Finite Difference Solution of the Convection-Diffusion Equation in 1D

Initialization Code (optional)

Manipulate

```
Manipulate[
 gtick;

 {finalDisplayImage, u, u0, grid, systemMatrix, stepNumber, cpuTimeUsed, currentTime, state} =
  process[u, grid, systemMatrix, stepNumber, cpuTimeUsed, currentTime, state, u0,
   Unevaluated[animation3dBuffer], initialConditionFunction, event, h, centerGrid,
   length, k*h^2/cDiffusionTerm, aConvectionTerm, dAdvectionTerm, cDiffusionTerm,
   maxTime, addGrid, showIC, joinedType, 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]]
          },
          {Text@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 → Left, 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 -> {100, 30}, ImagePadding → {{70, 75}, {75, 75}}], SpanFromLeft
    }}, Spacings → {0.1, 0}
  ], Alignment → {Center, Top}
 ],
(*--------------------------*)
(*--- geometryMacro macro  --*)
(*--------------------------*)
geometryMacro = Item[Grid[{
    {
     Grid[{
       {
        Text@Style["test case", 12], PopupMenu[Dynamic[testCase, {testCase = #;
             Which[testCase == 1,
              (
               threeDViewSpeed = False;
               threeDView = False;
               h = 0.05;
               length = 3;
               k = 0.25;
               aConvectionTerm = 9.;
               dAdvectionTerm = 1.;
               cDiffusionTerm = 0.4;
               maxTime = 0.04;
               centerGrid = False;

               initialConditionsSelection = 13;
               ICa = 1.;
               ICb = -8;
               ICc = 2;
               ICd = 0.5;

               initialConditionFunction =
                makeInitialConditions[initialConditionsSelection, ICa, ICb, ICc, ICd, stdx, x0];

               addGrid = True;
               joinedType = "line";
               showIC = True;
               yscaleAuto = True;
               showIC = True

              ), testCase == 2,
              (
               threeDViewSpeed = False;
               threeDView = False;
               h = 0.05;
               length = 3;
               k = 0.1;
               aConvectionTerm = 9.;
               dAdvectionTerm = 1.;
               cDiffusionTerm = 0.4;
               maxTime = 0.1;
               centerGrid = True;
```

```
            initialConditionsSelection = 14;
            stdx = .08;
            x0 = -1.;

            initialConditionFunction =
             makeInitialConditions[initialConditionsSelection, ICa, ICb, ICc, ICd, stdx, x0];

            addGrid = True;
            joinedType = "line";
            showIC = True;
            yscaleAuto = True;
            showIC = True

           )
          ]; event = "reset"; gtick += delta} &],
      { 1 → Text@Style["1", 11],
       2 → Text@Style["2", 11]
      }, ImageSize -> All, ContinuousAction -> False]
    }
   }
  ]
},
{
 Framed[Text@Row[{Style["c", Italic], " ", Style["u", Italic]Style["x",Italic]Style["x",Italic], " = ",
     Style["d", Italic] , " ", Style["u", Italic]Style["t",Italic], " + ", Style["a", Italic],
     " ", Style["u", Italic]Style["x",Italic]}], FrameStyle -> Directive[Thickness[.005], Gray]]
},
{
 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, 3, 0.01}, ImageSize → Small, ContinuousAction -> False],
     Spacer[3],
     Text@Style[Dynamic@padIt2[length, {3, 2}], 11]
    },
    {
     Text@Style[Row[{"Δ", Style["t", Italic], " multiplier"}], 12],
     Spacer[3],
     Manipulator[Dynamic[k, {k = #; event = "reset"; gtick += delta} &],
      {0.05, 2, 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} &],
      {0.01, 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], " (convection)"}], 12],
        Spacer[3],
        Manipulator[Dynamic[aConvectionTerm, {aConvectionTerm = #; event = "reset"; gtick += delta} &],
         {0.01, 9, 0.01}, ImageSize → Small, ContinuousAction -> False],
        Spacer[3],
        Text@Style[Dynamic@padIt2[aConvectionTerm, {3, 2}], 11]
       },
       {
        Text@Style["run time", 12],
        Spacer[3],
        Manipulator[Dynamic[maxTime, {maxTime = #; event = "reset"; gtick += delta} &],
         {0.01, 0.1, 0.01}, ImageSize → Small, ContinuousAction -> False],
        Spacer[3],
        Text@Style[Dynamic@padIt2[maxTime, {3, 2}], 11],
        Spacer[10],
        SpanFromLeft
       }
      }, Spacings → {0.1, 0.1}, Alignment → Left, Frame → None
     ], FrameStyle -> Directive[Thickness[.005], Gray]
    ]
   },
   {
    Grid[{
      {
       Text@Style["centered grid ", 12],
       Checkbox[Dynamic[centerGrid, {centerGrid = #; event = "reset"; gtick += delta} &]]
      },
      {
       Text@Style[Row[{"auto ", Style["y", Italic], " scale "}], 12],
       Checkbox[Dynamic[yscaleAuto, {yscaleAuto = #; event = "plot_changed" , gtick += delta} &],
        Enabled → Dynamic[threeDView == False]],
       Spacer[5],
       Text@Style["manual", 12],
       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, {2, 1}], 10],
       SpanFromLeft
      }
     }, Alignment → Center, Frame → True, FrameStyle -> Directive[Thickness[.005], Gray]
    ]
   }
  }, Alignment → Center, Spacings → {0, .8}
 ], Alignment → {Center, Top}],
(*-----------------------------------*)
(*--- initialConditionsMacro macro --*)
(*-----------------------------------*)
(* Initial conditions for PDE are broken into 2 groups special
 initial conditions where one selects an IC like a step function or triangle
 and there is another menu where one selects a function using its parameters *)

initialConditionsMacro = Grid[{
  {TabView[{
     Text@Style["special function", 11] →
```

```
myGrid[{
  {
   Grid[{
     {
      RadioButtonBar[Dynamic[choiceOfSpecialICfunction, {choiceOfSpecialICfunction = #;
           initialConditionFunction = makeInitialConditionsSpecial[choiceOfSpecialICfunction,
             ICcenter, ICwidth, ICheight]; event = "reset"; gtick += delta} &],
       {
        0 → Plot[Evaluate@triangle[x, 1, 0, 1], {x, -1.1, 1.1}, Ticks → None,
          ImageSize → 50, PlotLabel → Text@Style["triangle", 10], Filling → Bottom
         ],

        1 → Plot[Evaluate@rectangle[x, 1, 0, 1], {x, -1.1, 1.1}, Ticks → None, ImageSize → 50,
          Exclusions → None, PlotLabel → Text@Style["rectangle", 10], Filling → Bottom],

        2 → Plot[Evaluate@triangle[x, 1, 0, 1] * UnitStep[-x],
          {x, -1.1, 1.1}, Ticks → None, ImageSize → 50, PlotLabel →
           Text@Style["half triangle", 10], Filling → Bottom, PlotRange → All],

        3 → Plot[Evaluate@triangle[x, 1, 0, 1] * UnitStep[x],
          {x, -1.1, 1.1}, Ticks → None, ImageSize → 50, PlotLabel →
           Text@Style["half triangle", 10], Filling → Bottom, PlotRange → All]
       }, Appearance → "Row"
      ]
     }
    }, Frame → True, FrameStyle -> Directive[Thickness[.005], Gray]
   ]
  },
  {
   Framed[Grid[{
       {
        Text@Style["center", 11],
        Spacer[2],
        Manipulator[Dynamic[ICcenter, {ICcenter = #;
             initialConditionFunction = makeInitialConditionsSpecial[choiceOfSpecialICfunction,
               ICcenter, ICwidth, ICheight]; event = "reset"; gtick += delta} &],
         {-2, 2, .01}, ImageSize → Small, ContinuousAction -> False, Enabled →
          Dynamic[IntervalMemberQ[Interval@{0, 3}, choiceOfSpecialICfunction]]],
        Text@Style[Dynamic@padIt1[ICcenter, {3, 2}], 11],
        Spacer[2],
        Button[Text@Style["zero", 10], {ICcenter = 0.; event = "reset"; initialConditionFunction =
           makeInitialConditionsSpecial[choiceOfSpecialICfunction, ICcenter, ICwidth,
            ICheight]; gtick += delta}, ImageSize → {45, 20}, Alignment → Center,
         Enabled → Dynamic[IntervalMemberQ[Interval@{0, 3}, choiceOfSpecialICfunction]]],
        SpanFromLeft
       },
       {
        Text@Style["width", 11],
        Spacer[2],
        Manipulator[Dynamic[ICwidth, {ICwidth = #;
             initialConditionFunction = makeInitialConditionsSpecial[choiceOfSpecialICfunction,
               ICcenter, ICwidth, ICheight]; event = "reset"; gtick += delta} &],
         {0.01, 2, 0.01}, ImageSize → Small, ContinuousAction -> False, Enabled →
          Dynamic[IntervalMemberQ[Interval@{0, 3}, choiceOfSpecialICfunction]] ],
        Text@Style[Dynamic@padIt2[ICwidth, {3, 2}], 11],
        Spacer[3],
        Button[Text@Style["0.5", 10], {ICwidth = 0.5; event = "reset"; initialConditionFunction =
           makeInitialConditionsSpecial[choiceOfSpecialICfunction, ICcenter, ICwidth,
            ICheight]; gtick += delta}, ImageSize → {45, 20}, Alignment → Center,
         Enabled → Dynamic[IntervalMemberQ[Interval@{0, 3}, choiceOfSpecialICfunction]]]
       },
       {
        Text@Style["height", 11],
        Spacer[2],
```

```
      Manipulator[Dynamic[ICheight, {ICheight = #;
           initialConditionFunction = makeInitialConditionsSpecial[choiceOfSpecialICfunction,
             ICcenter, ICwidth, ICheight]; event = "reset"; gtick += delta} &],
        {0, 3, 0.01}, ImageSize → Small, ContinuousAction -> False, Enabled →
         Dynamic[IntervalMemberQ[Interval@{0, 3}, choiceOfSpecialICfunction]] ],
      Text@Style[Dynamic@padIt2[ICheight, {3, 2}], 11],
      Spacer[3],
      Button[Text@Style["one", 10], {ICheight = 1.; event = "reset"; initialConditionFunction =
          makeInitialConditionsSpecial[choiceOfSpecialICfunctionn, ICcenter, ICwidth,
           ICheight]; gtick += delta}, ImageSize → {45, 20}, Alignment → Center,
        Enabled → Dynamic[IntervalMemberQ[Interval@{0, 3}, choiceOfSpecialICfunction]]]
      }

    }, Alignment → Center, Frame → None, Spacings → {0, .2}
   ], FrameStyle -> Directive[Thickness[.005], Gray]
  ]
 },
 Dividers → {Thin, Blue},
 {
  Grid[{
    {
     RadioButtonBar[Dynamic[choiceOfSpecialICfunction, {choiceOfSpecialICfunction = #;
          initialConditionFunction = makeInitialConditionsSpecial[choiceOfSpecialICfunction,
            ICunitStepShift, ICunitStepHeight]; event = "reset"; gtick += delta} &],
       {
        4 → Plot[Evaluate@UnitStep[x - .5], {x, -1, 2}, Ticks → None, ImageSize → 50, Exclusions →
           None, PlotLabel → Text@Style["  step function  ", 10], Filling → Bottom],

        5 -> Plot[Evaluate@UnitStep[-.5 - x], {x, -2, 1}, Ticks → None, ImageSize → 50,
          Exclusions → None, PlotLabel → Text@Style["  step function  ", 10], Filling → Bottom]
       }, Appearance → "Row"
      ]
    }
   }, Frame → True, FrameStyle -> Directive[Thickness[.005], Gray]
  ]
 },
 {
  Framed[Grid[{
     {Text@Style["step function parameters", 11], SpanFromLeft},
     {
      Text@Style["shift", 11],
      Manipulator[Dynamic[ICunitStepShift, {ICunitStepShift = #;
           initialConditionFunction = makeInitialConditionsSpecial[choiceOfSpecialICfunction,
             ICunitStepShift, ICunitStepHeight]; event = "reset"; gtick += delta} &],
        {-1., 1., .01}, ImageSize → Small, ContinuousAction -> False, Enabled →
         Dynamic[IntervalMemberQ[Interval@{4, 5}, choiceOfSpecialICfunction]]],
      Text@Style[Dynamic@padIt1[ICunitStepShift, {4, 2}], 11],
      Spacer[2],
      Button[Text@Style["zero", 10], {ICunitStepShift = 0.;
        event = "reset"; initialConditionFunction = makeInitialConditionsSpecial[
          choiceOfSpecialICfunction, ICunitStepShift, ICunitStepHeight];
        gtick += delta}, ImageSize → {45, 20}, Alignment → Center, Enabled →
        Dynamic[IntervalMemberQ[Interval@{4, 5}, choiceOfSpecialICfunction]]],
      SpanFromLeft
     },
     {
      Text@Style["height", 11],
      Manipulator[Dynamic[ICunitStepHeight, {ICunitStepHeight = #;
           initialConditionFunction = makeInitialConditionsSpecial[choiceOfSpecialICfunction,
             ICunitStepShift, ICunitStepHeight]; event = "reset"; gtick += delta} &],
        {0, 3, 0.01}, ImageSize → Small, ContinuousAction -> False, Enabled →
         Dynamic[IntervalMemberQ[Interval@{4, 5}, choiceOfSpecialICfunction]] ],
      Text@Style[Dynamic@padIt2[ICunitStepHeight, {3, 2}], 11],
      Spacer[2],
```

```
        Button[Text@Style["one", 10], {ICunitStepHeight = 1.;
          event = "reset"; initialConditionFunction = makeInitialConditionsSpecial[
            choiceOfSpecialICfunction, ICunitStepShift, ICunitStepHeight];
          gtick += delta}, ImageSize → {45, 20}, Alignment → Center, Enabled →
          Dynamic[IntervalMemberQ[Interval@{4, 5}, choiceOfSpecialICfunction]]], SpanFromLeft
       }
      }, Alignment → Center, Frame → None, Spacings → {.1, .4}
     ], FrameStyle -> Directive[Thickness[.005], Gray]
    ]
   }
  }, Alignment → Center, Spacings → {0, .6}
 ],
Text@Style["general", 11] →
 Grid[{
   {PopupMenu[Dynamic[initialConditionsSelection, {initialConditionsSelection = #;
       initialConditionFunction =
        makeInitialConditions[initialConditionsSelection, ICa, ICb, ICc, ICd, stdx, x0];
       event = "reset";
       gtick += delta} &],
     {1 → Text@Style[TraditionalForm[HoldForm[ζ]], 12],
      2 → Text@Style[TraditionalForm[HoldForm[ζ x]], 12],
      3 → Text@Style[TraditionalForm[HoldForm[ζ x + β x^2]], 12],
      4 → Text@Style[TraditionalForm[HoldForm[ζ x + β x^2 + γ x^3]], 12],
      5 → Text@Style[TraditionalForm[HoldForm[ζ x + β x^2 + γ x^3 + η x^4]], 12],
      6 → Text@Style[TraditionalForm[HoldForm[ζ Sin[β x]]], 12],
      7 → Text@Style[TraditionalForm[HoldForm[ζ Cos[β x]]], 12],
      8 → Text@Style[TraditionalForm[HoldForm[ζ Sin[β x] + γ Sin[η x]]], 12],
      9 → Text@Style[TraditionalForm[HoldForm[ζ Sin[β x] + γ Cos[η x]]], 12],
      10 → Text@Style[TraditionalForm[HoldForm[ζ Cos[β x] + γ Cos[η x]]], 12],
      11 → Text@Style[TraditionalForm[HoldForm[ζ (Sin[β x])^2]], 12],
      12 → Text@Style[TraditionalForm[HoldForm[ζ (Cos[β x])^2]], 12],
      13 → Text@Style[TraditionalForm[HoldForm[ζ Exp[β (x - η)^γ]]], 12],
      14 → Text@Style[TraditionalForm[HoldForm[1/(σ √(2 π))] * HoldForm[Exp[-(x - μ)^2/(2 σ^2)]]], 12]
     }, ContinuousAction -> False], SpanFromLeft
   },
   {
    Grid[{
      {Text@Style[TraditionalForm[HoldForm[ζ], 12]], Spacer[2],
       Manipulator[Dynamic[ICa, {ICa = #; initialConditionFunction =
            makeInitialConditions[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 = makeInitialConditions[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 = makeInitialConditions[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 =
```

```
            makeInitialConditions[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 = makeInitialConditions[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 = makeInitialConditions[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 =
            makeInitialConditions[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 = makeInitialConditions[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 = makeInitialConditions[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 =
            makeInitialConditions[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 = makeInitialConditions[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 = makeInitialConditions[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 =
            makeInitialConditions[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 = makeInitialConditions[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 = makeInitialConditions[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 =
          makeInitialConditions[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 = makeInitialConditions[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 = makeInitialConditions[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 = 115},
      If[centerGrid,
       (
        from = -length / 2;
        to = length / 2
       ),
       (
        from = 0;
        to = length
       )
      ];
      f = Evaluate@
        makeInitialConditions[initialConditionsSelection, ICa, ICb, ICc, ICd, stdx, x0][x];
      Plot[f, {x, from, to},
       ImagePadding → {{40, 10}, {20(*40*), 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 → {300(*322*), plotLength},
       TicksStyle → 9,
       AspectRatio → 0.3,
       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}
         ]
        }}
     ]
   },
   (*------------------------------*)
   (*--- LEVEL 2             -----*)
   (*------------------------------*)
   With[{

     pde = Grid[{
         {TabView[{
            Text@Style["geometry/boundary conditions", 12] → geometryMacro,
            Text@Style["initial conditions", 12] → initialConditionsMacro
           }]
         }
       }, Spacings → {0.2, .9}
      ]
    },
    (*--- end of level 2 ---*)
    ## &[
     Item[
      Grid[{
         {topRowMacro, plotOptionsMacro}
       }, Spacings → {3.9, 0}, Alignment → {Center, Top}
      ], ControlPlacement → Top
     ],

     Item[pde, ControlPlacement → Left]
    ]
   ]
  ],
(*----------- end of Manipulate controls ----------------------------*)

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

{{threeDViewSpeed, False}, None},
{{choiceOfSpecialICfunction, 5}, None},
{{ICheight, 1}, None},
{{ICwidth, .4}, None},
{{ICcenter, 0}, None},
{{ICunitStepShift, .2}, None},
{{ICunitStepHeight, 1}, None},
{{threeDView, True}, None},
{{animation3dBuffer, {0, {}}}, None},
{{finalDisplayImage, {}}, None},
{{testCase, 1}, None},
{{yscaleAuto, True}, None},
{{yscaleAmount, 1.1}, 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},
{{systemMatrix, {}}, None},
{{centerGrid, True}, None},
{{h, 0.03}, None},
{{length, 1}, None},
{{k, 0.25}, None},
{{aConvectionTerm, 8.}, None},
{{dAdvectionTerm, 1.}, None},
{{cDiffusionTerm, 1.}, None},
{{maxTime, 0.02}, None},
{{grid, generatePhysicalCoordinates1D[0.25, 1, True]}, None},
{{u, {}}, None},
{{u0, {}}, 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,
SynchronousInitialization → True,
Initialization :→
 {
  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 = 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_, $grid_, $AA_, $stepNumber_, $cpuTimeUsed_, $currentTime_, $state_, $u0_,
     animation3dBuffer_, initialConditionFunction_, event_, h_, centerGrid_, length_, k_,
     aConvectionTerm_, dAdvectionTerm_, cDiffusionTerm_, maxTime_, addGrid_, showIC_, joinedType_,
     yscaleAuto_, yscaleAmount_, threeDView_, threeDViewSpeed_, gtick_, delta_, gstatusMessage_] :=
    Module[{u = $u, u0 = $u0, grid = $grid, AA = $AA, stepNumber = $stepNumber, cpuTimeUsed = $cpuTimeUsed,
      currentTime = $currentTime, state = $state, finalDisplayImage, pde},

     pde = makePDE[cDiffusionTerm, dAdvectionTerm, aConvectionTerm];

     Which[state == "INIT",
      (
       (*system always starts with reset event and INIT state*)
       {u, grid, cpuTimeUsed, stepNumber, AA, currentTime, animation3dBuffer} = initializeSystem[
         initialConditionFunction, h, centerGrid,
         length, k, aConvectionTerm, dAdvectionTerm, cDiffusionTerm, 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"
      ),

      state == "PAUSE",
      (
```

```
 gstatusMessage = Row[{"paused [", stepNumber, "]"}];

 Which[
  event == "pause_button", state = "PAUSE",
  event == "reset",
  (
   state = "INIT";
   {u, grid, cpuTimeUsed, stepNumber, AA, currentTime, animation3dBuffer} = initializeSystem[
     initialConditionFunction, h, centerGrid,
     length, k, aConvectionTerm, dAdvectionTerm, cDiffusionTerm, maxTime];

   u0 = u;
   gtick += delta
  ),
  event == "run_button" || event == "step_button",
  (
   state = "RUNNING";
   gtick += delta
  )
 ]
),
state == "RUNNING",
(
 Which[event == "step_button" || event == "run_button" || event == "plot_changed",
  (
   If[currentTime < maxTime,
    (
     {u, cpuTimeUsed} = solve[u, AA];

     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, AA, currentTime, animation3dBuffer} = initializeSystem[
     initialConditionFunction, h, centerGrid,
     length, k, aConvectionTerm, dAdvectionTerm, cDiffusionTerm, maxTime];
   u0 = u;
   gtick += delta
  ),

  event == "pause_button",
  (
   state = "PAUSE";
```

```
            gtick += delta
           )
          ]
         )
        ];

        (* state machine completed, plot the final result *)
        finalDisplayImage =
         makeFinalPlot[u, grid, currentTime, u0, addGrid, showIC, joinedType, yscaleAuto, yscaleAmount,
          pde, Unevaluated[animation3dBuffer], stepNumber, maxTime, threeDView, k, threeDViewSpeed];
        {finalDisplayImage, u, u0, grid, AA, stepNumber, cpuTimeUsed, currentTime, state}
       ];
(*-----------------------------------------------------*)
    makePDE[ cDiffusionTerm_, dAdvectionTerm_, aConvectionTerm_] :=
     Module[{c, d, a, uxx, ut, ux, utTerm, uxxTerm, uxTerm},
      c = checkTerm@cDiffusionTerm;
      d = checkTerm@dAdvectionTerm;
      a = checkTerm@aConvectionTerm;
      uxx = Style["u", Italic, 11]Row[{Style["x",Italic,11],Style["x",Italic,11]}];
      ut = Style["u", Italic, 11]Style["t",Italic,11];
      ux = Style["u", Italic, 11]Style["x",Italic,11];
      utTerm = If[d == 1, ut, Row[{d, Spacer[1], ut}]];
      uxxTerm = Row[{If[c == 1, "", c], Spacer[1], uxx}];
      uxTerm = Row[{If[a == 1, "", a], Spacer[1], ux}];

      Text@Row[{Spacer[1], uxxTerm, " = ", utTerm, " + ", uxTerm}]
     ];
    (*-----------------------------------------------------*)
    makeFinalPlot[u_, grid_, currentTime_, u0_, addGrid_, showIC_, joinedType_, yscaleAuto_, yscaleAmount_,
      pde_, animation3dBuffer_, stepNumber_, maxTime_, threeDView_, timeStepDuration_, threeDViewSpeed_] :=
     Module[{finalDisplayImage, icData, data, h, n, m, nRow, timeScaleFor3Dplot, i, title, plotLabel},

      nRow = Dimensions[grid][[1]];

      (*-- use simple adaptive method to reduce memory use *)
      Which[
       stepNumber ≤ 10, timeScaleFor3Dplot = Min[maxTime, 10 * timeStepDuration],
       stepNumber > 10 && stepNumber ≤ 20, timeScaleFor3Dplot = Min[maxTime, 20 * timeStepDuration],
       stepNumber > 20 && stepNumber ≤ 50, timeScaleFor3Dplot = Min[maxTime, 50 * timeStepDuration],
       stepNumber > 50 && stepNumber ≤ 100, timeScaleFor3Dplot = Min[maxTime, 100 * timeStepDuration],
       stepNumber > 100 && stepNumber ≤ 200, timeScaleFor3Dplot = Min[maxTime, 200 * timeStepDuration],
       stepNumber > 200 && stepNumber ≤ 300, timeScaleFor3Dplot = Min[maxTime, 300 * timeStepDuration],
       stepNumber > 300 && stepNumber ≤ 500, timeScaleFor3Dplot = Min[maxTime, 500 * timeStepDuration],
       stepNumber > 500 && stepNumber ≤ 1000, timeScaleFor3Dplot = Min[maxTime, 1000 * timeStepDuration],
       stepNumber > 1000 && stepNumber ≤ 2000, timeScaleFor3Dplot = Min[maxTime, 2000 * timeStepDuration],
       True, timeScaleFor3Dplot = maxTime
      ];

      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,
```

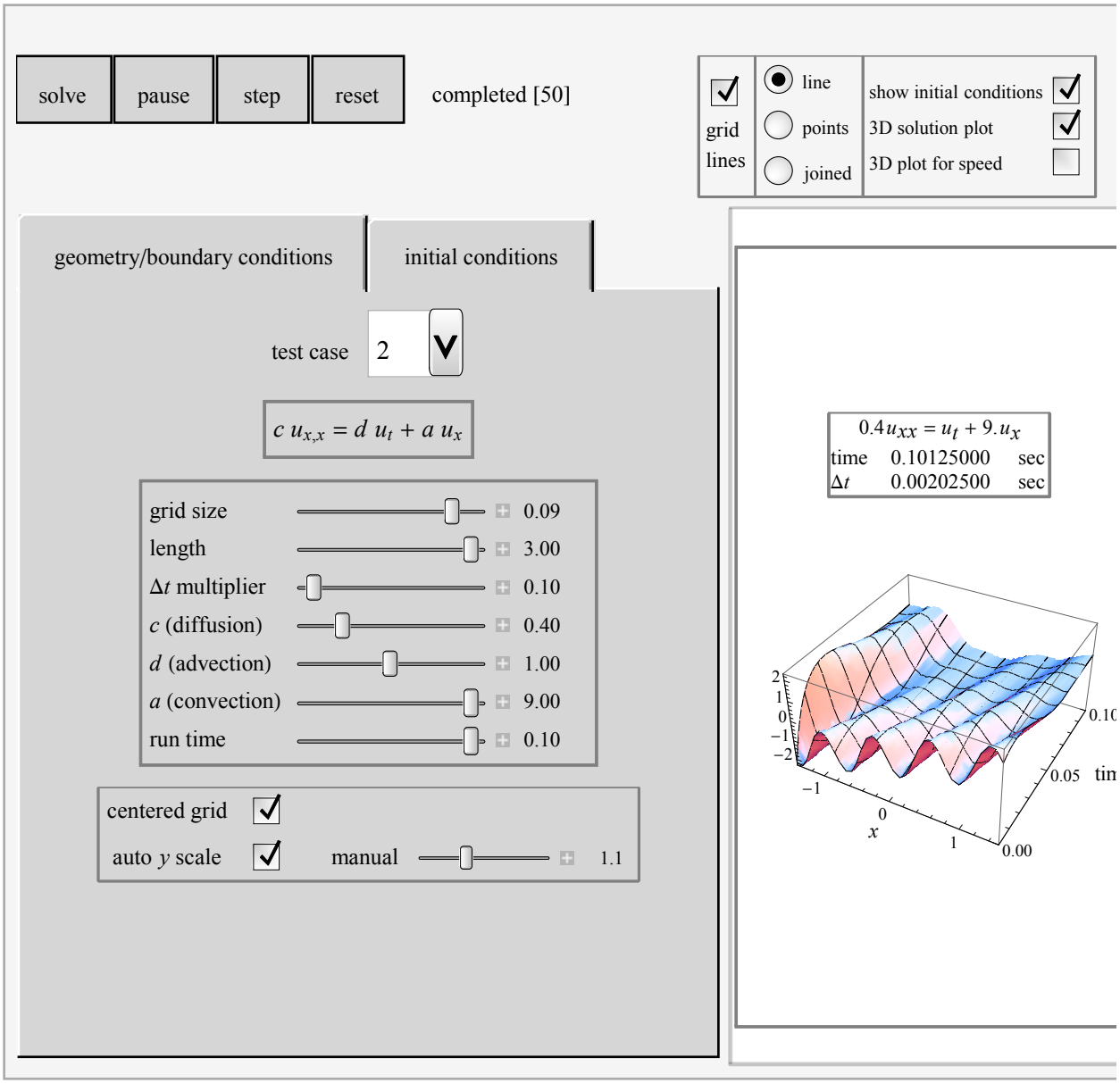
```
    (
     animation3dBuffer[[1]] = animation3dBuffer[[1]] + 1;
     n = animation3dBuffer[[1]];
     animation3dBuffer[[2]][[n]] = Table[{grid[[i]], currentTime, u[[i]]}, {i, Length[grid]}]
    ),
    n = animation3dBuffer[[1]]
   ]
  ),
  stepNumber > 20 && stepNumber ≤ 30,
  (
   If[Mod[stepNumber,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

 title = 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
  ] ;

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

```
      Which[threeDView == True,
       finalDisplayImage = ListPlot3D[animation3dBuffer[[2]][[1 ;; m]],
         AxesLabel → { (*add spacers to move labels away from axis a little *)
           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(*45*), 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},
         PlotRegion -> {{0.02, 0.98}, {0.02, 0.98}},
         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
     ];
(*------------------------------------------------------*)
   initializeSystem[initialConditionFunction_, h_, centerGrid_, length_, k_, a_, d_, c_, maxTime_] :=
    Module[{u, grid, cpuTimeUsed = 0., stepNumber = 0, AA, currentTime = 0., n, animation3dBuffer = {0, 0}},

     animation3dBuffer[[2]] = Table[0, {Ceiling[maxTime / k / 10] + 25}];
     animation3dBuffer[[1]] = 0;
     grid = N[generatePhysicalCoordinates1D[h, length, centerGrid]];
     n = Length[grid];
     u = Map[initialConditionFunction[#] &, grid];
     AA = makeSystemMatrix[k, h, d, c, a, n];
     {u, grid, cpuTimeUsed, stepNumber, AA, currentTime, animation3dBuffer}
    ];
(*------------------------------------------------------*)
   solve[$u_, AA_] := Module[{u = $u},
```

```
      u = AA.u;
      {u, 0}
     ];
(*-------------------------------------------------------*)
   makeSystemMatrix[k_, h_, d_, c_, a_, n_] := Module[{AA, v, mu},
      v = a*k/(d*h);
      mu = c*k/(d*h^2);
      AA = SparseArray[{
         Band[{1, 1}] → 1 - 2.*mu,
         Band[{2, 1}] → mu + v/2.0,
         Band[{1, 2}] → mu - v/2.0
        }, {n, n}
       ];

      AA[[-1, 1]] = mu - v/2.0;
      AA[[1, -1]] = v/2.0 + mu;

      AA
     ];
   (*-------------------------------------------------------*)
makeInitialConditions[sel_, a_, b_, c_, d_, stdx_, x0_] := Module[{f},

      f = 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)]]
       ];
      f
     ];
   (*-------------------------------------------------------*)
makeInitialConditionsSpecial[sel_, c_, w_, h_] := Module[{f},

      f = 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}
           }]]
        ];
       f
      ];
(*------------------------------------------------------*)
   makeInitialConditionsSpecial[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}
     }];
   (*----------------------------------------*)
   (* 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]]]];
   (*------------------------------------------------------*)
   ContentSizeW = 240;
   ContentSizeH = 420;
```

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

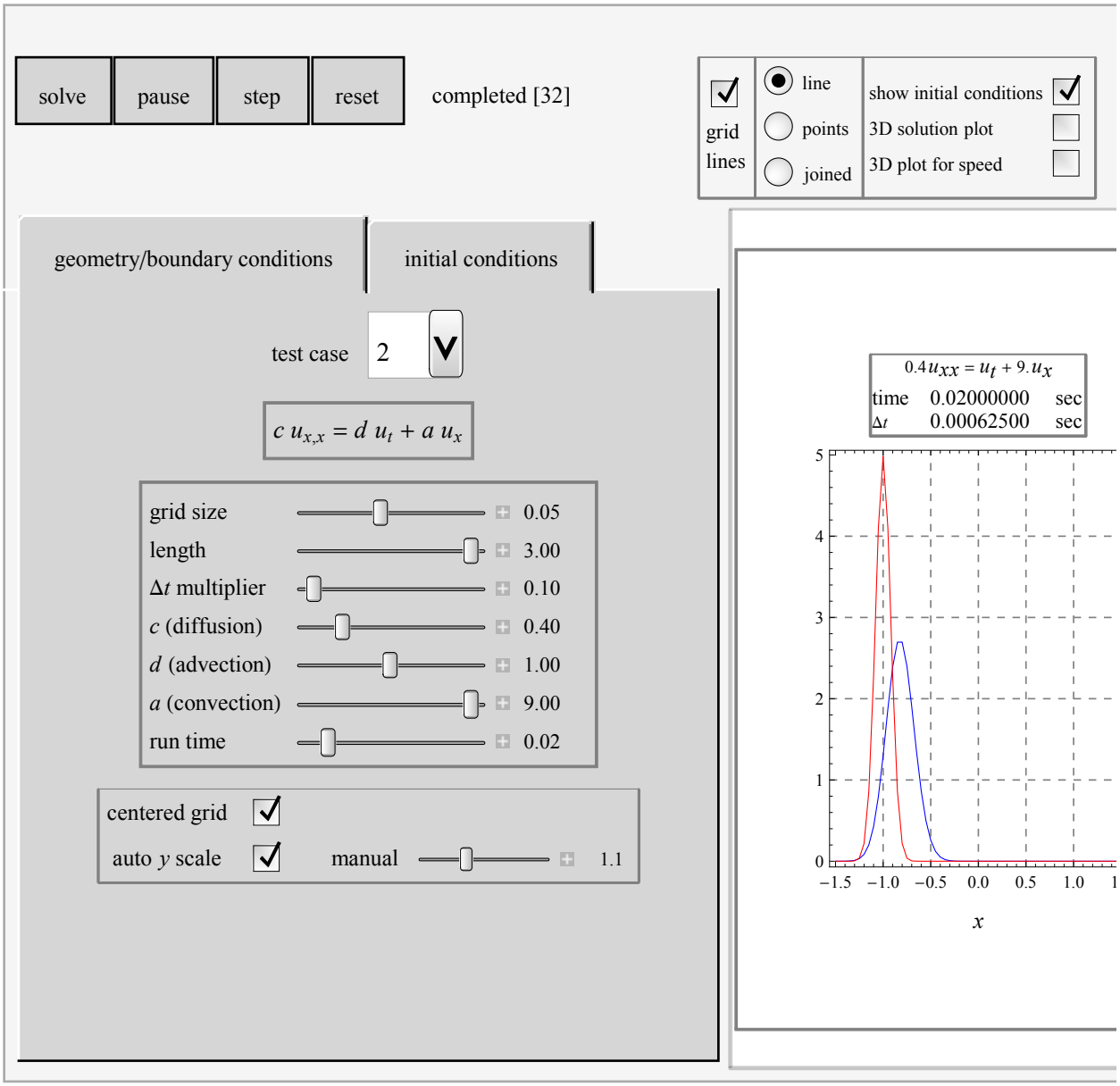


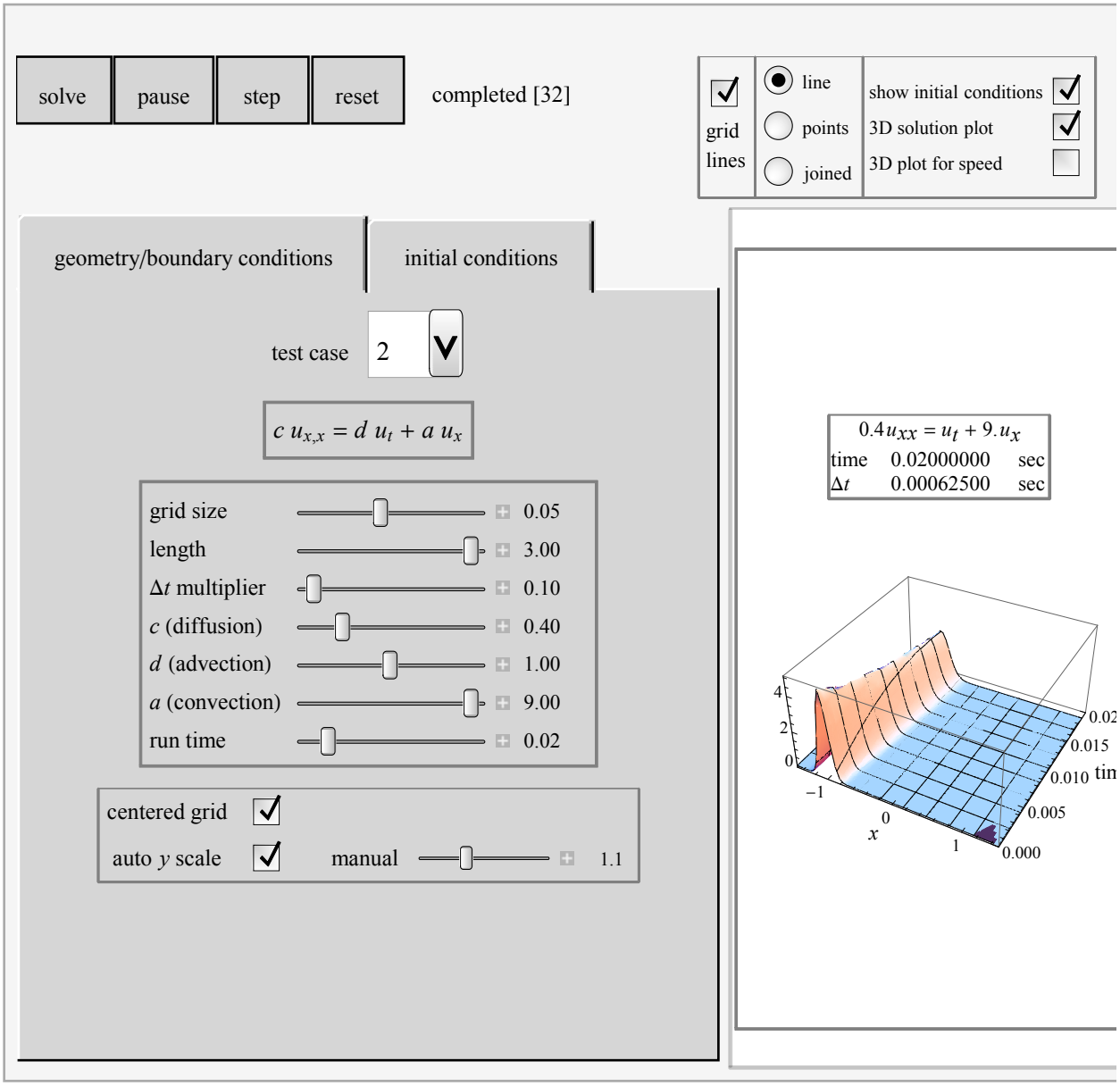
Caption

This Demonstration solves the convection-diffusion partial differential equation (PDE) $c\,u_{xx} = d\,u_t + a\,u_x$ in one dimension with periodic boundary conditions. You can specify different initial conditions. Selected preconfigured test cases are available from the pull down

menu.
The system is discretized in space and for each time step the solution is found using $u^{n+1} = A\,u^n$. The plot shown represents the solution $u(x, t)$. You can select to view the solution in 3D or in 2D using the controls at the top of the display.

Thumbnail





Details (optional)

The convection-diffusion partial differential equation (PDE) solved is $c\,u_{xx} = d\,u_t + a\,u_x$, where c is the diffusion parameter, d is the advection parameter (also called the transport parameter), and a is the convection parameter. The domain is $0 \le x \le L$ with periodic boundary conditions. Initial conditions are given by $u(x, 0) = g(x)$. You can specify $g(x)$ using the initial conditions button. The time step is $\Delta t = (\Delta t \text{ multiplier}) \frac{h^2}{c}$ where h is the grid size and c is the diffusion parameter. You can change the Δt multiplier using the slider. The total run time of the simulation is specified using the slider labeled "time".

The system solved at each time step is $\boldsymbol{u}^{n+1} = A\,\boldsymbol{u}^n$ where $\boldsymbol{u}$ is the solution of the PDE. The matrix A is given by

$$\begin{pmatrix} u_1^{n+1} \\ u_2^{n+1} \\ \vdots \\ u_{N-1}^{n+1} \\ u_N^{n+1} \end{pmatrix} = \begin{pmatrix} (1-2\mu) & \left(\mu-\frac{\nu}{2}\right) & 0 & \cdots & \left(\mu+\frac{\nu}{2}\right) \\ \left(\mu+\frac{\nu}{2}\right) & (1-2\mu) & \left(\mu-\frac{\nu}{2}\right) & \cdots & 0 \\ 0 & \left(\mu+\frac{\nu}{2}\right) & \ddots & \left(\mu-\frac{\nu}{2}\right) & 0 \\ \vdots & \vdots & \left(\mu+\frac{\nu}{2}\right) & (1-2\mu) & \left(\mu-\frac{\nu}{2}\right) \\ \left(\mu-\frac{\nu}{2}\right) & 0 & 0 & \left(\mu+\frac{\nu}{2}\right) & (1-2\mu) \end{pmatrix} \begin{pmatrix} u_1^n \\ u_2^n \\ \vdots \\ u_{N-1}^n \\ u_N^n \end{pmatrix},$$

where $\mu = \frac{c\,\Delta t}{d\,h^2}$ and $\nu = \frac{a\,\Delta t}{d\,h}$. In the above $\mathbf{u}^0$ is taken to be the vector of initial conditions. All values used are assumed to be in SI units.

S. J. Farlow, *Partial Differential Equations for Scientists and Engineers*, New York: Dover, 1993.

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)

diffusion
convection
heat equation
transport

Related Links (optional)

Heat Conduction Equation

Authoring Information

Contributed by: Nasser M. Abbasi