In[1]:=
```
(* The system is a unit square with a rod in the middle.
   We will do operator splitting on epsilon=
 0.02 and compare it to the non-split system and the without rod system.
     The diffusion constant of the sheet will be 0.02


    To save computation time in display the cross-section at y=0.5 is shown*)
```

In[2]:=

In[3]:=
```
maxT2 = 10; (*Time to calculate until*)
```

In[4]:=
```
initF[x_, y_] := 20 x (x - 1) y (y - 1); (*Initial temperature of sheet*)
```

In[5]:=
```
initFvals = Table[vals2[i / 100, 0.5, ti * maxT2 / 100], {i, 0, 100}, {ti, 0, 100}];
```

In[6]:=
```
allFvals = {initFvals}; (*Table of temperature values*)
```

In[7]:=
```
epsilon = 0.02;
```

In[8]:=
```
curY = 0.5;
```

In[9]:=
```
alphaRodValues = {0.05, 0.1, 0.2};
```

In[10]:=
```
alphaSheet = 0.02;
```

In[11]:=
```
(*Original System Before Operator Splitting*)
```

In[12]:=
```
vals2 =
 NDSolveValue[{D[u[x, y, t], t] - alphaSheet * Laplacian[u[x, y, t], {x, y}] == NeumannValue[0,
     x == 0 || x == 1 || y == 0 || y == 1], DirichletCondition[u[x, y, t] == initF[x, y], t == 0]},
  u, {x, 0, 1}, {y, 0, 1}, {t, 0, maxT2}] (*Temperature of sheet if no rod*)
```

NDSolveValue: The PDE is convection dominated and the result may not be stable. Adding artificial diffusion may help.

Out[12]=
```
InterpolatingFunction[ Domain: {{0., 1.}, {0., 1.}, {0., 10.}}
                       Output: scalar ]
```

In[13]:=
```
nonRodValues = Table[vals2[i / 100, curY, maxT2 * ti / 100], {i, 0, 100}, {ti, 0, 100}];
```

In[14]:=
```
epsilonValues = {0.1, 0.02, 0.01};
```

In[15]:=
```
allPreSplitValues = {nonRodValues};
```

In[16]:=
```
Do[(alphaRod = alphaRodValues[[curInd]];
   vals3 = NDSolveValue[{D[u[x, y, t], t] - alphaRod * Laplacian[u[x, y, t], {x, y}] ==
       NeumannValue[0, x == 0 || x == 1 || y == 0 || y == 1], DirichletCondition[
       u[x, y, t] == initF[x, y], t == 0]}, u, {x, 0, 1}, {y, 0, 1}, {t, 0, maxT2}];
   vals3Deriv[x_, y_, t_] := D[vals3[xx, yy, tt], tt] /. {xx → x, tt → t, yy → y};
   vals2Left =
    NDSolveValue[{D[u[x, y, t], t] - 0.02 * Laplacian[u[x, y, t], {x, y}] == NeumannValue[0,
         x == 0 || y == 0 || y == 1] + NeumannValue[vals3Deriv[x, y, t], x == 0.5 - epsilon],
      DirichletCondition[u[x, y, t] == vals3[x, y, t], x == 0.5 - epsilon],
      DirichletCondition[u[x, y, t] == initF[x, y], t == 0]},
     u, {x, 0, 0.5 - epsilon}, {y, 0, 1}, {t, 0, maxT2}];
   vals2Right = NDSolveValue[{D[u[x, y, t], t] - 0.02 * Laplacian[u[x, y, t], {x, y}] ==
       NeumannValue[vals3Deriv[x, y, t], x == 0.5 + epsilon] +
        NeumannValue[0, x == 1 || y == 0 || y == 1], DirichletCondition[u[x, y, t] == vals3[x, y, t],
       x == 0.5 + epsilon], DirichletCondition[u[x, y, t] == initF[x, y], t == 0]},
     u, {x, 0.5 + epsilon, 1}, {y, 0, 1}, {t, 0, maxT2}];
   wholePlotNoSplit[x_, y_, t_] := Piecewise[{{vals2Left[x, y, t], x ≤ 0.5 - epsilon},
      {vals3[x, y, t], x > 0.5 - epsilon && x ≤ 0.5 + epsilon},
      {vals2Right[x, y, t], x > 0.5 + epsilon}}];
   nonSplitValues = Table[wholePlotNoSplit[i / 100, curY, maxT2 * ti / 100],
     {i, 0, 100}, {ti, 0, 100}];
   AppendTo[allPreSplitValues, nonSplitValues]), {curInd, 1, Length[alphaRodValues]}];
```

NDSolveValue: The PDE is convection dominated and the result may not be stable. Adding artificial diffusion may help.

NDSolveValue: The PDE is convection dominated and the result may not be stable. Adding artificial diffusion may help.

NDSolveValue: The PDE is convection dominated and the result may not be stable. Adding artificial diffusion may help.

General: Further output of NDSolveValue::femcscd will be suppressed during this calculation.

In[17]:=

In[18]:=
```
(*Here I will attempt the operator split system*)
```

In[19]:=
```
NN = 100; tau = maxT2 / NN;
```

In[20]:=
```
timeInd = 1;
```

In[21]:=
```
tempEndT = 5;
```

In[22]:=
```
currentF [x_, y_] := initF[x, y];
```

In[23]:=
```
initialValues = Table[initF[i / 100, curY], {i, 0, 100}];
```

In[24]:=
```
foobar = Table[i, {i, 0, 3}]
```

Out[24]=
```
{0, 1, 2, 3}
```

In[25]:=
```
foobar2 = Table[i + 10, {i, 0, 3}]
```

Out[25]=
```
{10, 11, 12, 13}
```

In[26]:=
```
initFoo = {{foobar}, {foobar}, {foobar}}
```

Out[26]=
```
{{{0, 1, 2, 3}}, {{0, 1, 2, 3}}, {{0, 1, 2, 3}}}
```

In[27]:=
```
AppendTo[initFoo[[1]], foobar2]
```

Out[27]=
```
{{0, 1, 2, 3}, {10, 11, 12, 13}}
```

In[28]:=
```
initFoo
```
Out[28]=
```
{{{0, 1, 2, 3}, {10, 11, 12, 13}}, {{0, 1, 2, 3}}, {{0, 1, 2, 3}}}
```
In[29]:=
```
initFoo[[1, 2]]
```
Out[29]=
```
{10, 11, 12, 13}
```
In[30]:=

In[31]:=
```
allValues = {{initialValues}, {initialValues}, {initialValues}};
```
In[32]:=
```
Do[(alphaRod = alphaRodValues[[curInd]];
  Do[((*Iterate the rod for one time step. Doing O_{1D}. Neumann B.C. will be zero*)
    rodVals = NDSolveValue[{D[u[x, y, t], t] - alphaRod * Laplacian[u[x, y, t], {x, y}] ==
        NeumannValue[0, x == 0.5 - epsilon || x == 0.5 + epsilon || y == 0 || y == 1],
       DirichletCondition[u[x, y, t] == currentF[x, y], t == (timeInd - 1) * tau]}, u,
      {x, 0.5 - epsilon, 0.5 + epsilon}, {y, 0, 1}, {t, (timeInd - 1) * tau, timeInd * tau}];
    rodValsDeriv[x_, y_, t_] := D[rodVals[xx, yy, tt], tt] /. {xx → x, tt → t, yy → y};
    (*Iterate the left sheet for one time step. Doing O_{1→2}. Will
     match neumann values of rod and sheet to denote the energy transfer*)
    sheetLeftValsWithRod = NDSolveValue[
      {D[u[x, y, t], t] - alphaSheet * Laplacian[u[x, y, t], {x, y}] == NeumannValue[0,
          x == 0 || y == 0 || y == 1] + NeumannValue[rodValsDeriv[x, y, t], x == 0.5 - epsilon],
       DirichletCondition[u[x, y, t] == rodVals[x, y, t], x == 0.5 - epsilon],
       DirichletCondition[u[x, y, t] == currentF[x, y], t == (timeInd - 1) * tau]},
      u, {x, 0, 0.5 - epsilon}, {y, 0, 1}, {t, (timeInd - 1) * tau, timeInd * tau}];
    (*Iterate the right sheet for one time step. Doing O_{1→2}. Will match
     neumann values of rod and sheet to denote the energy transfer*)
    sheetRightValsWithRod = NDSolveValue[
      {D[u[x, y, t], t] - alphaSheet * Laplacian[u[x, y, t], {x, y}] == NeumannValue[
          rodValsDeriv[x, y, t], x == 0.5 + epsilon] + NeumannValue[0, x == 1 || y == 0 || y == 1],
       DirichletCondition[u[x, y, t] == rodVals[x, y, t], x == 0.5 + epsilon],
       DirichletCondition[u[x, y, t] == currentF[x, y], t == (timeInd - 1) * tau]},
      u, {x, 0.5 + epsilon, 1}, {y, 0, 1}, {t, (timeInd - 1) * tau, timeInd * tau}];

    (*Starting time step was rod then influencing
     the sheet. Next time step is sheet and then it influences the rod*)
    curT = timeInd * tau;
    currentF[x_, y_] := Piecewise[{{sheetLeftValsWithRod[x, y, curT], x < 0.5 - epsilon},
       {rodVals[x, y, curT], x ≥ 0.5 - epsilon && x ≤ 0.5 + epsilon},
       {sheetRightValsWithRod[x, y, curT], x > 0.5 + epsilon}}];

    (*Iterate the left sheet assuming no rod
     influence. Doing O_{2D}. Make neumann b.c. equal to zero for this. *)
    sheetLeftValsNoRod = NDSolveValue[{D[u[x, y, t], t] - alphaSheet * Laplacian[
          u[x, y, t], {x, y}] == NeumannValue[0, x == 0 || x == 0.5 - epsilon || y == 0 || y == 1],
       DirichletCondition[u[x, y, t] == currentF[x, y], t == timeInd * tau]}, u,
      {x, 0, 0.5 - epsilon}, {y, 0, 1}, {t, timeInd * tau, (timeInd + 1) * tau}];
    sheetLeftDeriv[x_, y_, t_] := D[sheetLeftValsNoRod[xx, yy, tt], tt] /.
      {xx → x, tt → t, yy → y};
    (*Iterate the right sheet assuming no rod influence. Doing O_
     {2D}. Make neumann b.c. equal to zero for this *)
    sheetRightValsNoRod = NDSolveValue[{D[u[x, y, t], t] - alphaSheet * Laplacian[
```

```
        u[x, y, t], {x, y}] == NeumannValue[0, x == 0.5 + epsilon || x == 1 || y == 0 || y == 1],
      DirichletCondition[u[x, y, t] == currentF[x, y], t == timeInd * tau]}, u,
     {x, 0.5 + epsilon, 1}, {y, 0, 1}, {t, timeInd * tau, (timeInd + 1) * tau}];
   sheetRightDeriv[x_, y_, t_] := D[sheetRightValsNoRod[xx, yy, tt], tt] /.
     {xx → x, tt → t, yy → y};
   (*Iterate the rod assuming sheet influence. Doing O_{2→1}. Make
    neumann b.c. equals for this to happen*)
   rodValsWithSheet = NDSolveValue[{D[u[x, y, t], t] - alphaRod * Laplacian[u[x, y, t], {x, y}] ==
       NeumannValue[0, y == 0 || y == 1] + NeumannValue[sheetLeftDeriv[x, y, t], x == 0.5 - epsilon] +
        NeumannValue[sheetRightDeriv[x, y, t], x == 0.5 + epsilon],
      DirichletCondition[u[x, y, t] == sheetLeftValsNoRod[x, y, t], x == 0.5 - epsilon],
      DirichletCondition[u[x, y, t] == sheetRightValsNoRod[x, y, t], x == 0.5 + epsilon],
      DirichletCondition[u[x, y, t] == currentF[x, y], t == timeInd * tau]}, u,
     {x, 0.5 - epsilon, 0.5 + epsilon}, {y, 0, 1}, {t, timeInd * tau, (timeInd + 1) * tau}];

   curT = (timeInd + 1) * tau;
   currentF[x_, y_] := Piecewise[{{sheetLeftValsNoRod[x, y, curT], x < 0.5 - epsilon},
      {rodValsWithSheet[x, y, curT], x ≥ 0.5 - epsilon && x ≤ 0.5 + epsilon},
      {sheetRightValsNoRod[x, y, curT], x > 0.5 + epsilon}}];

   currentValues = Table[currentF[i / 100, curY], {i, 0, 100}];
   AppendTo[allValues[[curInd]], currentValues];), {timeInd, 1, tempEndT}];), {curInd, 1, 3}]
(*wholePlotNoSplit[x_,y_,t_]:=Piecewise[{{vals2Left[x,y,t],x≤ 0.5-epsilon},
   {vals3[x,y,t],x>0.5-epsilon && x ≤ 0.5+epsilon},{vals2Right[x,y,t],x>0.5+epsilon}}];*)
```

NDSolveValue: The PDE is convection dominated and the result may not be stable. Adding artificial diffusion may help.

NDSolveValue: The PDE is convection dominated and the result may not be stable. Adding artificial diffusion may help.

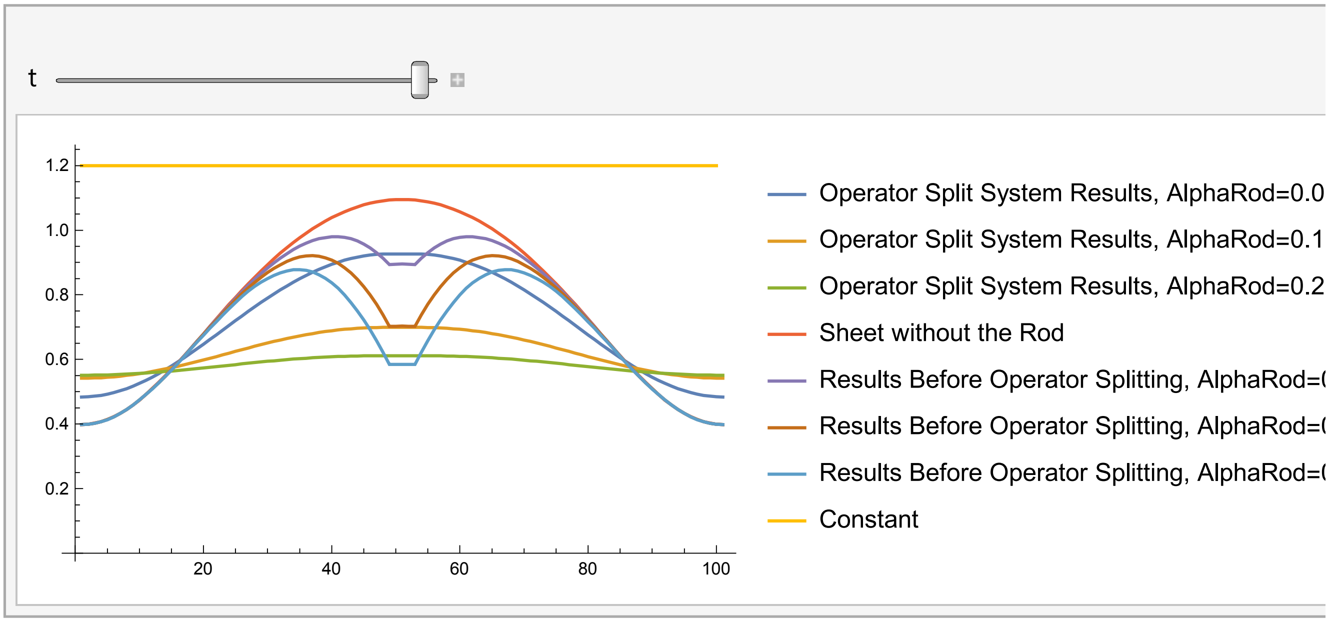
NDSolveValue: The PDE is convection dominated and the result may not be stable. Adding artificial diffusion may help.

General: Further output of NDSolveValue::femcscd will be suppressed during this calculation.

In[33]:=
```
(*Plot the results*)
```

In[34]:=
```
Manipulate[ListPlot[{allValues[[1, t]], allValues[[2, t]], allValues[[3, t]],
   allPreSplitValues[[1]][[All, t]], allPreSplitValues[[2]][[All, t]],
   allPreSplitValues[[3]][[All, t]], allPreSplitValues[[4]][[All, t]], Table[1.2, {i, 100}]},
  Joined → True, PlotLegends → {"Operator Split System Results, AlphaRod=0.05",
    "Operator Split System Results, AlphaRod=0.1",
    "Operator Split System Results, AlphaRod=0.2", "Sheet without the Rod",
    "Results Before Operator Splitting, AlphaRod=0.05",
    "Results Before Operator Splitting, AlphaRod=0.1",
    "Results Before Operator Splitting, AlphaRod=0.2", "Constant"}], {t, 1, tempEndT, 1}]
```

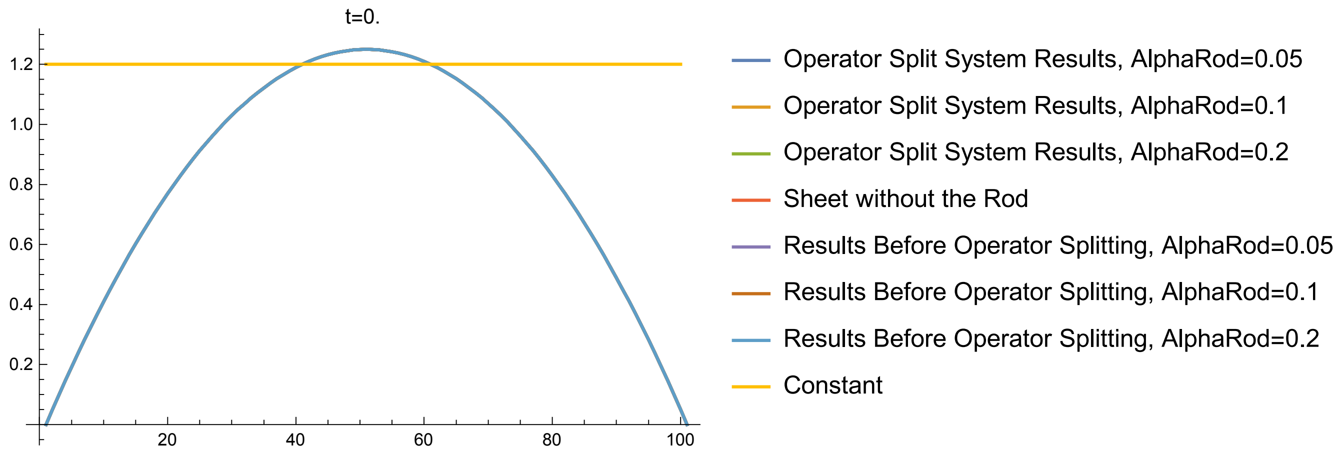
Out[34]=



In[35]:=
```
(*Shows the First Step and Plot Legend*)
```

In[39]:=
```
(t = 1;
 dispT = (t - 1) / 100 // N;
 ListPlot[{allValues[[1, t]], allValues[[2, t]], allValues[[3, t]],
   allPreSplitValues[[1]][[All, t]], allPreSplitValues[[2]][[All, t]],
   allPreSplitValues[[3]][[All, t]], allPreSplitValues[[4]][[All, t]], Table[1.2, {i, 100}]},
  Joined → True, PlotLegends → {"Operator Split System Results, AlphaRod=0.05", "Operator Split
     System Results, AlphaRod=0.1", "Operator Split System Results, AlphaRod=0.2",
    "Sheet without the Rod", "Results Before Operator Splitting, AlphaRod=0.05",
    "Results Before Operator Splitting, AlphaRod=0.1", "Results Before Operator Splitting,
     AlphaRod=0.2", "Constant"}, PlotLabel → StringJoin["t=", ToString[dispT]]])
```

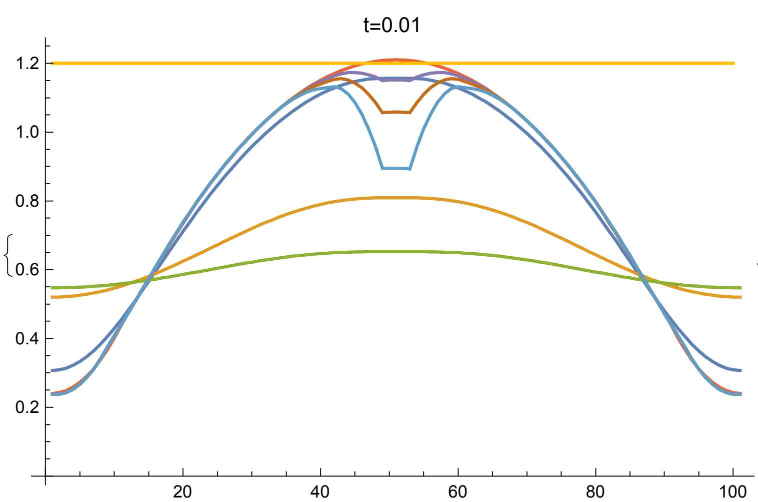
Out[39]=



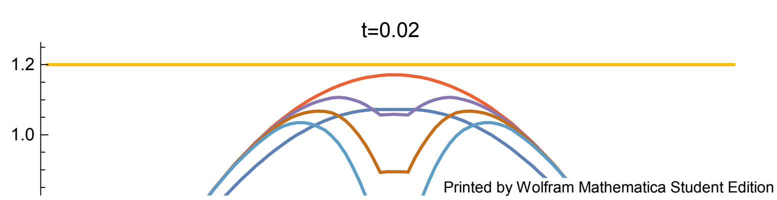
In[37]:=
```
(*Shows Subsequent Time Steps*)
```

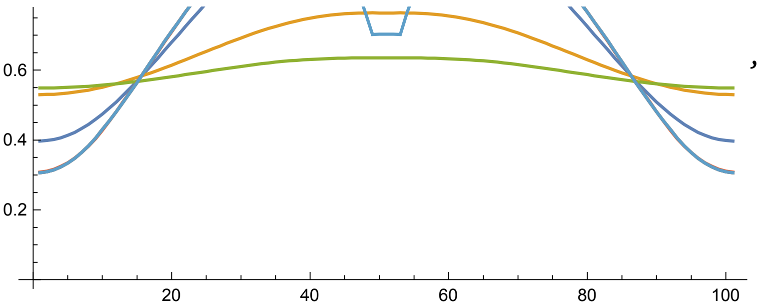
In[41]:=
```
Table[(dispT = (t - 1) / 100 // N;
  ListPlot[{allValues[[1, t]], allValues[[2, t]], allValues[[3, t]],
    allPreSplitValues[[1]][[All, t]], allPreSplitValues[[2]][[All, t]],
    allPreSplitValues[[3]][[All, t]], allPreSplitValues[[4]][[All, t]], Table[1.2, {i, 100}]},
   Joined → True, PlotLabel → StringJoin["t=", ToString[dispT]]]), {t, 2, tempEndT, 1}]
```

Out[41]= {



,

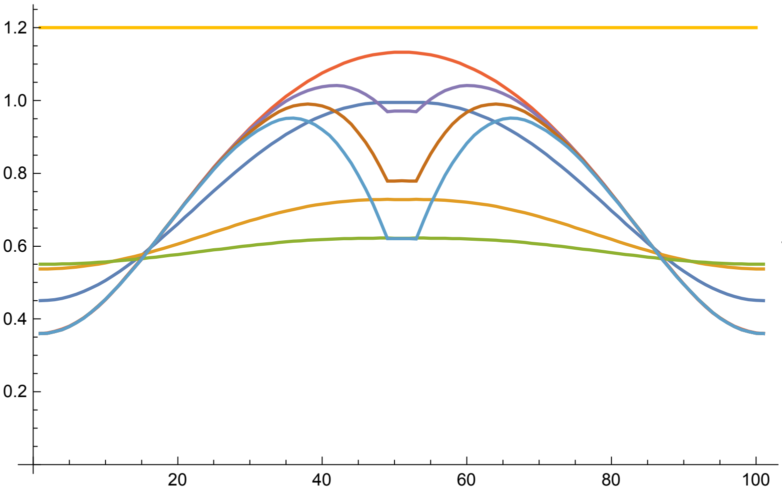




0.6
0.4
0.2
20
40
60
80
100

,

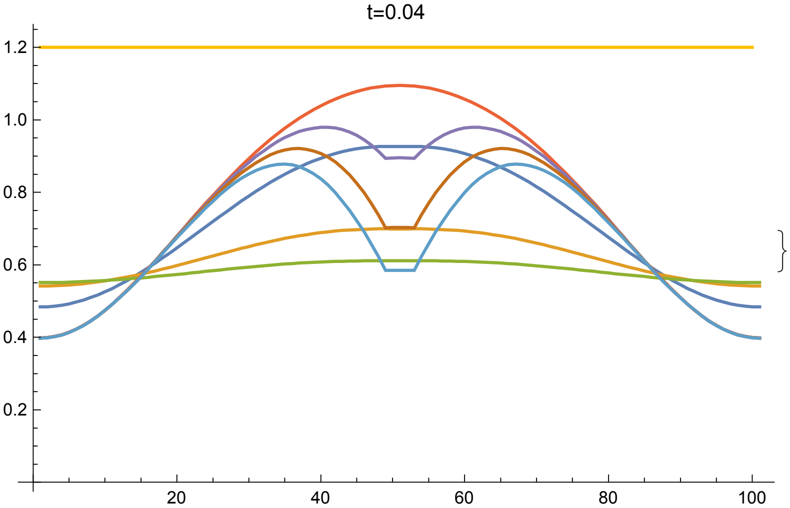
t=0.03



1.2
1.0
0.8
0.6
0.4
0.2
20
40
60
80
100

,

t=0.04



1.2
1.0
0.8
0.6
0.4
0.2
20
40
60
80
100

}