionTerm, dAdvectionTerm, aReactionTerm];
 (*----- INIT STATE -----*)
 Which[state == "INIT",
  (
   {u, grid, cpuTimeUsed, stepNumber,
     Aleft, Aright, currentTime, animation3dBuffer} = initializeSystemPDE4[
     initialConditionFunction, h, centerGrid, length, k, aReactionTerm, dAdvectionTerm,
     cDiffusionTerm, isPeriodicBC, westBCtype, eastBCtype, maxTime];
   u0 = u;

   Which[
    event == "reset", gstatusMessage = "reset complete",
    event == "run_button",
    (
     state = "RUNNING";
     gtick += delta
    ),
    event == "pause_button",
    (
     state = "PAUSE";
     gtick += delta
    ),
    event == "step_button",
    (
     state = "RUNNING";
     gtick += delta
    )
   ];
   gstatusMessage = "initialized"
  ),

  (*----- PAUSE STATE -----*)
  state == "PAUSE",
  (
   gstatusMessage = Row[{"paused [", stepNumber, "]"}];

   Which[
    event == "pause_button", state = "PAUSE",
    event == "reset",
    (
     state = "INIT";
     {u, grid, cpuTimeUsed, stepNumber,
       Aleft, Aright, currentTime, animation3dBuffer} = initializeSystemPDE4[
       initialConditionFunction, h, centerGrid, length, k, aReactionTerm, dAdvectionTerm,
       cDiffusionTerm, isPeriodicBC, westBCtype, eastBCtype, maxTime];

     u0 = u;
      gtick += delta
    ),
    event == "run_button" || event == "step_button",
    (
     state = "RUNNING";
     gtick += delta
```

```
      )
     ]
    ),

    (*----- RUNNING STATE -----*)
    state == "RUNNING",
    (
     Which[event == "step_button" || event == "run_button" || event == "plot_changed",
      (
       If[currentTime < maxTime,
        (
         {u, cpuTimeUsed} = solvePDE4[u, Aleft, Aright, grid, k, h, dAdvectionTerm,
           cDiffusionTerm, westBoundaryFunction, eastBoundaryFunction, sourceFunction,
           currentTime, westBCtype, eastBCtype, sourceDurationType, sourceDuration, isPeriodicBC
          ];

         currentTime = currentTime + k;
         stepNumber = stepNumber + 1;

         (*-- only re-loop if in running state --*)
         If[event == "run_button" || event == "plot_changed",
          (
           gtick += delta;
           gstatusMessage = Row[{"running [", stepNumber, "]"}]
          ),
          (
           gstatusMessage = Row[{"paused [", stepNumber, "]"}];
           state = "PAUSE"
          )
         ];
        ),
        gstatusMessage = Row[{"completed [", stepNumber, "]"}];
       ]
      ),

      event == "reset",
      (
       state = "INIT";
       {u, grid, cpuTimeUsed, stepNumber,
         Aleft, Aright, currentTime, animation3dBuffer} = initializeSystemPDE4[
         initialConditionFunction, h, centerGrid, length, k, aReactionTerm, dAdvectionTerm,
         cDiffusionTerm, isPeriodicBC, westBCtype, eastBCtype, maxTime];

       u0 = u;
        gtick += delta
      ),
      event == "pause_button",
      (
       state = "PAUSE";
       gtick += delta
      )
     ]
    )
   ];
   (* state machine completed, plot the final result *)
   finalDisplayImage =
    makeFinalPlotPDE4[u, grid, currentTime, u0, addGrid, showIC, joinedType, yscaleAuto, yscaleAmount,
     pde, Unevaluated[animation3dBuffer], stepNumber, maxTime, threeDView, k, threeDviewSpeed];

   {finalDisplayImage, u, u0, grid, Aleft, Aright, stepNumber, cpuTimeUsed, currentTime, state}
  ];
(*------------------------------------------------------------------------------*)
```

```
makePDE4[ cDiffusionTerm_, dAdvectionTerm_, aReactionTerm_] :=
  Module[{c, d, a, uxx, ut, u, utTerm, uxxTerm, uTerm},

   c = checkTerm@cDiffusionTerm;
   d = checkTerm@dAdvectionTerm;
   a = checkTerm@aReactionTerm;

   uxx = Style["u", Italic, 11]Row[{Style["x",Italic,11],Style["x",Italic,11]}];
   ut = Style["u", Italic, 11]Style["t",Italic,11];
   u = Style["u", Italic, 11];
   utTerm = If[d == 1, ut, Row[{d, Spacer[1], ut}]];
   uxxTerm = Row[{If[c == 1, "", c], Spacer[1], uxx}];
   uTerm = Which[a == 0, " + ",
     a < 0, Row[{a, Spacer[1], u, " + "}],
     a > 0, Row[{" + ", a, Spacer[1], u, " + "}],
     True, Row[{" + ", Spacer[1], u, " + "}]
    ];

   Text@Row[{Spacer[1], uxxTerm, " = ", utTerm, uTerm,
      Style["f", Italic], "(", Style["x", Italic], ", ", Style["t", Italic], ")"}]
  ];
(*------------------------------------------------------------------------------*)
makeFinalPlotPDE4[u_, grid_, currentTime_, u0_, addGrid_, showIC_, joinedType_, yscaleAuto_, yscaleAmount_,
   pde_, animation3dBuffer_, stepNumber_, maxTime_, threeDView_, timeStepDuration_, threeDviewSpeed_] :=
  Module[{finalDisplayImage, icData, data, h, m, nRow, timeScaleFor3Dplot, title, plotLabel},

   nRow = Dimensions[grid][[1]];
   timeScaleFor3Dplot = findOptimalTimeScale[stepNumber, maxTime, timeStepDuration];
   m = updateAnimationBuffer[stepNumber, Unevaluated@animation3dBuffer, grid, currentTime, u];
   title = makeTopTitleForPlot[currentTime, timeStepDuration];

   plotLabel = Grid[{
      {pde},
      {title}
     }, Alignment → Center, Spacings → {.2, .3}, Frame → True, FrameStyle -> Directive[Thickness[.003], Gray]
    ];

   Which[
    threeDView,
    finalDisplayImage = ListPlot3D[animation3dBuffer[[2]][[1 ;; m]],
      AxesLabel → { Text@Style["x", Italic, 11], Text@Row[{Spacer[15], Style["time", 11]}], None},
      PlotLabel -> plotLabel,
      MaxPlotPoints → 10,
      PlotRange → {{grid[[1]], grid[[-1]]}, {0, timeScaleFor3Dplot}, All},
      DataRange → All,
      If[threeDviewSpeed && stepNumber > 1, PerformanceGoal → "Speed", PerformanceGoal → "Quality"],
      If[threeDviewSpeed, Mesh → Automatic, Mesh → 8],
      ImageSize → {ContentSizeW - 20, ContentSizeH - 20},
      BoxRatios → {1, 1, .5},
      ImagePadding → {{20, 35}, {10, 40}}
     ],

    True, (*2D view*)
    icData = Thread[{grid, u0}];
    If[Not[yscaleAuto], h = Mean[u0] - Min[u0]];

    data = Thread[{grid, u}];

    finalDisplayImage =
     ListPlot[Evaluate@If[showIC, {data, icData}, data],
      If[joinedType == "joined" || joinedType == "line", Joined → True, Joined → False],
      ImagePadding → {{40, 15}, {40, 60}},
      If[yscaleAuto, PlotRange → All,
```

```
        PlotRange → {All, {Mean[u0] - yscaleAmount * h, Mean[u0] + h * yscaleAmount}}],
       ImageSize → {ContentSizeW - 20, ContentSizeH - 20},
       Frame → True,
       Axes → False,
       FrameLabel → {{None, None}, {Text@Style["x", Italic, 12], plotLabel}},
       AspectRatio → 1.4,
       Evaluate@If[addGrid,
          (
           {GridLines → Automatic, GridLinesStyle → Directive[Thickness[.005], Gray, Dashed]}
          ),
          GridLines → None
        ],
       Evaluate@If[showIC, PlotStyle -> {Blue, Red}, PlotStyle → Blue],
       Evaluate@
        If[joinedType == "joined" || joinedType == "points", PlotMarkers → Automatic, PlotMarkers → None]
      ];
   ];

   finalDisplayImage
  ];
(*-------------------------------------------------------------------------------*)
initializeSystemPDE4[initialConditionFunction_, h_, centerGrid_,
   length_, k_, a_, d_, c_, isPeriodicBC_, westBCtype_, eastBCtype_, maxTime_] := Module[
   {u, grid, cpuTimeUsed = 0., stepNumber = 0, Aleft, Aright, currentTime = 0., n, animation3dBuffer = {0, 0}},
   animation3dBuffer[[2]] = Table[0, {Ceiling[maxTime / k] + 25}];
   animation3dBuffer[[1]] = 0;
   grid = N[generatePhysicalCoordinates1D[h, length, centerGrid]];
   n = Length[grid];
   u = Map[initialConditionFunction[#] &, grid];
   {Aleft, Aright} = makeSystemMatrix1DHeatEqPDE4[k, h, d, c, a, n, westBCtype, eastBCtype, isPeriodicBC];
   {u, grid, cpuTimeUsed, stepNumber, Aleft, Aright, currentTime, animation3dBuffer}
  ];
(*-------------------------------------------------------------------------------*)
solvePDE4[$u_, Aleft_, Aright_, grid_, k_, h_, d_, c_, westBoundaryFunction_,
   eastBoundaryFunction_, sourceFunction_, currentTime_, westBCtype_, eastBCtype_,
   sourceDurationType_, sourceDuration_, isPeriodicBC_] := Module[{u = $u, b, n, rhs},

   n = Length[grid];
   b = makeForceVector1DHeatEqPDE4[k, h, d, c, westBoundaryFunction, eastBoundaryFunction, sourceFunction,
     currentTime, n, westBCtype, eastBCtype, sourceDurationType, sourceDuration, grid, isPeriodicBC];
   rhs = Aright.u + b;
   u = Chop@LinearSolve[Aleft, rhs];
   If[isPeriodicBC, u[[1]] = u[[-1]]];
   {u, 0}
  ];
(*-------------------------------------------------------------------------------*)
makeSystemMatrix1DHeatEqPDE4[k_, h_, d_, c_, a_, n_, leftBC_, rightBC_, isPeriodicBC_] :=
  Module[{Aleft, Aright, r1, r2, r3},
   r1 = 1. + k * c / (d * h^2) + a * k / (2 d);
   r2 = k * c / (2. * d * h^2);
   r3 = 1. - k * c / (d * h^2) - a * k / (2 d);

   If[isPeriodicBC,
    (
     Aleft = SparseArray[{
        Band[{1, 1}] → r1,
        Band[{2, 1}] → -r2,
        Band[{1, 2}] → -r2
```

```
    }, {n, n}
   ];
  Aleft[[1, 1]] = 1;
  Aleft[[1, 2]] = 0;
  Aleft[[2, 1]] = 0;
  Aleft[[2, -1]] = -r2;
  Aleft[[-1, 2]] = -r2;

  Aright = SparseArray[{
     Band[{1, 1}] → r3,
     Band[{2, 1}] → r2,
     Band[{1, 2}] → r2
    }, {n, n}
   ];

  Aright[[1, 1]] = 1;
  Aright[[1, 2]] = 0;
  Aright[[2, 1]] = 0;
  Aright[[2, -1]] = r2;
  Aright[[-1, 2]] = r2;
 ),
 (
  Which[leftBC == "Neumann" && rightBC == "Neumann",
   (
    Aleft = SparseArray[{
       Band[{1, 1}] → r1,
       Band[{2, 1}] → -r2,
       Band[{1, 2}] → -r2
      }, {n, n}
     ];

    Aleft[[1, 2]] = -2.0 r2;
    Aleft[[-1, -2]] = -2.0 r2;

    Aright = SparseArray[{
       Band[{1, 1}] → r3,
       Band[{2, 1}] → r2,
       Band[{1, 2}] → r2
      }, {n, n}
     ];

    Aright[[1, 2]] = 2.0 r2;
    Aright[[-1, -2]] = 2.0 r2
   ),
   leftBC == "Neumann" && rightBC == "Dirichlet",
   (
    Aleft = SparseArray[{
       Band[{1, 1}] → r1,
       Band[{2, 1}] → -r2,
       Band[{1, 2}] → -r2
      }, {n, n}
     ];

    Aleft[[1, 2]] = -2.0 r2;
    Aleft[[-1, -1]] = 1.;
    Aleft[[-1, -2]] = 0.;
    Aleft[[-2, -1]] = 0.;

    Aright = SparseArray[{
       Band[{1, 1}] → r3,
       Band[{2, 1}] → r2,
       Band[{1, 2}] → r2
      }, {n, n}
     ];
```

```
    Aright[[1, 2]] = 2.0 r2;
    Aright[[-1, -1]] = 0.;
    Aright[[-1, -2]] = 0.;
    Aright[[-2, -1]] = 0.
   ),
   leftBC == "Dirichlet" && rightBC == "Neumann",
   (
    Aleft = SparseArray[{
        Band[{1, 1}] → r1,
        Band[{2, 1}] → -r2,
        Band[{1, 2}] → -r2
       }, {n, n}
     ];

    Aleft[[1, 1]] = 1.;
    Aleft[[1, 2]] = 0.;
    Aleft[[2, 1]] = 0.;
    Aleft[[-1, -2]] = -2. * r2;

    Aright = SparseArray[{
        Band[{1, 1}] → r3,
        Band[{2, 1}] → r2,
        Band[{1, 2}] → r2
       }, {n, n}
     ];

    Aright[[1, 1]] = 0.0;
    Aright[[1, 2]] = 0.;
    Aright[[2, 1]] = 0.;
    Aright[[-1, -2]] = 2 * r2
   ),
   leftBC == "Dirichlet" && rightBC == "Dirichlet",
   (
    Aleft = SparseArray[{
        Band[{1, 1}] → r1,
        Band[{2, 1}] → -r2,
        Band[{1, 2}] → -r2
       }, {n, n}
     ];

    Aleft[[1, 1]] = 1.;
    Aleft[[1, 2]] = 0.;
    Aleft[[2, 1]] = 0.;
    Aleft[[-1, -1]] = 1.;
    Aleft[[-1, -2]] = 0.;
    Aleft[[-2, -1]] = 0.;

    Aright = SparseArray[{
        Band[{1, 1}] → r3,
        Band[{2, 1}] → r2,
        Band[{1, 2}] → r2
       }, {n, n}
     ];
    Aright[[1, 1]] = 0.;
    Aright[[1, 2]] = 0.;
    Aright[[2, 1]] = 0.;
    Aright[[-1, -1]] = 0.;
    Aright[[-1, -2]] = 0.;
    Aright[[-2, -1]] = 0.
   )
  ]
 )
];
```

```
    {Aleft, Aright}
   ];
(*-------------------------------------------------------------------------------*)
makeForceVector1DHeatEqPDE4[k_, h_, d_, c_, leftFunction_, rightFunction_, forceFunction_,
    tNow_, n_, leftBC_, rightBC_, sourceDurationType_, sourceDuration_, grid_, isPeriodicBC_] :=
  Module[{b, r2, r4, forceFunctionNow, forceFunctionNext},

    forceFunctionNow = Map[forceFunction[#, tNow] &, grid];
    forceFunctionNext = Map[forceFunction[#, tNow + k] &, grid];
    r4 = k/(2 d);

    b = Table[0, {n}];
    If[isPeriodicBC,
     (
      b[[2 ;; -2]] = r4 * (forceFunctionNow[[2 ;; -2]] + forceFunctionNext[[2 ;; -2]])
     )
     ,
     (
      r2 = (k * c)/(2 * d * h^2);

      Which[leftBC == "Neumann" && rightBC == "Neumann",
       (
        b[[1]] = 2 * r2 * h * (leftFunction[tNow] + leftFunction[tNow + k]);

        Which[sourceDurationType == "simulation duration" ||
           (sourceDurationType == "specify duration" && tNow < sourceDuration),
         b[[2 ;; -2]] = r4 * (forceFunctionNow[[2 ;; -2]] + forceFunctionNext[[2 ;; -2]]),

         (sourceDurationType == "one time step" && tNow ≤ k) ||
          (sourceDurationType == "specify duration" && tNow == sourceDuration),
         b[[2 ;; -2]] = r4 * forceFunctionNow[[2 ;; -2]],

         True, b[[2 ;; -2]] = 0.
        ];

        b[[n]] = 2 * r2 * h * (rightFunction[tNow] + rightFunction[tNow + k])
       ),

       leftBC == "Neumann" && rightBC == "Dirichlet",
       (
        b[[1]] = 2 * r2 * h * (leftFunction[tNow] + leftFunction[tNow + k]);

        Which[sourceDurationType == "simulation duration" ||
           (sourceDurationType == "specify duration" && tNow < sourceDuration),
         (
          b[[2 ;; -3]] = r4 * (forceFunctionNow[[2 ;; -3]] + forceFunctionNext[[2 ;; -3]]);
          b[[-2]] = r4 * (forceFunctionNow[[-2]] + forceFunctionNext[[-2]]) + r2 * rightFunction[tNow]
         ),

         (sourceDurationType == "one time step" && tNow ≤ k) ||
          (sourceDurationType == "specify duration" && tNow == sourceDuration),
         (
          b[[2 ;; -3]] = r4 * (forceFunctionNow[[2 ;; -3]]);
          b[[-2]] = r4 * (forceFunctionNow[[-2]]) + r2 * rightFunction[tNow]
         ),
         True,
         (
```

```
    b[[2 ;; -3]] = 0;
    b[[-2]] = r2 * rightFunction[tNow]
   )
  ];

  b[[-1]] = rightFunction[tNow + k]
 ),

 leftBC == "Dirichlet" && rightBC == "Neumann",
 (
  b[[1]] = leftFunction[tNow + k];
  Which[sourceDurationType == "simulation duration" ||
     (sourceDurationType == "specify duration" && tNow < sourceDuration),
   (
    b[[2]] = r4 * (forceFunctionNow[[2]] + forceFunctionNext[[2]]) + r2 * leftFunction[tNow];
    b[[3 ;; -2]] = r4 * (forceFunctionNow[[3 ;; -2]] + forceFunctionNext[[3 ;; -2]])
   ),

   (sourceDurationType == "one time step" && tNow ≤ k) ||
    (sourceDurationType == "specify duration" && tNow == sourceDuration),
   (
    b[[2]] = r4 * (forceFunctionNow[[2]]) + r2 * leftFunction[tNow];
    b[[3 ;; -2]] = r4 * (forceFunctionNow[[3 ;; -2]])
   ),
   True,
   (
    b[[2]] = r2 * leftFunction[tNow];
    b[[3 ;; -2]] = 0.
   )
  ];

  b[[-1]] = 2 * r2 * h * (rightFunction[tNow] + rightFunction[tNow + k])

 ),
 leftBC == "Dirichlet" && rightBC == "Dirichlet",
 (
  b[[1]] = leftFunction[tNow + k];

  Which[sourceDurationType == "simulation duration" ||
     (sourceDurationType == "specify duration" && tNow < sourceDuration),
   (
    b[[2]] = r2 * (leftFunction[tNow] + leftFunction[tNow + k]) +
       r4 * (forceFunctionNow[[2]] + forceFunctionNext[[2]]);
    b[[3 ;; -3]] = r4 * (forceFunctionNow[[3 ;; -3]] + forceFunctionNext[[3 ;; -3]]);
    b[[-2]] = r2 * (rightFunction[tNow] + rightFunction[tNow + k]) +
       r4 * (forceFunctionNow[[-2]] + forceFunctionNext[[-2]])
   ),
   (sourceDurationType == "one time step" && tNow ≤ k) ||
    (sourceDurationType == "specify duration" && tNow == sourceDuration),
   (
    b[[2]] = r2 * (leftFunction[tNow] + leftFunction[tNow + k]) + r4 * (forceFunctionNow[[2]]);
    b[[3 ;; -3]] = r4 * (forceFunctionNow[[3 ;; -3]]);
    b[[-2]] = r2 * (rightFunction[tNow] + rightFunction[tNow + k]) + r4 * (forceFunctionNow[[-2]])
   ), True,
   (
    b[[2]] = r2 * (leftFunction[tNow] + leftFunction[tNow + k]);
    b[[3 ;; -3]] = 0.;
    b[[-2]] = r2 * (rightFunction[tNow] + rightFunction[tNow + k])
   )

  ];

  b[[-1]] = rightFunction[tNow + k]
 )
```

```
      ]
     )
    ];
   b
  ];
(*-------------------------------------------------------------------------------*)
makeForceFunction[sel_, a_, b_, c_, d_, currentTime___] := Module[{},
   Which[
    sel == 0, {a, Function[{x, t}, a*DiscreteDelta[x]]},
    sel == 1, {a, Function[{x, t}, a]},
    sel == 2, {a + d*HoldForm[Exp[-currentTime]], Function[{x, t}, a + d*Exp[-t]]},
    sel == 3,
    If[b == 0,
     {Chop@a + Chop@d*HoldForm[Exp[-currentTime]], Function[{x, t}, a + d*Exp[-t]]},
     {Chop@a*HoldForm[x^b] + Chop@d*HoldForm[Exp[-currentTime]], Function[{x, t}, a*x^b + d*Exp[-t]]}
    ],
    sel == 4,
    If[b == 0,
     {Chop@a*HoldForm[Exp[-currentTime]], Function[{x, t}, a*Exp[-t]]},
     {Chop@a*HoldForm[x^b*Exp[-currentTime]], Function[{x, t}, a*x^b*Exp[-t]]}
    ]
    ,
    sel == 5,
    If[b == 0,
     {Chop@a + Chop@d*HoldForm[Exp[-currentTime]], Function[{x, t}, a + d *Exp[-t]]},
     {Chop@a*HoldForm[(Sin[c*x])^b] + Chop@d *HoldForm[Exp[-currentTime]],
      Function[{x, t}, a*(Sin[c*x])^b + d *Exp[-t]]}
    ]
    ,
    sel == 6,
    If[b == 0,
     {Chop@a*HoldForm[Exp[-currentTime]], Function[{x, t}, a*Exp[-t]]},
     {Chop@a*HoldForm[(Sin[c*x])^b *Exp[-currentTime]], Function[{x, t}, a*(Sin[c*x])^b *Exp[-t]]}
    ]
    ,
    sel == 7,
    If[b == 0,
     {Chop@a + Chop@d*HoldForm[Exp[-currentTime]], Function[{x, t}, a + d *Exp[-t]]},
     {Chop@a*HoldForm[(Cos[c*x])^b] + Chop@d *HoldForm[Exp[-currentTime]],
      Function[{x, t}, a*(Cos[c*x])^b + d *Exp[-t]]}
    ]
    ,
    sel == 8,
    If[b == 0,
     {Chop@a*HoldForm[Exp[-currentTime]], Function[{x, t}, a *Exp[-t]]},
     {Chop@a*HoldForm[(Cos[c*x])^b *Exp[-currentTime]], Function[{x, t}, a*(Cos[c*x])^b *Exp[-t]]}
    ]
   ]
  ];
(*-------------------------------------------------------------------------------*)
makeBoundaryCondition[sel_, a_, b_] := Module[{},
   Which[
    sel == 1, {setForm[a, 11], Function[{t}, a]},
    sel == 2, {setForm[a Sin[b t], 11], Function[{t}, a Sin[b t] ]},
    sel == 3, {setForm[a Cos[b t], 11], Function[{t}, a Cos[b t] ]},
    sel == 4, {setForm[a Sin[b t] Exp[-t], 11], Function[{t}, a Sin[b t] Exp[-t] ]},
    sel == 5, {setForm[a Cos[b t] Exp[-t], 11], Function[{t}, a Cos[b t] Exp[-t] ]},
    sel == 6, {setForm[t, 11], Function[{t}, t]}
   ]
  ];
(*-------------------------------------------------------------------------------*)
```

```
setForm[s_, siz_] := Text@Style[HoldForm[TraditionalForm[s]], siz];
(*------------------------------------------------------------------------------*)
makeInitialCondition[sel_, a_, b_, c_, d_, stdx_, x0_] :=
  Which[sel == 1, Function[{x}, a],
   sel == 2, Function[{x}, a x],
   sel == 3, Function[{x}, a x + b x^2],
   sel == 4, Function[{x}, a x + b x^2 + c x^3],
   sel == 5, Function[{x}, a x + b x^2 + c x^3 + d x^4],
   sel == 6, Function[{x}, a Sin[b x]],
   sel == 7, Function[{x}, a Cos[b x]],
   sel == 8, Function[{x}, a Sin[b x] + c Sin[d x]],
   sel == 9, Function[{x}, a Sin[b x] + c Cos[d x]],
   sel == 10, Function[{x}, a Cos[b x] + c Cos[d x]],
   sel == 11, Function[{x}, a (Sin[b x])^2],
   sel == 12, Function[{x}, a (Cos[b x])^2],
   sel == 13, Function[{x}, a* Exp[b* (x - d)^c]],
   sel == 14, Function[{x}, 1/(stdx*Sqrt[2*Pi]) Exp[- (x - x0)^2/(2*stdx^2)]]
  ];
(*------------------------------------------------------------------------------*)
makeInitialConditionSpecial[sel_, c_, w_, h_] :=
  Which[sel == 0,
   Function[{x}, Piecewise[{
      {0, x < (c - w/2)},
      {0, x > (c + w/2)},
      {h/(w/2)*x + h (1 - c/(w/2)), x ≤ c},
      {-h/(w/2)*x + h (1 + c/(w/2)), x > c}
     }]],

   sel == 1,
   Function[{x}, Piecewise[{
      {0, x < (c - w/2)},
      {0, x > (c + w/2)},
      {h, True}
     }]],

   sel == 2,
   Function[{x},
    Piecewise[{
      {h/w*x + h (1 - c/w), x ≤ c && x > (c - w)},
      {0, True}
     }]],

   sel == 3,
   Function[{x}, Piecewise[{
      {-h/w*x + h (1 + c/w), x >= c && x < (c + w)},
      {0, True}
     }]]
  ];
(*------------------------------------------------------------------------------*)
makeInitialConditionSpecial[sel_, unitStepShift_, unitStepHeight_] :=
  Which[
   sel == 4, Function[{x}, unitStepHeight*UnitStep[x - unitStepShift]],
   sel == 5, Function[{x}, unitStepHeight*UnitStep[unitStepShift - x]]
  ];
(*------------------------------------------------------------------------------*)
triangle[x_, h_?(NumericQ[#] && # > 0 &), (*height*)
   c_?(NumericQ[#] &), (*center of triangle*)
   w_?(NumericQ[#] && # > 0 &)(*width of triangle*)] := Piecewise[{
    {0, x < (c - w/2)},
    {0, x > (c + w/2)},
    {h + h/(w/2)*x, x ≤ c},
    {h - h/(w/2)*x, x > c}
   }];
```

```
(*------------------------------------------------------------------------------*)
rectangle[x_, h_?(NumericQ[#] && # > 0 &), (*height*)
    c_?(NumericQ[#] &), (*center of triangle*)
    w_?(NumericQ[#] && # > 0 &)(*width of triangle*)] := Piecewise[{
     {0, x < (c - w/2)},
     {0, x > (c + w/2)},
     {h, True}
    }];
(*-------------------------------------------------------*)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];
ContentSizeW = 270 ;
ContentSizeH = 415;
```

Manipulate

```
Manipulate[
 gtick; (*system tick*)
 {finalDisplayImage, u, u0, grid, Aleft, Aright, stepNumber, cpuTimeUsed, currentTime, state} =
  processPDE4[u, grid, Aleft, Aright, stepNumber, cpuTimeUsed, currentTime, state, u0,
   Unevaluated[animation3dBuffer], initialConditionFunction, event, h, centerGrid, length,
   k, aReactionTerm, dAdvectionTerm, cDiffusionTerm, maxTime, isPeriodicBC, westBCtype,
   eastBCtype, westBoundaryFunction, eastBoundaryFunction, sourceFunction, addGrid, showIC,
   joinedType, sourceDurationType, sourceDuration, yscaleAuto, yscaleAmount, threeDView,
   threeDviewSpeed, Unevaluated[gtick], Unevaluated[delta], Unevaluated@gstatusMessage];
 FinishDynamic[];
 Framed[finalDisplayImage , FrameStyle -> Directive[Thickness[.005], Gray]],

 Evaluate@With[{
    plotOptionsMacro = myGrid[{
       {
        Grid[{{Checkbox[Dynamic[addGrid, {addGrid = #; event = "plot_changed" , gtick += delta} &],
             Enabled → Dynamic[threeDView == False]]
          },
          {Style[Column[{"grid", "lines"}], 11]}
         }],

        RadioButtonBar[Dynamic[joinedType, {joinedType = #; event = "plot_changed"; gtick += delta} &],
         {"line" → Text@Style["line", 10], "points" → Text@Style["points", 10], "joined" →
           Text@Style["joined", 10]}, Appearance → "Vertical", Enabled → Dynamic[threeDView == False]],

        Grid[{
          {Text@Style["show initial conditions", 10],
           Checkbox[Dynamic[showIC, {showIC = #; event = "plot_changed" , gtick += delta} &],
            Enabled → Dynamic[threeDView == False]]
          },
          {Text@Style["3D solution plot", 10],
           Checkbox[Dynamic[threeDView, {threeDView = #; event = "plot_changed" , gtick += delta} &]]
          },
          {Text@Style["3D plot for speed", 10],
           Checkbox[Dynamic[threeDviewSpeed, {threeDviewSpeed = #; event =
                 "plot_changed" , gtick += delta} &], Enabled → Dynamic[threeDView == True]]
          }
         }, Spacings → {.2, 0}, Alignment → Left]
       }
```

```
  },
  Alignment → Center, Spacings → {.6, .5},
  Dividers -> {All, True}, FrameStyle -> Directive[Thickness[.005], Gray]
 ],
(*------------------------*)
(*--- TOP ROW macro  -----*)
(*------------------------*)
topRowMacro = Item[Grid[{
    {
     Button[Text[Style["solve", 12]], {event = "run_button"; gtick += delta}, ImageSize -> {50, 35}],
     Button[Text[Style["pause", 12]], {event = "pause_button"; gtick += delta}, ImageSize -> {52, 35}],
     Button[Text[Style["step", 12]], {event = "step_button"; gtick += delta}, ImageSize -> {48, 35}],
     Button[Text[Style["reset", 12]], {event = "reset"; gtick += delta}, ImageSize -> {48, 35}],
     "",
     Graphics[Text[Style[Dynamic@gstatusMessage, 12]],
      ImageSize -> {90, 30}, ImagePadding → {{70, 75}, {75, 75}}], SpanFromLeft
    }}, Spacings → {0.1, 0}
  ], Alignment → {Center, Top}],
(*---------------------------*)
(*--- parametersMacro macro --*)
(*---------------------------*)
parametersMacro = Item[Grid[{
    {
     Grid[{
       {
        Text@Style["test", 12], PopupMenu[Dynamic[testCase, {testCase = #;
            Which[testCase == 1,
             (
              h = 0.03; length = 1; k = 0.02; aReactionTerm = 0.; dAdvectionTerm = 1.; cDiffusionTerm =
               0.0616; maxTime = 5; centerGrid = False; isPeriodicBC = False; westBCtype = "Neumann";
              westbc = 1; awestBCconstantValue = 0; bwestBCconstantValue = 0; eastBCtype = "Neumann";
              eastbc = 1; aeastBCconstantValue = 0; beastBCconstantValue = 0; westBoundaryFunction =
               makeBoundaryCondition[westbc, awestBCconstantValue, bwestBCconstantValue][[2]];
              eastBoundaryFunction = makeBoundaryCondition[eastbc, aeastBCconstantValue,
                 beastBCconstantValue][[2]];
              forceTermSelection = 1; Sa = 0; Sb = 0; Sc = 0; Sd = 0; sourceFunction =
               makeForceFunction[forceTermSelection, Sa, Sb, Sc, Sd][[2]];
              sourceDurationType = "simulation duration"; initialConditionsSelection = 7;
              ICa = 1; ICb = 2; ICc = 0; ICd = 0;
              initialConditionFunction = makeInitialCondition[
                initialConditionsSelection, ICa, ICb, ICc, ICd, stdx, x0]; addGrid = True;
              joinedType = "line"; showIC = True; yscaleAuto = False; yscaleAmount = 1.1
             ), testCase == 2,
             (
              h = 0.02; length = 1; k = 0.01; aReactionTerm = 0.; dAdvectionTerm = 1.;
              cDiffusionTerm = 0.0616; maxTime = 4; centerGrid = False; isPeriodicBC = False;
              westBCtype = "Dirichlet"; westbc = 1; awestBCconstantValue = 0; bwestBCconstantValue = 0;
              eastBCtype = "Dirichlet"; eastbc = 1; aeastBCconstantValue = 0; beastBCconstantValue = 0;
              westBoundaryFunction = makeBoundaryCondition[westbc, awestBCconstantValue,
                 bwestBCconstantValue][[2]];
              eastBoundaryFunction = makeBoundaryCondition[eastbc, aeastBCconstantValue,
                 beastBCconstantValue][[2]];
              forceTermSelection = 1; Sa = 0; Sb = 0; Sc = 0; Sd = 0;
              sourceFunction = makeForceFunction[forceTermSelection, Sa, Sb, Sc, Sd][[2]];
              sourceDurationType = "simulation duration";
              initialConditionsSelection = 7; ICa = 1; ICb = 2; ICc = 0; ICd = 0;
              initialConditionFunction = makeInitialCondition[initialConditionsSelection,
                ICa, ICb, ICc, ICd, stdx, x0]; addGrid = True;
              joinedType = "line"; showIC = True; yscaleAuto = False; yscaleAmount = 1.1
             ), testCase == 3,
             (
              h = 0.1; length = 1; k = 0.05; aReactionTerm = 0.; dAdvectionTerm = 1.; cDiffusionTerm = 0.2;
              maxTime = 5; centerGrid = False; isPeriodicBC = False; westBCtype = "Dirichlet";
```

```
 westbc = 1; awestBCconstantValue = 0; bwestBCconstantValue = 0; eastBCtype = "Dirichlet";
 eastbc = 1; aeastBCconstantValue = 0; beastBCconstantValue = 0;
 westBoundaryFunction = makeBoundaryCondition[westbc, awestBCconstantValue,
    bwestBCconstantValue][[2]];
 eastBoundaryFunction = makeBoundaryCondition[eastbc, aeastBCconstantValue,
    beastBCconstantValue][[2]];
 forceTermSelection = 1; Sa = 0; Sb = 0; Sc = 1; Sd = 0;
 sourceFunction = makeForceFunction[forceTermSelection, Sa, Sb, Sc, Sd][[2]];
 sourceDurationType = "simulation duration";
 initialConditionsSelection = 6; ICa = 1; ICb = 1; ICc = 0; ICd = 0;
 initialConditionFunction = makeInitialCondition[
   initialConditionsSelection, ICa, ICb, ICc, ICd, stdx, x0]; addGrid = True;
 joinedType = "line"; showIC = True; yscaleAuto = False; yscaleAmount = 1.1
), testCase == 4,
(
 h = 0.01; length = 1; k = 0.01; aReactionTerm = 0.; dAdvectionTerm = 1.;
 cDiffusionTerm = 0.2; maxTime = 1; centerGrid = False; isPeriodicBC = False;
 westBCtype = "Neumann"; westbc = 1; awestBCconstantValue = 0; bwestBCconstantValue = 0;
 eastBCtype = "Neumann"; eastbc = 1; aeastBCconstantValue = 0; beastBCconstantValue = 0;
 westBoundaryFunction = makeBoundaryCondition[westbc, awestBCconstantValue,
    bwestBCconstantValue][[2]];
 eastBoundaryFunction = makeBoundaryCondition[eastbc, aeastBCconstantValue,
    beastBCconstantValue][[2]];
 forceTermSelection = 1; Sa = 0; Sb = 0; Sc = 1; Sd = 0;
 sourceFunction = makeForceFunction[forceTermSelection, Sa, Sb, Sc, Sd][[2]];
 sourceDurationType = "simulation duration";
 choiceOfSpecialICfunction = 4 ; (*unit step*)
 unitStepShift = .2; unitStepHeight = 1;
 initialConditionFunction = makeInitialConditionSpecial[
   choiceOfSpecialICfunction, unitStepShift, unitStepHeight]; addGrid = True;
 joinedType = "line"; showIC = True; yscaleAuto = False; yscaleAmount = 1.1
),
testCase == 5,
(h = 0.03; length = 1; k = 0.01; aReactionTerm = 0.; dAdvectionTerm = 1.; cDiffusionTerm = 0.1;
 maxTime = 1; centerGrid = True; isPeriodicBC = False; westBCtype = "Dirichlet";
 westbc = 1; awestBCconstantValue = 0; bwestBCconstantValue = 0;
 eastBCtype = "Neumann"; eastbc = 1; aeastBCconstantValue = 0; beastBCconstantValue = 0;
 westBoundaryFunction =
  makeBoundaryCondition[westbc, awestBCconstantValue, bwestBCconstantValue][[2]];
 eastBoundaryFunction = makeBoundaryCondition[eastbc, aeastBCconstantValue,
    beastBCconstantValue][[2]];
 forceTermSelection = 1; Sa = 0; Sb = 0; Sc = 1; Sd = 0;
 sourceFunction = makeForceFunction[forceTermSelection, Sa, Sb, Sc, Sd][[2]];
 sourceDurationType = "simulation duration";
 choiceOfSpecialICfunction = 5; (*unit step*)unitStepShift = .2; unitStepHeight = 1;
 initialConditionFunction = makeInitialConditionSpecial[
   choiceOfSpecialICfunction, unitStepShift, unitStepHeight];
 addGrid = True; joinedType = "line"; showIC = True; yscaleAuto = False; yscaleAmount = 1.6
), testCase == 6,
(
 h = 0.02; length = 1; k = 0.01; aReactionTerm = 0.; dAdvectionTerm = 1.; cDiffusionTerm = 0.1;
 maxTime = 3; centerGrid = False; isPeriodicBC = False; westBCtype = "Dirichlet";
 westbc = 3; awestBCconstantValue = 0.5; bwestBCconstantValue = 10.;
 eastBCtype = "Neumann"; eastbc = 1; aeastBCconstantValue = 0;
 beastBCconstantValue = 0;
 westBoundaryFunction =
  makeBoundaryCondition[westbc, awestBCconstantValue, bwestBCconstantValue][[2]];
 eastBoundaryFunction = makeBoundaryCondition[eastbc, aeastBCconstantValue,
    beastBCconstantValue][[2]]; forceTermSelection = 1; Sa = 0; Sb = 0; Sc = 1;
 Sd = 0; sourceFunction = makeForceFunction[forceTermSelection, Sa, Sb, Sc, Sd][[2]];
 sourceDurationType = "simulation duration";
 choiceOfSpecialICfunction = 5; (*unit step*)unitStepShift = .3;
 unitStepHeight = 1; initialConditionFunction = makeInitialConditionSpecial[
   choiceOfSpecialICfunction, unitStepShift, unitStepHeight];
```

```
      yscaleAuto = True; addGrid = True; joinedType = "line"; showIC = True
     ), testCase == 7,
     (
      h = 0.02; length = 1; k = 0.01; aReactionTerm = 0.; dAdvectionTerm = 1.;
      cDiffusionTerm = 0.1; maxTime = 2; centerGrid = True; isPeriodicBC = True;
      forceTermSelection = 1; Sa = 0; Sb = 0; Sc = 1; Sd = 0;
      sourceFunction = makeForceFunction[forceTermSelection, Sa, Sb, Sc, Sd][[2]];
      sourceDurationType = "simulation duration"; choiceOfSpecialICfunction = 5; (*unit step*)
      unitStepShift = 0.0; unitStepHeight = 2;
      initialConditionFunction = makeInitialConditionSpecial[
        choiceOfSpecialICfunction, unitStepShift, unitStepHeight];
      addGrid = True; joinedType = "line"; showIC = True; yscaleAuto = False; yscaleAmount = 1.4
     ), testCase == 8,
     (
      h = 0.02; length = 1; k = 0.1; aReactionTerm = 0.; dAdvectionTerm = 1.; cDiffusionTerm = 1;
      maxTime = 2; centerGrid = False; isPeriodicBC = False; westBCtype = "Dirichlet";
      westbc = 1; awestBCconstantValue = 1.; bwestBCconstantValue = 0.; eastBCtype = "Dirichlet";
      eastbc = 1; aeastBCconstantValue = 0.; beastBCconstantValue = 0.;
      westBoundaryFunction =
       makeBoundaryCondition[westbc, awestBCconstantValue, bwestBCconstantValue][[2]];
      eastBoundaryFunction = makeBoundaryCondition[eastbc, aeastBCconstantValue,
         beastBCconstantValue][[2]];
      forceTermSelection = 1; Sa = 0; Sb = 0; Sc = 0; Sd = 0;
      sourceFunction = makeForceFunction[forceTermSelection, Sa, Sb, Sc, Sd][[2]];
      sourceDurationType = "simulation duration";
      choiceOfSpecialICfunction = 5; (*unit step*)unitStepShift = 0.5; unitStepHeight = 1;
      initialConditionFunction = makeInitialConditionSpecial[choiceOfSpecialICfunction,
        unitStepShift, unitStepHeight]; addGrid = True; joinedType = "line";
      showIC = True; yscaleAuto = False; yscaleAmount = 1.1
     ), testCase == 9,
     (
      h = 0.02; length = 1; k = 0.01; aReactionTerm = 0.; dAdvectionTerm = 1.;
      cDiffusionTerm = 1; maxTime = .9; centerGrid = True; isPeriodicBC = True;
      westBoundaryFunction = makeBoundaryCondition[westbc, awestBCconstantValue,
         bwestBCconstantValue][[2]];
      eastBoundaryFunction = makeBoundaryCondition[eastbc, aeastBCconstantValue,
         beastBCconstantValue][[2]];
      forceTermSelection = 1; Sa = 0; Sb = 0; Sc = 0; Sd = 0;
      sourceFunction = makeForceFunction[forceTermSelection, Sa, Sb, Sc, Sd][[2]];
      sourceDurationType = "simulation duration";
      choiceOfSpecialICfunction = 1; (*rectangle step*)ICcenter = 0.; ICwidth = .4;
      ICheight = 1.; initialConditionFunction =
       makeInitialConditionSpecial[choiceOfSpecialICfunction, ICcenter,
        ICwidth, ICheight]; addGrid = True; joinedType = "line"; showIC = True;
      yscaleAuto = True
     )
    ]; event = "reset"; gtick += delta} &],
  { 1 → Style["homogeneous Neumann BC", 11],
   2 → Style["homogeneous Dirichlet BC", 11],
   3 → Style["homogeneous Dirichlet BC", 11],
   4 → Style["homogeneous Neumann BC, unit step IC", 11],
   5 → Style["mixed BC, unit step IC", 11],
   6 → Style["inhomogeneous mixed BC, unit step IC", 11],
   7 → Style["periodic BC, unit step IC", 11],
   8 → Style["inaccuracy due to large spatial frequency", 11],
   9 → Style["large spatial frequency", 11]
  }, ImageSize -> All, ContinuousAction -> False]
 }
 }, Alignment → Center, Spacings → {.2, 0}, Frame → None
], SpanFromLeft
},
{
 Framed[Text[Style[Row[{Style["c", Italic], " ", Style["u", Italic]_{Row[{Style["x",Italic]Style["x",Italic]}]},
      " = ", Style["d", Italic] , " ", Style["u", Italic]_{Style["t",Italic]}, " + ", Style["a", Italic],
```

```
      " ", Style["u", Italic], " + ", Style["f", Italic], "(", Style["x", Italic], ", ",
      Style["t", Italic], ")"}], 12]], FrameStyle -> Directive[Thickness[.005], Gray]],
 SpanFromLeft
},
{
 Framed[Grid[{
    {
     Text@Style["grid size", 12],
     Spacer[3],
     Manipulator[Dynamic[h, {h = #; event = "reset"; gtick += delta} &],
      {0.01, 0.1, 0.01}, ImageSize → Small, ContinuousAction -> False],
     Spacer[3],
     Text@Style[Dynamic@padIt2[h, {3, 2}], 11]
    },
    {
     Text@Style["length", 12],
     Spacer[3],
     Manipulator[Dynamic[length, {length = #; event = "reset"; gtick += delta} &],
      {0.3, 5, 0.1}, ImageSize → Small, ContinuousAction -> False],
     Spacer[3],
     Text@Style[Dynamic@padIt2[length, {2, 1}], 11]
    },
    {
     Text@Style["time step", 12],
     Spacer[3],
     Manipulator[Dynamic[k, {k = #; event = "reset"; gtick += delta} &],
      {0.01, 0.1, 0.01}, ImageSize → Small, ContinuousAction -> False], Spacer[3],
     Text@Style[Dynamic@padIt2[k, {3, 2}], 11]
    },
    {
     Text@Style[Row[{Style["c", Italic], " (diffusion)"}], 12],
     Spacer[3],
     Manipulator[Dynamic[cDiffusionTerm, {cDiffusionTerm = #; event = "reset"; gtick += delta} &],
      {0.0, 2, 0.01}, ImageSize → Small, ContinuousAction -> False],
     Spacer[3],
     Text@Style[Dynamic@padIt2[cDiffusionTerm, {3, 2}], 11]
    },
    {
     Text@Style[Row[{Style["d", Italic], " (advection)"}], 12],
     Spacer[3],
     Manipulator[Dynamic[dAdvectionTerm, {dAdvectionTerm = #; event = "reset"; gtick += delta} &],
      {0.01, 2, 0.01}, ImageSize → Small, ContinuousAction -> False],
     Spacer[3],
     Text@Style[Dynamic@padIt2[dAdvectionTerm, {3, 2}], 11]
    },
    {
     Text@Style[Row[{Style["a", Italic], " (reaction)"}], 12],
     Spacer[3],
     Manipulator[Dynamic[aReactionTerm, {aReactionTerm = #; event = "reset"; gtick += delta} &],
      {-10, 10, 0.1}, ImageSize → Small, ContinuousAction -> False],
     Spacer[3],
     Text@Style[Dynamic@padIt1[aReactionTerm, {3, 1}], 11]
    },
    {
     Text@Style["run time", 12],
     Spacer[3],
     Manipulator[Dynamic[maxTime, {maxTime = #; event = "reset"; gtick += delta} &],
      {0.1, 5, 0.1}, ImageSize → Small, ContinuousAction -> False],
     Spacer[3],
     Text@Style[Dynamic@padIt2[maxTime, {2, 1}], 11],
     Spacer[10],
     SpanFromLeft
    }
   }, Spacings → {0.1, 0.1}, Alignment → Left, Frame → None
```

```
      ], FrameStyle -> Directive[Thickness[.005], Gray]
     ], SpanFromLeft
    }}, Alignment → Center, Spacings → {0, .6}, Frame → None
  ], Alignment → {Center, Top}
 ],
(*-------------------------*)
(*--- geometryMacro macro --*)
(*-------------------------*)
geometryMacro = Item[Grid[{
    {
     Grid[{
       {
        Grid[{
          {Text@Style["centered grid ", 11],
           Spacer[3],
           Checkbox[Dynamic[centerGrid, {centerGrid = #; event = "reset"; gtick += delta} &]]
          },
          {
           Text@Style["periodic boundary conditions", 11],
           Spacer[3],
           Checkbox[Dynamic[isPeriodicBC, {isPeriodicBC = #; event = "reset"; gtick += delta} &]]
          }
         }, Spacings → {0, 0.1}, Alignment → Center, Frame → None
        ], SpanFromLeft
       },
       {
        Grid[{
          {
           myGrid[{
             {Text@Style["left side", 12]},
             Dividers → {Thin, Blue},
             {
              RadioButtonBar[Dynamic[westBCtype, {
                   westBCtype = #; event = "reset"; gtick += delta} &],
               {"Dirichlet" → Text@Style["Dirichlet", 10], "Neumann" → Text@Style["Neumann", 10]},
               Appearance → "Vertical", Enabled → Dynamic[isPeriodicBC == False]]
             },
             {
              PopupMenu[Dynamic[westbc,
                {westbc = #; westBoundaryFunction = makeBoundaryCondition[westbc, awestBCconstantValue,
                      bwestBCconstantValue][[2]]; event = "reset"; gtick += delta} &],
               { 1 → Style["a", Italic, 12],
                 2 → Style[TraditionalForm[HoldForm[a Sin[b t]]], 12],
                 3 → Style[TraditionalForm[HoldForm[a Cos[b t]]], 12],
                 4 → Style[TraditionalForm[HoldForm[a Sin[b t] Exp[-t]]], 12],
                 5 → Style[TraditionalForm[HoldForm[a Cos[b t] Exp[-t]]], 12],
                 6 → Style["t", Italic, 12]},
               ImageSize -> All, ContinuousAction -> False, Enabled → Dynamic[isPeriodicBC == False]]
             },
             {Grid[{
                {
                 Grid[{
                   {
                    Text@Style["a", Italic, 12],
                    Manipulator[Dynamic[awestBCconstantValue,
                      {awestBCconstantValue = #; westBoundaryFunction = makeBoundaryCondition[
                            westbc, awestBCconstantValue, bwestBCconstantValue][[2]];
                        event = "reset"; gtick += delta} &], {-10, 10, 0.1}, ImageSize -> Tiny,
                     ContinuousAction -> False, Enabled → Dynamic[isPeriodicBC == False]],

                    Text@Style[Dynamic@padIt1[awestBCconstantValue, {3, 1}], 10]
                   },
                   {
```

```
          Button[Text@Style["zero", 10], {awestBCconstantValue = 0.0;
            westBoundaryFunction = makeBoundaryCondition[westbc, awestBCconstantValue,
                bwestBCconstantValue][[2]]; event = "reset"; gtick += delta}, ImageSize →
            {45, 20}, Enabled → Dynamic[isPeriodicBC == False]], SpanFromLeft
          }
         }, Alignment → Center, Frame → True,
         FrameStyle -> Directive[Thickness[.005], Gray], Spacings → {.1, 0}
        ]
       },
       {
        Grid[{
          {
           Text@Style["b", Italic, 12],

           Manipulator[Dynamic[bwestBCconstantValue,
             {bwestBCconstantValue = #; westBoundaryFunction = makeBoundaryCondition[
                   westbc, awestBCconstantValue, bwestBCconstantValue][[2]];
               event = "reset"; gtick += delta} &], {-10, 10, 0.1}, ImageSize -> Tiny,
            ContinuousAction -> False, Enabled → Dynamic[isPeriodicBC == False]],

           Text@Style[Dynamic@padIt1[bwestBCconstantValue, {3, 1}], 10]
          },
          {
           Button[Text@Style["zero", 10], {bwestBCconstantValue = 0.0;
             westBoundaryFunction = makeBoundaryCondition[westbc, awestBCconstantValue,
                 bwestBCconstantValue][[2]]; event = "reset"; gtick += delta}, ImageSize →
             {45, 20}, Enabled → Dynamic[isPeriodicBC == False]], SpanFromLeft
           }
         }, Alignment → Center, Frame → True,
         FrameStyle -> Directive[Thickness[.005], Gray], Spacings → {.1, 0}
        ]
       }
      }, Alignment → Center, Spacings → {0, .5}
     ]
    },
    {
     Dynamic@makeBoundaryCondition[westbc, awestBCconstantValue, bwestBCconstantValue][[1]]
    }
   }, Spacings → {0, .8}, Alignment → Center
  ],
  myGrid[{
    {Text@Style["right side", 12]},
    Dividers → {Thin, Blue},
    {
     RadioButtonBar[Dynamic[eastBCtype, {eastBCtype = #; event = "reset"; gtick += delta} &],
      {"Dirichlet" → Text@Style["Dirichlet", 10], "Neumann" → Text@Style["Neumann", 10]},
      Appearance → "Vertical", Enabled → Dynamic[isPeriodicBC == False]]
    },
    {
     PopupMenu[Dynamic[eastbc,
       {eastbc = #; eastBoundaryFunction = makeBoundaryCondition[eastbc, aeastBCconstantValue,
            beastBCconstantValue][[2]]; event = "reset"; gtick += delta} &],
      { 1 → Style["a", Italic, 12],
       2 → Style[TraditionalForm[HoldForm[a Sin[b t]]], 12], 3 → Style[TraditionalForm[
          HoldForm[a Cos[b t]]], 12], 4 → Style[TraditionalForm[HoldForm[a Sin[b t] Exp[-t]]],
         12], 5 → Style[TraditionalForm[HoldForm[a Cos[b t] Exp[-t]]], 12],
       6 → Style["t", Italic, 12]},
      ImageSize -> All, ContinuousAction -> False, Enabled → Dynamic[isPeriodicBC == False]]
    },
    {Grid[{
       {
        Grid[{
          {Text@Style["a", Italic, 12],
```

```
                Manipulator[Dynamic[aeastBCconstantValue,
                  {aeastBCconstantValue = #; eastBoundaryFunction = makeBoundaryCondition[
                        eastbc, aeastBCconstantValue, beastBCconstantValue][[2]];
                    event = "reset"; gtick += delta} &], {-10, 10, 0.1}, ImageSize -> Tiny,
                 ContinuousAction -> False, Enabled → Dynamic[isPeriodicBC == False]],

                Text@Style[Dynamic@padIt1[aeastBCconstantValue, {3, 1}], 10]
               },
               {
                Button[Text@Style["zero", 10], {aeastBCconstantValue = 0.0;
                  eastBoundaryFunction = makeBoundaryCondition[eastbc, aeastBCconstantValue,
                      beastBCconstantValue][[2]]; event = "reset"; gtick += delta}, ImageSize →
                  {45, 20}, Enabled → Dynamic[isPeriodicBC == False]], SpanFromLeft
               }
              }, Alignment → Center, Frame → True,
              FrameStyle -> Directive[Thickness[.005], Gray], Spacings → {.1, 0}
             ]
            },
            {
             Grid[{
               {
                Text@Style["b", Italic, 12],

                Manipulator[Dynamic[beastBCconstantValue,
                  {beastBCconstantValue = #; eastBoundaryFunction = makeBoundaryCondition[
                        eastbc, aeastBCconstantValue, beastBCconstantValue][[2]];
                    event = "reset"; gtick += delta} &], {-10, 10, 0.1}, ImageSize -> Tiny,
                 ContinuousAction -> False, Enabled → Dynamic[isPeriodicBC == False]],

                Text@Style[Dynamic@padIt1[beastBCconstantValue, {3, 1}], 10]
               },
               {
                Button[Text@Style["zero", 10], {beastBCconstantValue = 0.0;
                  eastBoundaryFunction = makeBoundaryCondition[eastbc, aeastBCconstantValue,
                      beastBCconstantValue][[2]]; event = "reset"; gtick += delta}, ImageSize →
                  {45, 20}, Enabled → Dynamic[isPeriodicBC == False]], SpanFromLeft
               }
              }, Alignment → Center, Frame → True,
              FrameStyle -> Directive[Thickness[.005], Gray], Spacings → {.1, 0}
             ]
            }
           }, Alignment → Center, Spacings → {0, .5}
          ]

         },
         {
          Dynamic@makeBoundaryCondition[eastbc, aeastBCconstantValue, beastBCconstantValue][[1]]
         }
        }, Spacings → {0, .8}, Alignment → Center]
    }
   }, Alignment → Center, Spacings → {0.6, 0.15},
   Frame → All, FrameStyle -> Directive[Thickness[.005], Gray]], SpanFromLeft
},
{
 Grid[{
   {
    Text@Style[Row[{"auto ", Style["y", Italic], " scale "}], 11],
    Checkbox[Dynamic[yscaleAuto, {yscaleAuto = #; event = "plot_changed" , gtick += delta} &],
     Enabled → Dynamic[threeDView == False]], SpanFromLeft
   },
   {
    Text@Style["manual", 11],
    Manipulator[Dynamic[yscaleAmount,
```

```
            {yscaleAmount = #; event = "plot_changed"; gtick += delta} &], {.1, 3, 0.1}, ImageSize -> Tiny,
           ContinuousAction -> False, Enabled → Dynamic[yscaleAuto == False && threeDView == False]],
          Text@Style[Dynamic@padIt2[yscaleAmount, {3, 1}], 11]
         }
        }, Alignment → Left, Frame → None], SpanFromLeft
      }
     }, Alignment → Center, Frame → None, FrameStyle -> Directive[Thickness[.005], Gray]
    ], SpanFromLeft
   }

  }, Alignment → Center, Spacings → {0, 1}, Frame → None
 ], Alignment → {Center, Top}],
(*-----------------------------------*)
(*---sourceMacro macro --*)
(*-----------------------------------*)
sourceMacro = Item[Grid[{
    {PopupMenu[Dynamic[forceTermSelection, {forceTermSelection = #;
          sourceFunction = makeForceFunction[forceTermSelection, Sa, Sb, Sc, Sd][[2]];
          event = "reset";
          gtick += delta} &],
      {0 → Style[TraditionalForm[HoldForm[ξ δ[0]]], 12],
       1 → Style[TraditionalForm[HoldForm[ξ]], 12],
       2 → Style[TraditionalForm[HoldForm[ξ + γ Exp[-t]]], 12],
       3 → Style[TraditionalForm[HoldForm[ξ x^β + γ Exp[-t]]], 12],
       4 → Style[TraditionalForm[HoldForm[ξ x^β Exp[-t]]], 12],
       5 → Style[TraditionalForm[HoldForm[ξ (Sin[η x])^β + γ Exp[-t]]], 12],
       6 → Style[TraditionalForm[HoldForm[ξ (Sin[η x])^β Exp[-t]]], 12],
       7 → Style[TraditionalForm[HoldForm[ξ (Cos[η x])^β + γ Exp[-t]]], 12],
       8 → Style[TraditionalForm[HoldForm[ξ (Cos[η x])^β  Exp[-t]]], 12]
      }, ImageSize → {260, 35}, ContinuousAction -> False], SpanFromLeft
    },
    {
     Grid[{
       {Text@Style[TraditionalForm[HoldForm[ξ]], 12]],
        Spacer[4],
        Manipulator[Dynamic[Sa,
          {Sa = #; sourceFunction = makeForceFunction[forceTermSelection, Sa, Sb, Sc, Sd][[2]]; event =
              "reset"; gtick += delta} &], {-99, 99, 0.1}, ImageSize → Small, ContinuousAction -> False],
        Spacer[2],
        Text@Style[Dynamic@padIt1[Sa, {3, 1}], 11],
        Spacer[2],
        Button[Text@Style["zero", 10], {Sa = 0.; event = "reset";
          sourceFunction = makeForceFunction[forceTermSelection, Sa, Sb, Sc, Sd][[2]];
          gtick += delta}, ImageSize → {45, 20}, Alignment → Center]
       },
       {Text@Style[TraditionalForm[HoldForm[β], 12]],
        Spacer[4],
        Manipulator[Dynamic[Sb,
          {Sb = #; sourceFunction = makeForceFunction[forceTermSelection, Sa, Sb, Sc, Sd][[2]]; event =
              "reset"; gtick += delta} &], {0, 10, 1}, ImageSize → Small, ContinuousAction -> False],
        Spacer[2],
        Text@Style[Dynamic@padIt2[Sb, {3, 1}], 11],
        Spacer[2],
        Button[Text@Style["zero", 10], {Sb = 0; event = "reset";
          sourceFunction = makeForceFunction[forceTermSelection, Sa, Sb, Sc, Sd][[2]];
          gtick += delta}, ImageSize → {45, 20}, Alignment → Center]
       },
       {
        Text@Style[TraditionalForm[HoldForm[η], 12]],
        Spacer[4],
        Manipulator[Dynamic[Sc,
          {Sc = #; sourceFunction = makeForceFunction[forceTermSelection, Sa, Sb, Sc, Sd][[2]]; event =
              "reset"; gtick += delta} &], {-10, 10, 0.1}, ImageSize → Small, ContinuousAction -> False],
```

```
    Spacer[2],
    Text@Style[Dynamic@padIt1[Sc, {3, 1}], 11],
    Spacer[2],
    Button[Text@Style["zero", 10], {Sc = 0.; event = "reset";
      sourceFunction = makeForceFunction[forceTermSelection, Sa, Sb, Sc, Sd][[2]];
      gtick += delta}, ImageSize → {45, 20}, Alignment → Center]
   }
   ,
   {Text@Style[TraditionalForm[HoldForm[ɣ], 12]],
    Spacer[4],
    Manipulator[Dynamic[Sd,
      {Sd = #; sourceFunction = makeForceFunction[forceTermSelection, Sa, Sb, Sc, Sd][[2]]; event =
          "reset"; gtick += delta} &], {-10, 10, 0.1}, ImageSize → Small, ContinuousAction -> False],
    Spacer[2],
    Text@Style[Dynamic@padIt1[Sd, {3, 1}], 11],
    Spacer[2],
    Button[Text@Style["zero", 10], {Sd = 0.; event = "reset";
      sourceFunction = makeForceFunction[forceTermSelection, Sa, Sb, Sc, Sd][[2]];
      gtick += delta}, ImageSize → {45, 20}, Alignment → Center]
   }}, Spacings → {0, .5}
 ], SpanFromLeft
},
{
 Grid[{
   {Text@Style["source duration", 12]},
   {RadioButtonBar[
     Dynamic[sourceDurationType, {sourceDurationType = #; event = "reset"; gtick += delta} &],
     {
      "simulation duration" → Text@Style["simulation duration", 10],
      "one time step" → Text@Style["one time step", 10],
      "specify duration" → Text@Style["duration", 10]
     }, Appearance → "Horizontal"
    ]
   },
   {
    Row[{Text@Style["duration", 11], Spacer[3],
      Manipulator[Dynamic[sourceDuration, {sourceDuration = #; event = "reset"; gtick += delta} &],
       {0., 5, 0.1}, ImageSize → Small, ContinuousAction -> False, Enabled →
        Dynamic[sourceDurationType == "specify duration"]],
      Spacer[3],
      Text@Style[Dynamic@padIt2[sourceDuration, {2, 1}], 11]
     }]
   }
  }, Spacings → {0, .3}, Alignment → Center, Frame → None
 ], SpanFromLeft
},
{
 Dynamic@Grid[{
    {
     Block[{from, to, f, fh, emptyPlot, plotLength = 115, aspectRatio = 0.3},

      emptyPlot = ListPlot[{0},
        ImagePadding → {{40, 10}, {20, 30}},
        ImageMargins → 0,
        PlotRange → All,
        Frame → True,
        Axes → None,
        PlotStyle → Red,
        FrameLabel → {{None, None}, {None, Text@Style[Row[{Style["f", Italic],
                "(", Style["x", Italic], ",", Style["t", Italic], ") = 0"}], 12]}},
        ImageSize → {260 , plotLength},
        TicksStyle → 9,
        AspectRatio → aspectRatio,
        Evaluate@If[addGrid, GridLines → Automatic, GridLines → None]
```

```
 ];

If[forceTermSelection == 0, (*impulse*)
 (
  If[(sourceDurationType == "simulation duration") ||
     (sourceDurationType == "one time step" && currentTime ≤ k) ||
     (sourceDurationType == "specify duration" && currentTime <= sourceDuration),
   (
    Graphics[{Arrow[{{0, 0}, {0, Sa}}]},
     ImagePadding → {{40, 10}, {20, 30}},
     ImageMargins → 0,
     PlotRange → All,
     Frame → True,
     Axes → None,
     FrameLabel → {{None, None},
       {None, Text@Row[{Style[Row[{Style["f", Italic], "(", Style["x", Italic],
               ", ", Style["t", Italic], ") = "}], 11], Spacer[4], "δ(0)"}]}},
     ImageSize → {260 , plotLength},
     TicksStyle → 9,
     AspectRatio → aspectRatio,
     Evaluate@If[addGrid, GridLines → Automatic, GridLines → None]
    ]
   ),
   (
    emptyPlot
   )
  ]
 )
 ,
 (
  If[(sourceDurationType == "specify duration" && currentTime > sourceDuration) ||
     (sourceDurationType == "one time step" && currentTime > k),
   (
    emptyPlot
   ),
   (
    If[centerGrid,
     (
      from = -length / 2;
      to = length / 2
     ),
     (
      from = 0;
      to = length
     )
    ];

    {fh, f} = makeForceFunction[forceTermSelection, Sa, Sb, Sc, Sd, currentTime];

    Plot[f[x, currentTime], {x, from, to},
     ImagePadding → {{40, 10}, {20, 30}},
     ImageMargins → 0,
     PlotRange → All,
     Frame → True,
     Axes → None,
     PlotStyle → Red,
     FrameLabel → {{None, None}, {None, Text@Row[{Style[Row[{Style["f", Italic], "(", Style[
                 "x", Italic], ", ", Style["t", Italic], ") = "}], 11], Spacer[4], fh}]}},
     ImageSize → {260 (*300*), plotLength},
     TicksStyle → 9,
     AspectRatio → aspectRatio,
     Evaluate@If[addGrid, GridLines → Automatic, GridLines → None]
    ]
   )
```

```
            ]
           )
          ]
         ]
        }
       }, Spacings → {0, 0}, Frame → None], SpanFromLeft
     }
    }, Spacings → {.25, .4},
    Alignment → Center, Frame → All, FrameStyle -> Directive[Thickness[.005], Gray]
   ], Alignment → {Center, Top}],

(*-----------------------------------*)
(*--- initialConditionsMacro macro --*)
(*-----------------------------------*)
(* Initial conditions for PDE4 are broken into 2 groups special
 initial conditions where one selects an IC like a step function, triangle, and such,
and there is another menu where one selects a function using its parameters *)
initialConditionsMacro = Item[Grid[{
    {TabView[{
       Text@Style["special function", 11] →

        myGrid[{
          {
           With[{plotOptions =
              {Ticks → None, ImageSize → 40, Filling → Bottom, PlotRange → All, ImagePadding → 1}},
            Grid[{
              {
               RadioButtonBar[Dynamic[choiceOfSpecialICfunction, {choiceOfSpecialICfunction = #;
                    initialConditionFunction = makeInitialConditionSpecial[choiceOfSpecialICfunction,
                      ICcenter, ICwidth, ICheight]; event = "reset"; gtick += delta} &],
                 {
                  0 → Plot[Evaluate@triangle[x, 1, 0, 1],
                    {x, -1.1, 1.1}, plotOptions, PlotLabel → Style["triangle", 9]],
                  1 → Plot[Evaluate@rectangle[x, 1, 0, 1], {x, -1.1, 1.1}, plotOptions,
                    PlotLabel → Style["rectangle", 9]],
                  2 → Plot[Evaluate@triangle[x, 1, 0, 1] * UnitStep[-x], {x, -1.1, 1.1},
                    plotOptions, PlotLabel → Style["triangle", 9]],
                  3 → Plot[Evaluate@triangle[x, 1, 0, 1] * UnitStep[x], {x, -1.1, 1.1},
                    plotOptions, PlotLabel → Style["triangle", 9]]
                 }, Appearance → "Row", ImageMargins → 0
               ]
              }
             }, Frame → True, FrameStyle -> Directive[Thickness[.005], Gray]
            ]
           ]
          },
          {
           Grid[{
             {
              Text@Style["center", 12],
              Manipulator[Dynamic[ICcenter, {ICcenter = #;
                   initialConditionFunction = makeInitialConditionSpecial[choiceOfSpecialICfunction,
                     ICcenter, ICwidth, ICheight]; event = "reset"; gtick += delta} &],
               {-.8, .8, .1}, ImageSize → Tiny, ContinuousAction -> False, Enabled →
                Dynamic[IntervalMemberQ[Interval@{0, 3}, choiceOfSpecialICfunction]]],
              Text@Style[Dynamic@padIt1[ICcenter, {4, 2}], 11],
              Spacer[2],
              Button[Text@Style["zero", 10], {ICcenter = 0.; event = "reset"; initialConditionFunction =
                  makeInitialConditionSpecial[choiceOfSpecialICfunction, ICcenter, ICwidth, ICheight];
                 gtick += delta}, ImageSize → {45, 20}, Alignment → Center],
              SpanFromLeft
             },
             {
```

```
    Text@Style["width", 12],
    Manipulator[Dynamic[ICwidth, {ICwidth = #;
        initialConditionFunction = makeInitialConditionSpecial[choiceOfSpecialICfunction,
          ICcenter, ICwidth, ICheight]; event = "reset"; gtick += delta} &],
     {0.01, 1, 0.01}, ImageSize → Tiny, ContinuousAction -> False, Enabled →
      Dynamic[IntervalMemberQ[Interval@{0, 3}, choiceOfSpecialICfunction]] ],
    Text@Style[Dynamic@padIt1[ICwidth, {4, 2}], 11],
    Spacer[2],
    Button[Text@Style["1/2", 10], {ICwidth = 1.; event = "reset"; initialConditionFunction =
       makeInitialConditionSpecial[choiceOfSpecialICfunction, ICcenter, ICwidth, ICheight];
      gtick += delta}, ImageSize → {45, 20}, Alignment → Center]
   },
   {
    Text@Style["height", 12],
    Manipulator[Dynamic[ICheight, {ICheight = #;
        initialConditionFunction = makeInitialConditionSpecial[choiceOfSpecialICfunction,
          ICcenter, ICwidth, ICheight]; event = "reset"; gtick += delta} &],
     {0, 5, 0.1}, ImageSize → Tiny, ContinuousAction -> False, Enabled →
      Dynamic[IntervalMemberQ[Interval@{0, 3}, choiceOfSpecialICfunction]] ],
    Text@Style[Dynamic@padIt1[ICheight, {4, 2}], 11],
    Spacer[2],
    Button[Text@Style["one", 10], {ICheight = 1.; event = "reset"; initialConditionFunction =
       makeInitialConditionSpecial[choiceOfSpecialICfunctionn, ICcenter, ICwidth,
        ICheight]; gtick += delta}, ImageSize → {45, 20}, Alignment → Center]
   }

  }, Alignment → Center, Frame → True,
  Spacings → {.2, .2}, FrameStyle -> Directive[Thickness[.005], Gray]
 ]
},
Dividers → {Thin, Blue},
{
 Grid[{
   {
    RadioButtonBar[Dynamic[choiceOfSpecialICfunction, {choiceOfSpecialICfunction = #;
        initialConditionFunction = makeInitialConditionSpecial[choiceOfSpecialICfunction,
          unitStepShift, unitStepHeight]; event = "reset"; gtick += delta} &],
     {
      4 → Plot[Evaluate@UnitStep[x - .5], {x, -1, 2}, Ticks → None, ImageSize → 50,
        Exclusions → None, PlotLabel → Style["step function", 10], Filling → Bottom],

      5 -> Plot[Evaluate@UnitStep[-.5 - x], {x, -2, 1}, Ticks → None, ImageSize → 50,
        Exclusions → None, PlotLabel → Style["step function", 10], Filling → Bottom]

     }, Appearance → "Row"
    ]
   }
  }, Frame → True, FrameStyle -> Directive[Thickness[.005], Gray]
 ]
},
{
 Grid[{
   {Text@Style["step function parameters", 11], SpanFromLeft},
   {
    Text@Style["shift amount", 12],
    Manipulator[Dynamic[unitStepShift, {unitStepShift = #;
        initialConditionFunction = makeInitialConditionSpecial[choiceOfSpecialICfunction,
          unitStepShift, unitStepHeight]; event = "reset"; gtick += delta} &],
     {-1., 1., .1}, ImageSize → Tiny, ContinuousAction -> False, Enabled →
      Dynamic[IntervalMemberQ[Interval@{4, 5}, choiceOfSpecialICfunction]]],
    Text@Style[Dynamic@padIt1[unitStepShift, {4, 2}], 11],
    Spacer[2],
    Button[Text@Style["zero", 10],
     {unitStepShift = 0.; event = "reset"; initialConditionFunction =
```

```
            makeInitialConditionSpecial[choiceOfSpecialICfunction, unitStepShift,
              unitStepHeight]; gtick += delta}, ImageSize → {45, 20}, Alignment → Center],
          SpanFromLeft
         },
         {
          Text@Style["height", 12],
          Manipulator[Dynamic[unitStepHeight, {unitStepHeight = #;
               initialConditionFunction = makeInitialConditionSpecial[choiceOfSpecialICfunction,
                 unitStepShift, unitStepHeight]; event = "reset"; gtick += delta} &],
           {0, 10, 0.1}, ImageSize → Tiny, ContinuousAction -> False, Enabled →
            Dynamic[IntervalMemberQ[Interval@{4, 5}, choiceOfSpecialICfunction]] ],
          Text@Style[Dynamic@padIt1[unitStepHeight, {4, 2}], 11],
          Spacer[2],
          Button[Text@Style["one", 10], {unitStepHeight = 1.;
            event = "reset"; initialConditionFunction = makeInitialConditionSpecial[
              choiceOfSpecialICfunction, unitStepShift, unitStepHeight];
            gtick += delta}, ImageSize → {45, 20}, Alignment → Center], SpanFromLeft
         }

       }, Alignment → Center, Frame → True,
       Spacings → {.1, .4}, FrameStyle -> Directive[Thickness[.005], Gray]
      ]
     }
    }, Alignment → Center, Spacings → {0, .6}
   ],

Text@Style["general", 11] → (*generalFunctionsPDE4*)
 Grid[{
   {PopupMenu[Dynamic[initialConditionsSelection, {initialConditionsSelection = #;
        initialConditionFunction =
         makeInitialCondition[initialConditionsSelection, ICa, ICb, ICc, ICd, stdx, x0];
        event = "reset";
        gtick += delta} &],
     {1 → Style[TraditionalForm[HoldForm[ξ]], 12],
      2 → Style[TraditionalForm[HoldForm[ξ x]], 12],
      3 → Style[TraditionalForm[HoldForm[ξ x + β x^2]], 12],
      4 → Style[TraditionalForm[HoldForm[ξ x + β x^2 + γ x^3]], 12],
      5 → Style[TraditionalForm[HoldForm[ξ x + β x^2 + γ x^3 + η x^4]], 12],
      6 → Style[TraditionalForm[HoldForm[ξ Sin[β x]]], 12],
      7 → Style[TraditionalForm[HoldForm[ξ Cos[β x]]], 12],
      8 → Style[TraditionalForm[HoldForm[ξ Sin[β x] + γ Sin[η x]]], 12],
      9 → Style[TraditionalForm[HoldForm[ξ Sin[β x] + γ Cos[η x]]], 12],
      10 → Style[TraditionalForm[HoldForm[ξ Cos[β x] + γ Cos[η x]]], 12],
      11 → Style[TraditionalForm[HoldForm[ξ (Sin[β x])^2]], 12],
      12 → Style[TraditionalForm[HoldForm[ξ (Cos[β x])^2]], 12],
      13 → Style[TraditionalForm[HoldForm[ξ Exp[β (x - η)^γ]]], 12],
      14 → Style[TraditionalForm[HoldForm[1/(σ Sqrt[2 π])] * HoldForm[Exp[-(x - μ)^2/(2 σ^2)]]], 12]
     }, ContinuousAction -> False], SpanFromLeft
   },
   {
    Grid[{
      {Text@Style[TraditionalForm[HoldForm[ξ], 12]], Spacer[2],
       Manipulator[Dynamic[ICa, {ICa = #; initialConditionFunction =
            makeInitialCondition[initialConditionsSelection, ICa, ICb, ICc, ICd, stdx, x0];
           event = "reset"; gtick += delta} &], {-10, 10, 1}, ImageSize → Small,
        ContinuousAction -> False, Enabled → Dynamic[Not[initialConditionsSelection == 14]]],
       Text@Style[Dynamic@padIt1[ICa, {4, 2}], 11],
       Spacer[10],
       Button[Text@Style["zero", 10], {ICa = 0.; event = "reset";
```

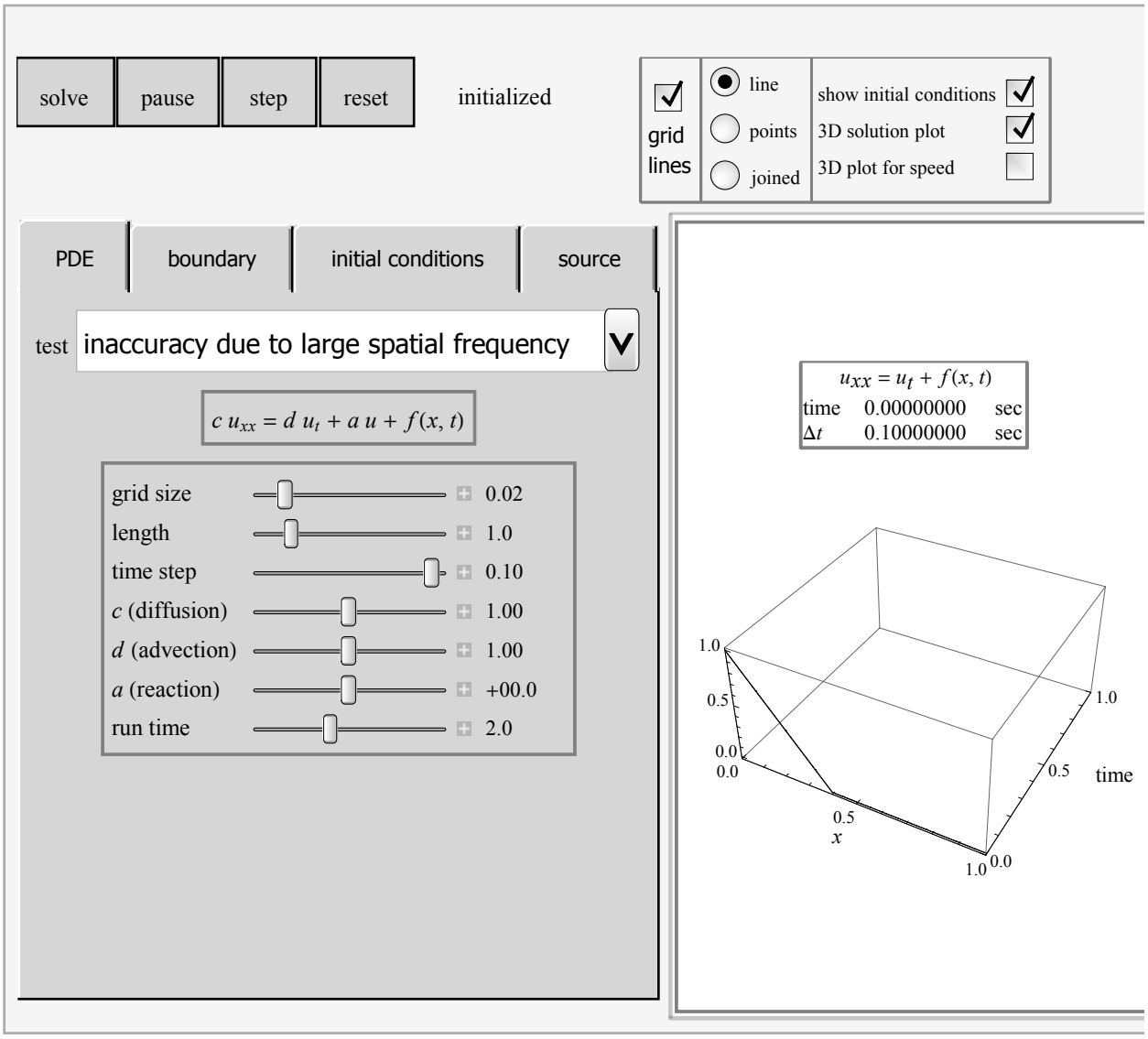
```
      initialConditionFunction = makeInitialCondition[initialConditionsSelection,
        ICa, ICb, ICc, ICd, stdx, x0]; gtick += delta}, ImageSize → {45, 20},
     Alignment → Center, Enabled → Dynamic[Not[initialConditionsSelection == 14]]],
    Spacer[2],
    Button[Text@Style["one", 10], {ICa = 1.; event = "reset";
      initialConditionFunction = makeInitialCondition[initialConditionsSelection,
        ICa, ICb, ICc, ICd, stdx, x0]; gtick += delta}, ImageSize → {45, 20},
     Alignment → Center, Enabled → Dynamic[Not[initialConditionsSelection == 14]]]
   }}, Spacings → {0, .5}], SpanFromLeft
},
{
 Grid[{
   {Text@Style[TraditionalForm[HoldForm[β], 12]], Spacer[2],
    Manipulator[Dynamic[ICb, {ICb = #; initialConditionFunction =
         makeInitialCondition[initialConditionsSelection, ICa, ICb, ICc, ICd, stdx, x0];
        event = "reset"; gtick += delta} &], {-10, 10, 1}, ImageSize → Small,
     ContinuousAction -> False, Enabled → Dynamic[Not[initialConditionsSelection == 14]]],
    Text@Style[Dynamic@padIt1[ICb, {4, 2}], 11],
    Spacer[10],
    Button[Text@Style["zero", 10], {ICb = 0.; event = "reset";
      initialConditionFunction = makeInitialCondition[initialConditionsSelection,
        ICa, ICb, ICc, ICd, stdx, x0]; gtick += delta}, ImageSize → {45, 20},
     Alignment → Center, Enabled → Dynamic[Not[initialConditionsSelection == 14]]],
    Spacer[2],
    Button[Text@Style["one", 10], {ICb = 1.; event = "reset";
      initialConditionFunction = makeInitialCondition[initialConditionsSelection,
        ICa, ICb, ICc, ICd, stdx, x0]; gtick += delta}, ImageSize → {45, 20},
     Alignment → Center, Enabled → Dynamic[Not[initialConditionsSelection == 14]]]
   }}, Spacings → {0, .5}], SpanFromLeft
},
{
 Grid[{
   {Text@Style[TraditionalForm[HoldForm[γ], 12]], Spacer[2],
    Manipulator[Dynamic[ICc, {ICc = #; initialConditionFunction =
         makeInitialCondition[initialConditionsSelection, ICa, ICb, ICc, ICd, stdx, x0];
        event = "reset"; gtick += delta} &], {0, 5, .1}, ImageSize → Small,
     ContinuousAction -> False, Enabled → Dynamic[Not[initialConditionsSelection == 14]]],
    Text@Style[Dynamic@padIt1[ICc, {4, 2}], 11],
    Spacer[10],
    Button[Text@Style["zero", 10], {ICc = 0.; event = "reset";
      initialConditionFunction = makeInitialCondition[initialConditionsSelection,
        ICa, ICb, ICc, ICd, stdx, x0]; gtick += delta}, ImageSize → {45, 20},
     Alignment → Center, Enabled → Dynamic[Not[initialConditionsSelection == 14]]],
    Spacer[2],
    Button[Text@Style["one", 10], {ICc = 1.0; event = "reset";
      initialConditionFunction = makeInitialCondition[initialConditionsSelection,
        ICa, ICb, ICc, ICd, stdx, x0]; gtick += delta}, ImageSize → {45, 20},
     Alignment → Center, Enabled → Dynamic[Not[initialConditionsSelection == 14]]]
   }}, Spacings → {0, .5}], SpanFromLeft
},
{
 Grid[{
   {Text@Style[TraditionalForm[HoldForm[η], 12]], Spacer[2],
    Manipulator[Dynamic[ICd, {ICd = #; initialConditionFunction =
         makeInitialCondition[initialConditionsSelection, ICa, ICb, ICc, ICd, stdx, x0];
        event = "reset"; gtick += delta} &], {-20, 20, .1}, ImageSize → Small,
     ContinuousAction -> False, Enabled → Dynamic[Not[initialConditionsSelection == 14]]],
    Text@Style[Dynamic@padIt1[ICd, {4, 2}], 11],
    Spacer[10],
    Button[Text@Style["zero", 10], {ICd = 0.; event = "reset";
      initialConditionFunction = makeInitialCondition[initialConditionsSelection,
        ICa, ICb, ICc, ICd, stdx, x0]; gtick += delta}, ImageSize → {45, 20},
     Alignment → Center, Enabled → Dynamic[Not[initialConditionsSelection == 14]]],
    Spacer[2],
```

```
    Button[Text@Style["one", 10], {ICd = 1.0; event = "reset";
      initialConditionFunction = makeInitialCondition[initialConditionsSelection,
        ICa, ICb, ICc, ICd, stdx, x0]; gtick += delta}, ImageSize → {45, 20},
     Alignment → Center, Enabled → Dynamic[Not[initialConditionsSelection == 14]]]
   }}, Spacings → {0, .5}], SpanFromLeft
},
{
 Grid[{
   {Text@Style[TraditionalForm[HoldForm[σ], 12]], Spacer[2],
    Manipulator[Dynamic[stdx, {stdx = #; initialConditionFunction =
         makeInitialCondition[initialConditionsSelection, ICa, ICb, ICc, ICd, stdx, x0];
        event = "reset"; gtick += delta} &], {0.01, 2.0, .01}, ImageSize → Small,
     ContinuousAction -> False, Enabled → Dynamic[initialConditionsSelection == 14]],
    Text@Style[Dynamic@padIt1[stdx, {4, 2}], 11],
    Spacer[10],
    Button[Text@Style["one", 10], {stdx = 1.0; event = "reset";
      initialConditionFunction = makeInitialCondition[initialConditionsSelection,
        ICa, ICb, ICc, ICd, stdx, x0]; gtick += delta}, ImageSize → {45, 20},
     Alignment → Center, Enabled → Dynamic[initialConditionsSelection == 14]],
    Spacer[2],
    Button[Text@Style["0.5", 10], {stdx = 0.5; event = "reset";
      initialConditionFunction = makeInitialCondition[initialConditionsSelection,
        ICa, ICb, ICc, ICd, stdx, x0]; gtick += delta}, ImageSize → {45, 20},
     Alignment → Center, Enabled → Dynamic[initialConditionsSelection == 14]]
   }}, Spacings → {0, .5}], SpanFromLeft
},
{
 Grid[{
   {Text@Style[TraditionalForm[HoldForm[μ], 12]], Spacer[2],
    Manipulator[Dynamic[x0, {x0 = #; initialConditionFunction =
         makeInitialCondition[initialConditionsSelection, ICa, ICb, ICc, ICd, stdx, x0];
        event = "reset"; gtick += delta} &], {-1.5, 1.5, .1}, ImageSize → Small,
     ContinuousAction -> False, Enabled → Dynamic[initialConditionsSelection == 14]],
    Text@Style[Dynamic@padIt1[x0, {4, 2}], 11],
    Spacer[10],
    Button[Text@Style["zero", 10], {x0 = 0.; event = "reset";
      initialConditionFunction = makeInitialCondition[initialConditionsSelection,
        ICa, ICb, ICc, ICd, stdx, x0]; gtick += delta}, ImageSize → {45, 20},
     Alignment → Center, Enabled → Dynamic[initialConditionsSelection == 14]],
    Spacer[2],
    Button[Text@Style["0.5", 10], {x0 = 0.5; event = "reset";
      initialConditionFunction = makeInitialCondition[initialConditionsSelection,
        ICa, ICb, ICc, ICd, stdx, x0]; gtick += delta}, ImageSize → {45, 20},
     Alignment → Center, Enabled → Dynamic[initialConditionsSelection == 14]]
   }}, Spacings → {0, .5}], SpanFromLeft
},
{
 Dynamic@Grid[{
    {
     Block[{from, to, f, plotLength = 100 (*115*)},
      If[centerGrid,
       (
        from = -length / 2;
        to = length / 2
       ),
       (
        from = 0;
        to = length
       )
      ];
      f = Evaluate@
        makeInitialCondition[initialConditionsSelection, ICa, ICb, ICc, ICd, stdx, x0][x];
      Plot[f, {x, from, to},
       ImagePadding → {{40, 10}, {20, 30}},
```

```
                    ImageMargins → 0,
                    PlotRange → All,
                    Frame → True,
                    Axes → None,
                    Exclusions → None,
                    FrameLabel → {{None, None}, {None, Text@Row[{Style[Row[{Style["u", Italic], "(",
                              Style["x", Italic], ", ", 0, ") = "}], 11], Spacer[4], f}]}},
                    ImageSize → {260, plotLength},
                    TicksStyle → 9,
                    AspectRatio → 0.25,
                    PlotStyle → Red,
                    Evaluate@If[addGrid, GridLines → Automatic, GridLines → None]
                   ]
                  ]
                 }}, Spacings → {0, 0}, Frame → True,
                FrameStyle -> Directive[Thickness[.005], Gray]], SpanFromLeft
              }
             }, Spacings → {.25, .4},
             Alignment → Center, Frame → None, FrameStyle -> Directive[Thickness[.005], Gray]
            ]

          }, Alignment → {Center, Top}
         ]
        }}
     ], Alignment → {Center, Top}]
  },
  (*------------------------------*)
  (*--- LEVEL 2 -----*)
  (*------------------------------*)
  With[{
    pde4 = Grid[{
        {TabView[{
           Style["PDE", 11] → parametersMacro,
           Style["boundary", 11] → geometryMacro,
           Style["initial conditions", 11] → initialConditionsMacro,
           Style["source", 11] → sourceMacro
          }, ImageSize → {290, 400}]
        }
      }, Spacings → {0.2, .9}
     ]
   },
   (*--- end of level 2 ---*)

   ## &[
    Item[
     Grid[{
       {topRowMacro, plotOptionsMacro}
      }, Spacings → {1.7, 0}, Alignment → {Center, Top}
     ], ControlPlacement → Top
    ],
    Item[pde4, ControlPlacement → Left]
   ]

  ]
 ],
(*----------- end of Manipulate controls --------------------------*)

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

```
{{threeDviewSpeed, False}, None},
{{choiceOfSpecialICfunction, 5}, None},
{{ICheight, 1}, None}, (* height of special initial condition, such as triangle, rectangle. *)
{{ICwidth, .4}, None}, (* height of special initial condition,
triangle rectangle. not used for unit step*)
{{ICcenter, 0}, None},
{{unitStepShift, .2}, None},
{{unitStepHeight, 1}, None},
{{threeDView, True}, None},
{{animation3dBuffer, {0, {}}}, None},
{{finalDisplayImage, {}}, None},
{{testCase, 5}, None},
{{yscaleAuto, True}, None},
{{yscaleAmount, 1.1}, None},
{{eastbc, 1}, None},
{{westbc, 1}, None},
{{westBCtype, "Dirichlet"}, None},
{{awestBCconstantValue, 1}, None},
{{bwestBCconstantValue, 0}, None},
{{eastBCtype, "Dirichlet"}, None},
{{aeastBCconstantValue, 1}, None},
{{beastBCconstantValue, 0}, None},
{{eastBoundaryFunction, Function[{t}, 0]}, None},
{{westBoundaryFunction, Function[{t}, 0]}, None},
{{sourceDurationType, "simulation duration"}, None},
{{sourceDuration, 0.01}, None},
{{joinedType, "line"}, None},
{{addGrid, True}, None},
{{initialConditionsSelection, 10}, None},
{{initialConditionFunction, Function[{x}, Cos[2 x]]}, None},
{{ICa, 1.}, None},
{{ICb, 2.}, None},
{{ICc, 1.}, None},
{{ICd, 1.}, None},
{{Sa, 0}, None},
{{Sb, 0}, None},
{{Sc, 1}, None},
{{Sd, 0}, None},
{{stepNumber, 0}, None},
{{cpuTimeUsed, 0}, None},
{{currentTime, 0}, None},
{{Aleft, {}}, None},
{{Aright, {}}, None},
{{centerGrid, True}, None},
{{h, 0.03}, None},
{{length, 1}, None},
{{k, 0.01}, None},
{{aReactionTerm, 0.}, None},
{{dAdvectionTerm, 1.}, None},
{{cDiffusionTerm, 0.1}, None},
{{maxTime, 1}, None},
{{isPeriodicBC, False}, None},
{{forceTermSelection, 1}, None},
{{grid, generatePhysicalCoordinates1D[0.25, 1, True]}, None},
{{u, {}}, None},
{{u0, {}}, None},
{{sourceFunction, Function[{x, t}, 0]}, None},
{{state, "INIT"}, None},
{{event, "reset"}, None},
{{stdx, 0.2}, None},
{{x0, 0}, None},
{{showIC, True}, None},
```

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

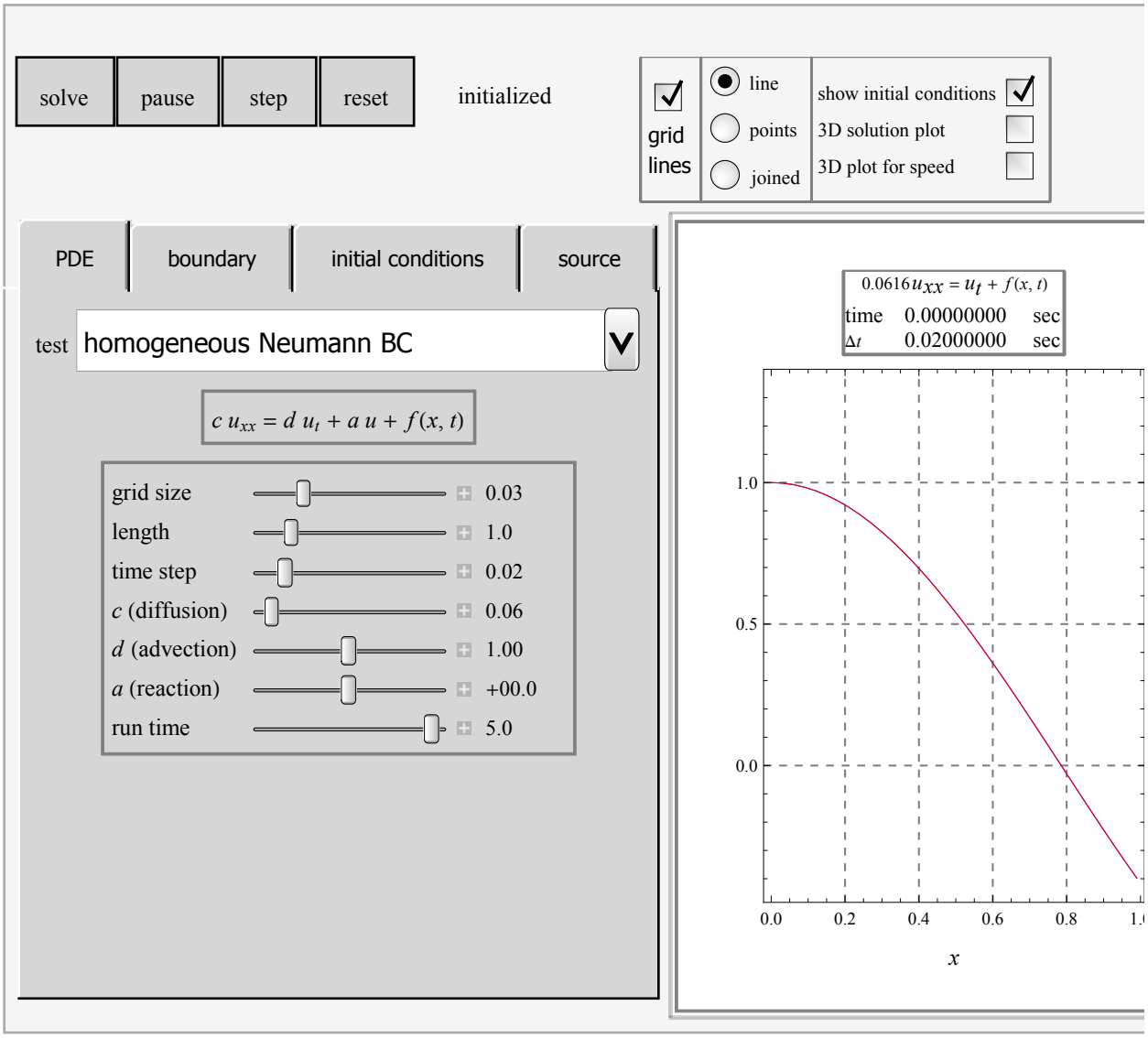


Caption

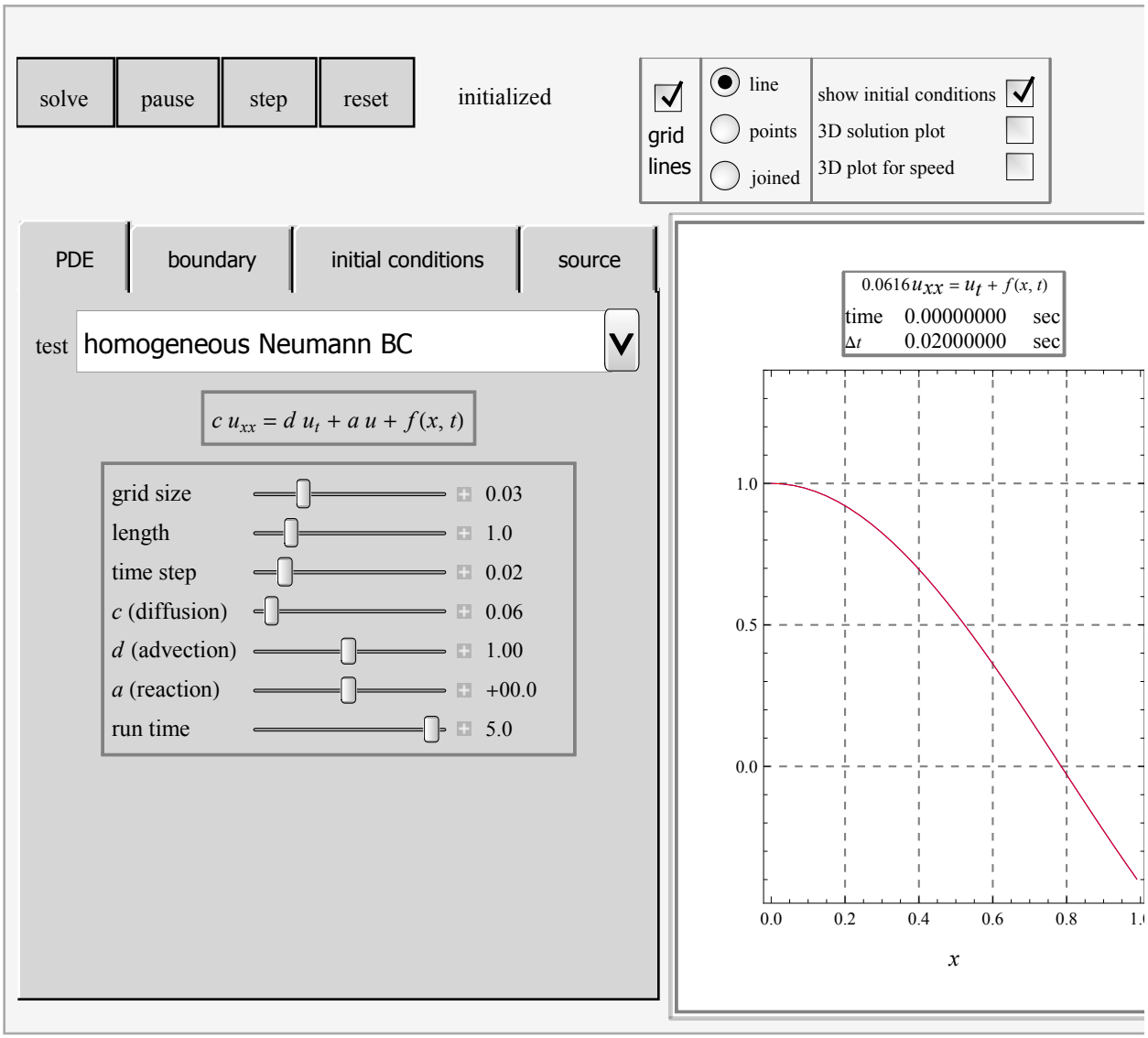
This Demonstration solves the diffusion-advection-reaction partial differential equation (PDE) $c\,u_{xx} = d\,u_t + a\,u + f(x,t)$ in one dimension. The domain is discretized in space and for each time step the solution u at time t_{n+1} is found by solving for u^{n+1} from $A\,u^{n+1} = B\,u^n + \frac{f^n + f^{n+1}}{2}$. The boundary conditions supported are periodic, Dirichlet and Neumann. The solution can be viewed in 3D as well as in 2D. You can select the source term $f(x,t)$ and the initial conditions from the menus in the main display. Selected

preconfigured test cases are available from the pull down menu. In the above PDE $c\,u_{xx}$ represents the diffusion, $d\,u_t$ represents the advection and $a\,u$ the reaction.

Thumbnail



Snapshots



Details (optional)

The discretized system was formulated by the matrix equation $A\,u^{n+1} = B\,u^n + b$. The matrices A and B and the vector b take different forms depending on the boundary conditions. The following equations shows the full expression of the above equation for the five possible combinations of the boundary conditions: periodic boundary conditions, Dirichlet on both sides of the domain, Dirichlet on the left side and Neumann on the right side, Neumann on the left side and Dirichlet on the right side, and finally Neumann on both sides. The following four variables are used to simplify the writing of expressions shown below in the matrix equation

$$r_1 = \frac{a}{2}\frac{\Delta t}{d} + \frac{c}{d}\frac{\Delta t}{h^2} + 1 \qquad r_2 = \frac{c}{2\,d}\frac{\Delta t}{h^2} \qquad r_3 = -\frac{a}{2}\frac{\Delta t}{d} - \frac{c}{d}\frac{\Delta t}{h^2} + 1 \qquad r_4 = \frac{\Delta t}{2\,d}$$

The left boundary condition is given by $\alpha(t)$ and the right boundary conditions by β(t). For example, if the left side has Dirichlet boundary conditions then $u_1 = \alpha(t)$, while if it has Neumann boundary conditions then $\frac{du}{dx} = \alpha(t)$, similarly for the right side. For periodic boundary conditions $u_1 = u_N$ where u_1 is the first node on the left side and u_N is the last node on the right side. Δt is the time step and h is the grid spacing. a, c and d are the parameters of the PDE itself taken from $c\,u_{xx} = d\,u_t + a\,u + f(x,t)$

periodic boundary conditions

$$\begin{pmatrix} r_1 & -r_2 & 0 & 0 & -r_2 \\ -r_2 & r_1 & -r_2 & 0 & 0 \\ 0 & -r_2 & r_1 & -r_2 & 0 \\ 0 & 0 & -r_2 & r_1 & 0 \\ -r_2 & 0 & 0 & 0 & r_1 \end{pmatrix} \begin{pmatrix} u_2^{n+1} \\ u_3^{n+1} \\ \vdots \\ u_{N-1}^{n+1} \\ u_N^{n+1} \end{pmatrix} = \begin{pmatrix} r_3 & r_2 & 0 & 0 & r_2 \\ r_2 & r_3 & r_2 & 0 & 0 \\ 0 & r_2 & r_3 & r_2 & 0 \\ 0 & 0 & r_2 & r_3 & 0 \\ r_2 & 0 & 0 & 0 & r_3 \end{pmatrix} \begin{pmatrix} u_2^n \\ u_3^n \\ \vdots \\ u_{N-1}^n \\ u_N^n \end{pmatrix} + \begin{pmatrix} r_4 \left(f_2^n + f_2^{n+1} \right) \\ r_4 \left(f_3^n + f_3^{n+1} \right) \\ \vdots \\ r_4 \left(f_{N-1}^n + f_{N-1}^{n+1} \right) \\ r_4 \left(f_N^n + f_N^{n+1} \right) \end{pmatrix}$$

$$u_1^{n+1} = u_N^{n+1}$$

Dirichlet boundary conditions on both ends

$$\begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & r_1 & -r_2 & 0 & 0 & 0 \\ 0 & -r_2 & r_1 & -r_2 & 0 & 0 \\ 0 & 0 & -r_2 & r_1 & -r_2 & 0 \\ 0 & 0 & 0 & -r_2 & r_1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} u_1^{n+1} \\ u_2^{n+1} \\ u_3^{n+1} \\ \vdots \\ u_{N-1}^{n+1} \\ u_N^{n+1} \end{pmatrix} = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & r_3 & r_2 & 0 & 0 & 0 \\ 0 & r_2 & r_3 & r_2 & 0 & 0 \\ 0 & 0 & r_2 & r_3 & r_2 & 0 \\ 0 & 0 & 0 & r_2 & r_3 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix} \begin{pmatrix} u_1^n \\ u_2^n \\ u_3^n \\ \vdots \\ u_{N-1}^n \\ u_N^n \end{pmatrix} + \begin{pmatrix} \alpha^{n+1} \\ r_4 \left(f_2^n + f_2^{n+1} \right) + r_2 \left(\alpha^{n+1} + \alpha^n \right) \\ r_4 \left(f_3^n + f_3^{n+1} \right) \\ \vdots \\ r_4 \left(f_{N-1}^n + f_{N-1}^{n+1} \right) + r_2 \left(\alpha^{n+1} + \alpha^n \right) \\ \beta^{n+1} \end{pmatrix}$$

Dirichlet boundary conditions on left side and Neumann boundary conditions on the right side

$$\begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & r_1 & -r_2 & 0 & 0 & 0 \\ 0 & -r_2 & r_1 & -r_2 & 0 & 0 \\ 0 & 0 & -r_2 & r_1 & -r_2 & 0 \\ 0 & 0 & 0 & -r_2 & r_1 & -r_2 \\ 0 & 0 & 0 & 0 & -2\,r_2 & r_1 \end{pmatrix} \begin{pmatrix} u_1^{n+1} \\ u_2^{n+1} \\ u_3^{n+1} \\ \vdots \\ u_{N-1}^{n+1} \\ u_N^{n+1} \end{pmatrix} = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & r_3 & r_2 & 0 & 0 & 0 \\ 0 & r_2 & r_3 & r_2 & 0 & 0 \\ 0 & 0 & r_2 & r_3 & r_2 & 0 \\ 0 & 0 & 0 & r_2 & r_3 & r_2 \\ 0 & 0 & 0 & 0 & 2\,r_2 & r_3 \end{pmatrix} \begin{pmatrix} u_1^n \\ u_2^n \\ u_3^n \\ \vdots \\ u_{N-1}^n \\ u_N^n \end{pmatrix} + \begin{pmatrix} \alpha^{n+1} \\ r_4 \left(f_2^n + f_2^{n+1} \right) + r_2\, \alpha^n \\ r_4 \left(f_3^n + f_3^{n+1} \right) \\ \vdots \\ r_4 \left(f_{N-1}^n + f_{N-1}^{n+1} \right) \\ 2\,h\,r_2 \left(\beta^{n+1} + \beta^n \right) \end{pmatrix}$$

Dirichlet boundary conditions on right side and Neumann boundary conditions on the left side

$$\begin{pmatrix} r_1 & -2\,r_2 & 0 & 0 & 0 & 0 \\ -r_2 & r_1 & -r_2 & 0 & 0 & 0 \\ 0 & -r_2 & r_1 & -r_2 & 0 & 0 \\ 0 & 0 & -r_2 & r_1 & -r_2 & 0 \\ 0 & 0 & 0 & -r_2 & r_1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} u_1^{n+1} \\ u_2^{n+1} \\ u_3^{n+1} \\ \vdots \\ u_{N-1}^{n+1} \\ u_N^{n+1} \end{pmatrix} = \begin{pmatrix} r_3 & 2\,r_2 & 0 & 0 & 0 & 0 \\ r_2 & r_3 & r_2 & 0 & 0 & 0 \\ 0 & r_2 & r_3 & r_2 & 0 & 0 \\ 0 & 0 & r_2 & r_3 & r_2 & 0 \\ 0 & 0 & 0 & r_2 & r_3 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix} \begin{pmatrix} u_1^n \\ u_2^n \\ u_3^n \\ \vdots \\ u_{N-1}^n \\ u_N^n \end{pmatrix} + \begin{pmatrix} 2\,h\,r_2 \left(\alpha^{n+1} + \alpha^n \right) \\ r_4 \left(f_2^n + f_2^{n+1} \right) \\ \vdots \\ r_4 \left(f_{N-2}^n + f_{N-2}^{n+1} \right) \\ r_4 \left(f_{N-1}^n + f_{N-1}^{n+1} \right) + r_2\, \beta^n \\ \beta^n \end{pmatrix}$$

Neumann boundary conditions on the both sides

$$\begin{pmatrix} r_1 & -2\,r_2 & 0 & 0 & 0 & 0 \\ -r_2 & r_1 & -r_2 & 0 & 0 & 0 \\ 0 & -r_2 & r_1 & -r_2 & 0 & 0 \\ 0 & 0 & -r_2 & r_1 & -r_2 & 0 \\ 0 & 0 & 0 & -r_2 & r_1 & -r_2 \\ 0 & 0 & 0 & 0 & -2\,r_2 & r_1 \end{pmatrix} \begin{pmatrix} u_1^{n+1} \\ u_2^{n+1} \\ u_3^{n+1} \\ \vdots \\ u_{N-1}^{n+1} \\ u_N^{n+1} \end{pmatrix} = \begin{pmatrix} r_3 & 2\,r_2 & 0 & 0 & 0 & 0 \\ r_2 & r_3 & r_2 & 0 & 0 & 0 \\ 0 & r_2 & r_3 & r_2 & 0 & 0 \\ 0 & 0 & r_2 & r_3 & r_2 & 0 \\ 0 & 0 & 0 & r_2 & r_3 & r_2 \\ 0 & 0 & 0 & 0 & 2\,r_2 & r_3 \end{pmatrix} \begin{pmatrix} u_1^n \\ u_2^n \\ u_3^n \\ \vdots \\ u_{N-1}^n \\ u_N^n \end{pmatrix} + \begin{pmatrix} 2\,h\,r_2 \left(\alpha^{n+1} + \alpha^n \right) \\ r_4 \left(f_2^n + f_2^{n+1} \right) \\ \vdots \\ r_4 \left(f_{N-2}^n + f_{N-2}^{n+1} \right) \\ r_4 \left(f_{N-1}^n + f_{N-1}^{n+1} \right) \\ 2\,h\,r_2 \left(\beta^{n+1} + \beta^n \right) \end{pmatrix}$$

Control Suggestions (optional)

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

Bookmark Animation

Search Terms (optional)

heat equation
diffusion
advection
reaction
finite difference method

Related Links (optional)

Heat Conduction Equation

Authoring Information

Contributed by: Nasser M. Abbasi