Things to possibly change:
- change potential: mexican hat, sine gordon, and ϕ^4+ϕ^2
- ϕ complex
- sine gordon for real ϕ: ϕ(t,x) is in U(1) -> use transversal unit circle in the plot
- λ strength of perturbation
- # of spacial dimensions
- friction term
- fourier

```
In[1]:= λ = 0.7;
        Vcoup[ϕ_, λ_] := λ (ϕ^2 - 1) ^2;
        V[ϕ_] := Vcoup[ϕ, λ];
        Equationcoup[ϕ_, λ_][t_, x_] := Derivative[2, 0][ϕ][t, x] -
          Derivative[0, 2][ϕ][t, x] + Derivative[1, 0][Vcoup][ϕ[t, x], λ]
        Equation[ϕ_][t_, x_] := Equationcoup[ϕ, λ][t, x]
```

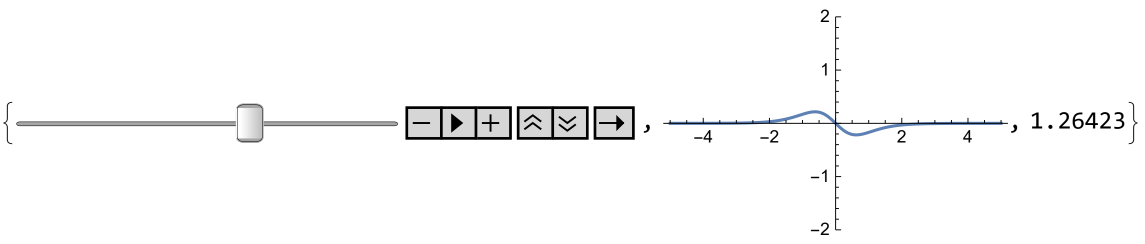
1) Analytic stuff: trivial solutions

```
In[6]:= stableneg[t_, x_] := -1
        stablepos[t_, x_] := 1
        unstable[t_, x_] := 0

In[9]:= Equation[unstable][t, x]

Out[9]= 0.
```

2) Kink: find time independent kink solution: partial_xx ϕ (x)= V'(ϕ(x))
One can interpret x as time and find a classical mechanics system with 1 d.o.f., then use energy conservation to lower the order of ODE, set energy to critical value, separate variables and integrate (even analytically). (! mind the sign in front of the potential)
Otherwise cheat like this: do a Tanh(k*x) ansatz for the solution, and search for the constant k; for dimensional reasons, k= Sqrt(λ)*a; for the numerical factor a, let it vary in the cartoon below and find out a=Sqrt(2) yields a solution

```
In[10]:= kinka[a_][t_, x_] := Tanh[a Sqrt[λ] x]

In[11]:= DynamicModule[{a}, {Animator[Dynamic[a], {0, 2}],
           Dynamic[Plot[Equation[kinka[a]][t, x], {x, -5, 5}, PlotRange → 2]], Dynamic[a]}]
```

Out[11]= {

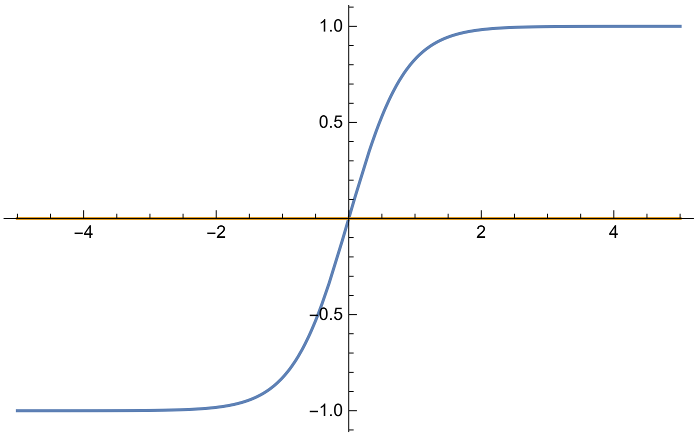


, 1.26423}

```
In[12]:= kink[t_, x_] := kinka[Sqrt[2]][t, x]
```

```
In[13]:=
Plot[{kink[t, x], Equation[kink][t, x]}, {x, -5, 5}]
```

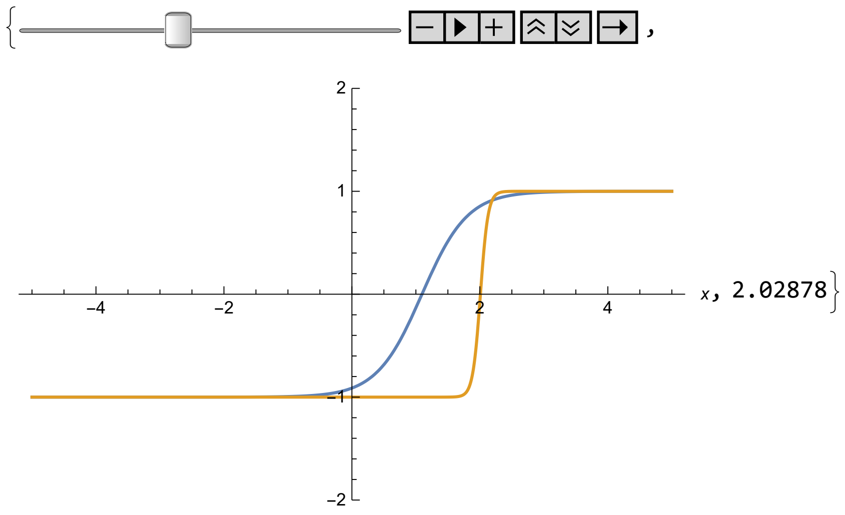
Out[13]=



3) Boost the kink to obtain time-dependent solutions: here 2 examples, notice space contraction:

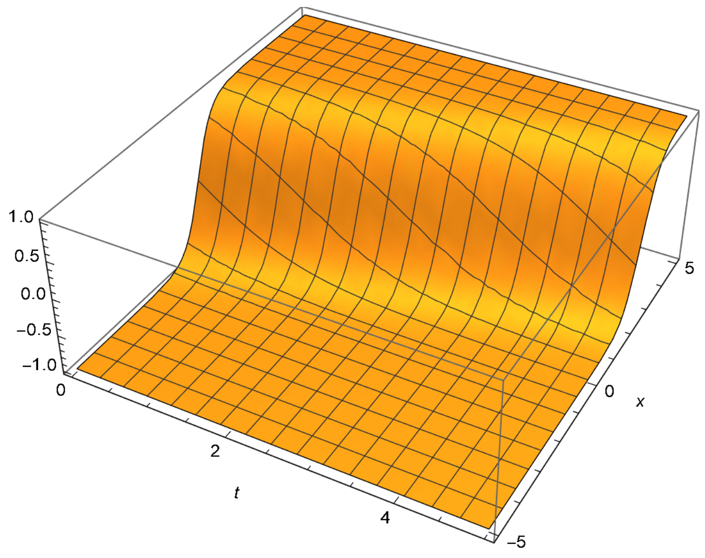
```
In[14]:= boostkink[Λ_][t_, x_] := kink[First[Λ.{t, x}], Last[Λ.{t, x}]]

In[15]:= timeplot[solution_, timestart_, timeend_, spacestart_, spaceend_] :=
          DynamicModule[{t}, {Animator[Dynamic[t], {timestart, timeend}],
            Dynamic[Plot[solution[t, x], {x, spacestart, spaceend}, PlotRange → {-2, 2},
              ImageSize → Medium, AxesLabel → Automatic]], Dynamic[t]}]
         doubletimeplot[solutions_, timestart_, timeend_, spacestart_, spaceend_] :=
          DynamicModule[{t}, {Animator[Dynamic[t], {timestart, timeend}], Dynamic[
             Plot[{solutions[[1]][t, x], solutions[[2]][t, x]}, {x, spacestart, spaceend},
              PlotRange → {-2, 2}, ImageSize → Medium, AxesLabel → Automatic]], Dynamic[t]}]

In[17]:= velocities = {0.54, 0.99};
         γ[v_] := (1 - v^2) ^ (-0.5);
         Λ[v_] := {{γ[v], -v γ[v]}, {-v γ[v], γ[v]}};
         Λs = Table[Λ[velocities[[i]]], {i, 1, 2}];
         spaceend = 5.;
         timeend = 5.;
         doubletimeplot[{boostkink[Λs[[1]]], boostkink[Λs[[2]]]},
          0, timeend, -spaceend, spaceend]
```

Out[23]=



```
In[24]:= Plot3D[boostkink[Λs[[1]]][t, x], {t, 0, 5}, {x, -5, 5}, AxesLabel → Automatic]
```

Out[24]=



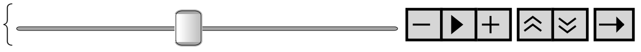
4) Now NUMERICAL solutions. I use a space with finite size (= 10 for now).

For a start: at space boundary fields have fixed values (dirichlet bdy conditions).

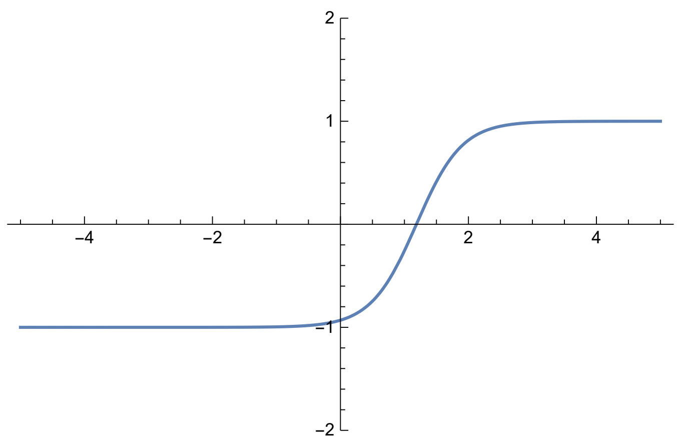
Find numerically the moving kink we found analytically (watch out on boundaries):

```
In[25]:= velocity 0.54
         v = 0.54;
         spaceend = 5.;
         timeend = 5.;
         ϕ0[x_] := Tanh[Sqrt[2 λ] × γ[v] x];
         sol = NDSolve[
            {Equation[ϕ][t, x] == 0 , ϕ[0, x] == ϕ0[x], Derivative[1, 0][ϕ][0, x] == -v ϕ0'[x],
             ϕ[t, -spaceend] == ϕ0[-spaceend], ϕ[t, spaceend] == ϕ0[spaceend]},
            ϕ, {t, 0, timeend}, {x, -spaceend, spaceend}];

         timeplot[Evaluate[ϕ[#1, #2] /. sol &], 0, timeend, -spaceend, spaceend]

Out[25]= 0.54 velocity
```

Out[31]= {



,



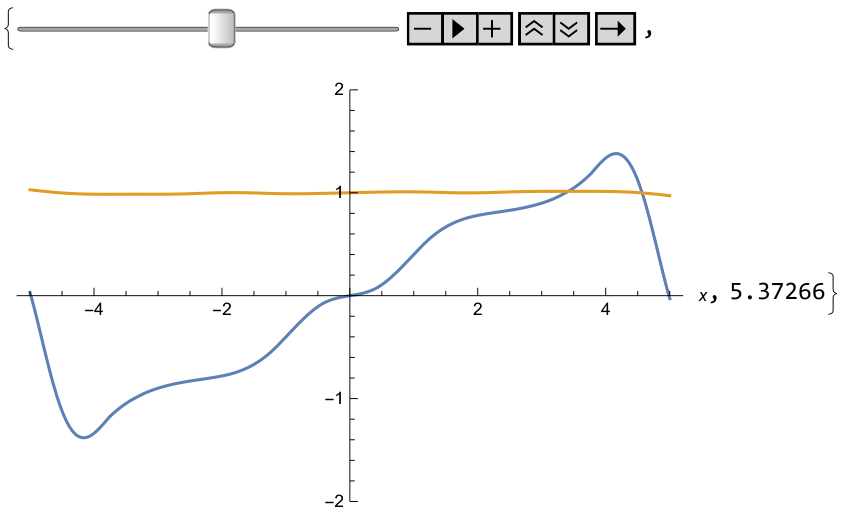
x, 2.20155}

5) Check (un) stability of the trivial solutions:

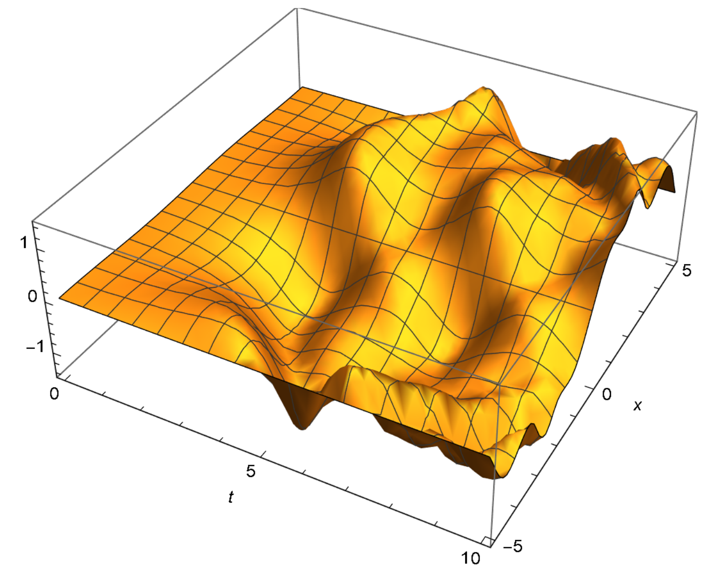
```
In[32]:= perturb[x_] := 0.03 Sin[x]

In[33]:= a =
           NDSolve[{Equation[ϕ][t, x] == 0, ϕ[0, x] == perturb[x], Derivative[1, 0][ϕ][0, x] == 0,
              ϕ[t, -spaceend] == perturb[-spaceend], ϕ[t, spaceend] == perturb[spaceend]},
            ϕ, {t, 0, 2 timeend}, {x, -spaceend, spaceend}];
         ϕ0stable[x_] := 1 + perturb[x];
         b = NDSolve[
            {Equation[ϕ][t, x] == 0, ϕ[0, x] == ϕ0stable[x], Derivative[1, 0][ϕ][0, x] == 0,
              ϕ[t, -spaceend] == ϕ0stable[-spaceend], ϕ[t, spaceend] == ϕ0stable[spaceend]},
            ϕ, {t, 0, 2 timeend}, {x, -spaceend, spaceend}];

         doubletimeplot[{Evaluate[ϕ[#1, #2] /. a &], Evaluate[ϕ[#1, #2] /. b &]},
          0, 2 * timeend, -spaceend, spaceend]

         Plot3D[Evaluate[ϕ[t, x] /. a], {t, 0, 2 timeend},
          {x, -spaceend, spaceend}, PlotRange → {-1.5, 1.5}, AxesLabel → Automatic]
         Plot3D[Evaluate[ϕ[t, x] /. b], {t, 0, 2 timeend}, {x, -spaceend, spaceend},
          PlotRange → {0.5, 1.5}, AxesLabel → Automatic]
```

NDSolve: Warning: scaled local spatial error estimate of 115.25968430947908` at t = 10.` in the direction of independent variable x is much greater than the prescribed error tolerance. Grid spacing with 25 points may be too large to achieve the desired accuracy or precision. A singularity may have formed or a smaller grid spacing can be specified using the MaxStepSize or MinPoints method options.

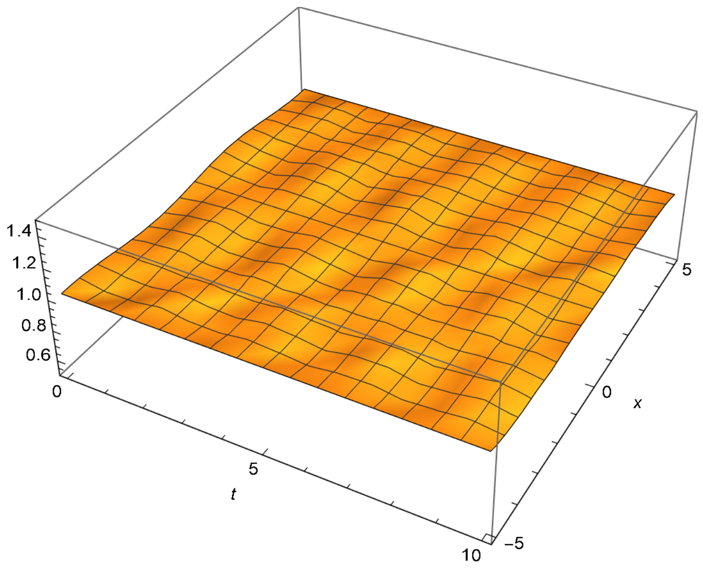
Out[36]= {



Out[37]=

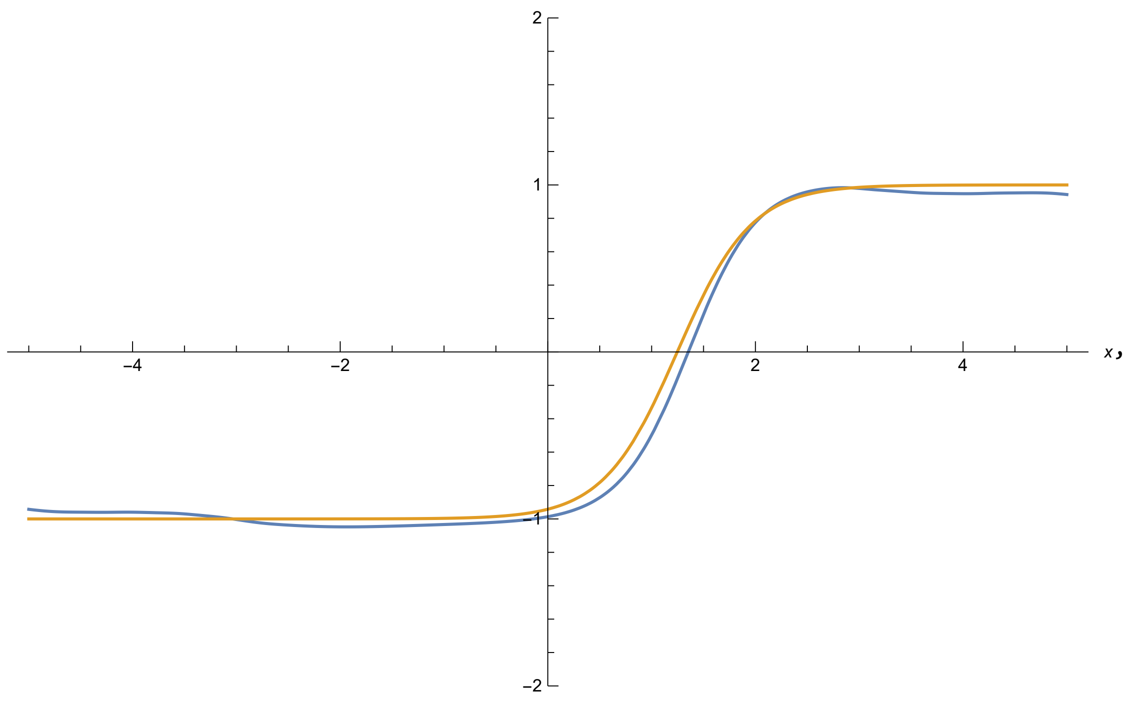


Out[38]=



6) check stability of moving kink solutions

```
In[39]:= ϕ0pert[x_] := Tanh[Sqrt[2 λ] × γ[v] x] + 2 perturb[x];
         z = NDSolve[{Equation[ϕ][t, x] == 0,
              ϕ[0, x] == ϕ0pert[x], Derivative[1, 0][ϕ][0, x] == -v ϕ0pert'[x],
              ϕ[t, -spaceend] == ϕ0pert[-spaceend], ϕ[t, spaceend] == ϕ0pert[spaceend]},
            ϕ, {t, 0, timeend}, {x, -spaceend, spaceend}];
         DynamicModule[{t}, {Animator[Dynamic[t], {0, 5}],
           Dynamic[Plot[{Evaluate[ϕ[t, x] /. z], boostkink[Λs[[1]]][t, x]}, {x, -5, 5},
              PlotRange → 2, ImageSize → Large, AxesLabel → Automatic]], Dynamic[t]}]

         NDSolve: Warning: boundary and initial conditions are inconsistent.

Out[41]= {
```



```
 2.30984}
```

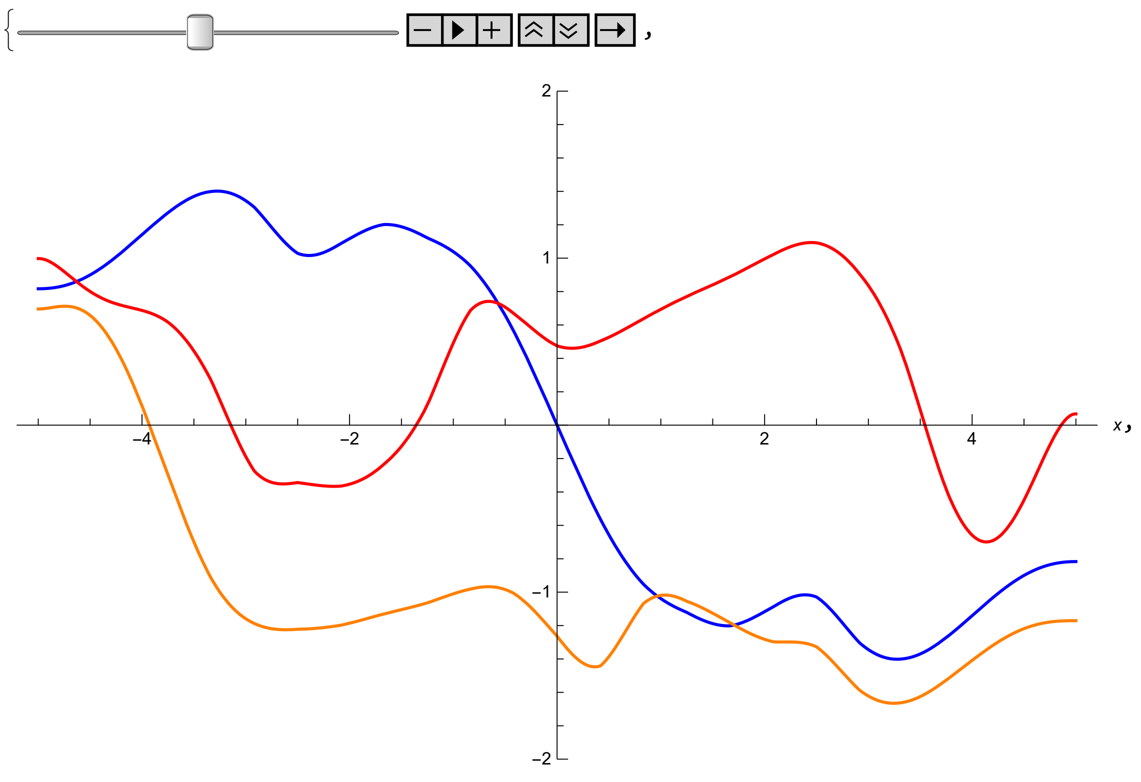
7) Some ideas that drive choices of boundary and initial conditions:
- with standard boundary conditions, a spatially odd (even) initial field configuration yields a spatially odd (even) field at all times. i.p. a spatially odd field can keep its (spacial) mean value on the unstable value <ϕ> = 0 - see the mean values plot below
- energy conservation keeps the field from falling in one of the two minima (this will change later when a friction term is added)
- for a field starting from unstable value 0: dirichlet bdy conditions ϕ (t, ±spaceend)= 0 further prevent it from falling into one of the two minima

Try von Neumann boundary conditions, partial_x(t,±spaceend)=0 or fixed value...change initial cond. accordingly
(NB can change speed of animation manually in output)

```
In[42]:= newtimeend = 40;
         perturbb[x_] := 0.03 Sin[x * 3 Pi / 10]
         aa =
           NDSolve[{Equation[ϕ][t, x] == 0, ϕ[0, x] == perturbb[x], Derivative[1, 0][ϕ][0, x] == 0,
             Derivative[0, 1][ϕ][t, -spaceend] == 0, Derivative[0, 1][ϕ][t, spaceend] == 0},
            ϕ, {t, 0, newtimeend}, {x, -spaceend, spaceend}];
         bb = NDSolve[{Equation[ϕ][t, x] == 0, ϕ[0, x] == perturbb[x - 1],
             Derivative[1, 0][ϕ][0, x] == 0, Derivative[0, 1][ϕ][t, -spaceend] == 0,
             Derivative[0, 1][ϕ][t, spaceend] == 0},
            ϕ, {t, 0, newtimeend}, {x, -spaceend, spaceend}];
         cc = NDSolve[{Equation[ϕ][t, x] == 0, ϕ[0, x] == perturbb[x] + 0.1,
             Derivative[1, 0][ϕ][0, x] == 0, Derivative[0, 1][ϕ][t, -spaceend] == 0,
             Derivative[0, 1][ϕ][t, spaceend] == 0},
            ϕ, {t, 0, newtimeend}, {x, -spaceend, spaceend}];
         colors = {Blue, Red, Orange};

         DynamicModule[{t},
          {Animator[Dynamic[t], {0, newtimeend}], Dynamic[Plot[{Evaluate[ϕ[t, x] /. aa],
              Evaluate[ϕ[t, x] /. bb], Evaluate[ϕ[t, x] /. cc]}, {x, -5, 5}, PlotRange → 2,
             ImageSize → Large, AxesLabel → Automatic, PlotStyle → colors]], Dynamic[t]}]
```

```
NDSolve: Warning: scaled local spatial error estimate of 181.75325582515165` at t = 40.` in the direction of independent
   variable x is much greater than the prescribed error tolerance. Grid spacing with 25 points may be too large to achieve
   the desired accuracy or precision. A singularity may have formed or a smaller grid spacing can be specified using the
   MaxStepSize or MinPoints method options.

NDSolve: Warning: boundary and initial conditions are inconsistent.

NDSolve: Warning: scaled local spatial error estimate of 348.7000261083287` at t = 40.` in the direction of independent
   variable x is much greater than the prescribed error tolerance. Grid spacing with 25 points may be too large to achieve
   the desired accuracy or precision. A singularity may have formed or a smaller grid spacing can be specified using the
   MaxStepSize or MinPoints method options.

NDSolve: Warning: scaled local spatial error estimate of 251.404833738897` at t = 40.` in the direction of independent
   variable x is much greater than the prescribed error tolerance. Grid spacing with 25 points may be too large to achieve
   the desired accuracy or precision. A singularity may have formed or a smaller grid spacing can be specified using the
   MaxStepSize or MinPoints method options.
```

Out[48]= {



```
18.9052}
```

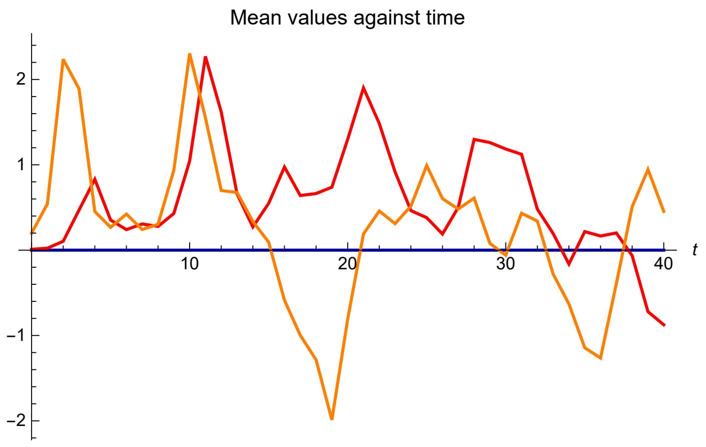
```
In[49]:= meanvalues[t_, solution_] :=
          NIntegrate[Evaluate[ϕ[t, x] /. solution], {x, -spaceend, spaceend}] / spaceend
         meanvaluesplot[solution_, integerstarttime_, integerendtime_, color_] :=
          DiscretePlot [meanvalues[t, solution], {t, integerstarttime, integerendtime},
           Joined → True, PlotRange → All, Filling → None,
           PlotLabel → "Mean values against time", AxesLabel → Automatic, PlotStyle → color]
```

Mean values of field against time (spacial mean values at fixed time):

Running the following takes some time

```
In[51]:= solutionscolors = {{aa, colors[[1]]}, {bb, colors[[2]]}, {cc, colors[[3]]}};
         plots = Map[meanvaluesplot[First[#], 0, newtimeend, Last[#]] &, solutionscolors];

In[53]:= Show[plots, PlotRange → All]
```

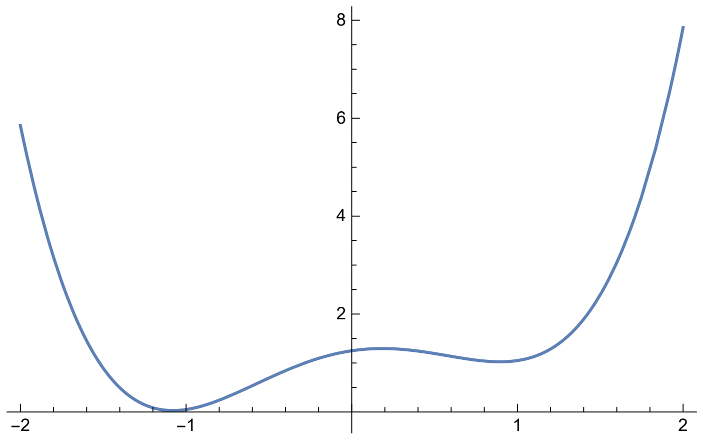
Out[53]=



Above here: Notice the spatially odd solution, blue, with mean value = 0 for all t

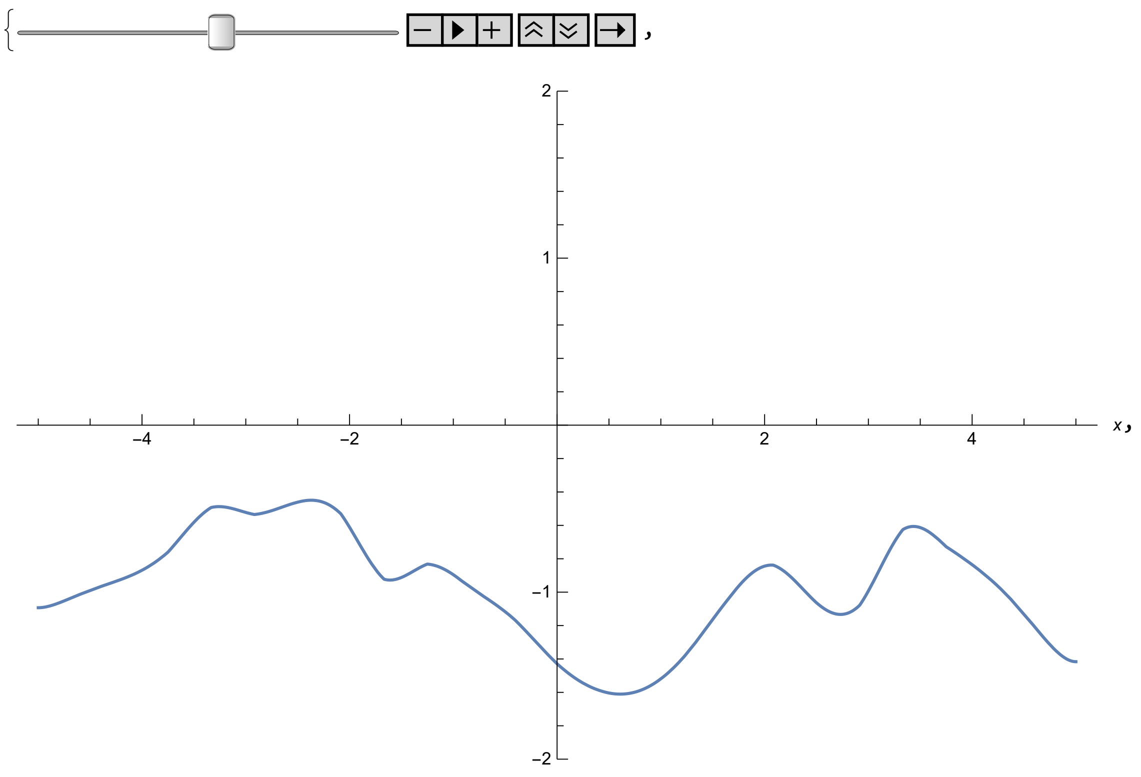
8) Slightly move the two minima; start with field around weaker minimum; give the field enough initial kinetic energy for it to drop to stronger minimum.
! time independent kink solutions can’t exist

```
In[54]:= moving = 0.5;
         W[ϕ_] := V[ϕ] + moving (ϕ + 1.1)
         MovedEquation[ϕ_][t_, x_] :=
          Derivative[2, 0][ϕ][t, x] - Derivative[0, 2][ϕ][t, x] + W'[ϕ[t, x]]
         Plot[W[ϕ], {ϕ, -2, 2}]
         A = NDSolve[
             {MovedEquation[ϕ][t, x] == 0, ϕ[0, x] == 1, Derivative[1, 0][ϕ][0, x] == Sin[x + 3],
              Derivative[0, 1][ϕ][t, -spaceend] == 0, Derivative[0, 1][ϕ][t, spaceend] == 0},
             ϕ, {t, 0, newtimeend}, {x, -spaceend, spaceend}];

         DynamicModule[{t},
          {Animator[Dynamic[t], {0, newtimeend}], Dynamic[Plot[{Evaluate[ϕ[t, x] /. A]},
              {x, -5, 5}, PlotRange → 2, ImageSize → Large, AxesLabel → Automatic]], Dynamic[t]}]
```

Out[57]=


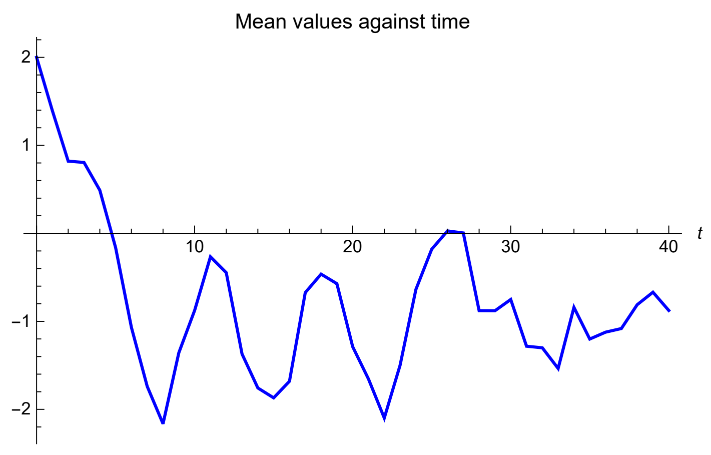
NDSolve: Warning: boundary and initial conditions are inconsistent.

NDSolve: Warning: scaled local spatial error estimate of 292.7798414400038` at t = 40.` in the direction of independent variable x is much greater than the prescribed error tolerance. Grid spacing with 25 points may be too large to achieve the desired accuracy or precision. A singularity may have formed or a smaller grid spacing can be specified using the MaxStepSize or MinPoints method options.

Out[59]= {


```
21.5026}
```

In[60]:= **meanvaluesplot[A, 0, newtimeend, Blue]**

Out[60]=


Above here we see that the stronger minimum $\phi \simeq -1$ is the more popular

9) Repeat section 7 with friction term, A_$\mu\ \partial\phi$^μ. I try A_μ = (fixed) function of t,x which I take to be constant in t,x. Then plot field mean value, field energy as a function of time (watch out running times). If we have chosen a proper friction term, the field will move slower and slower, fall into the two minima, and its energy will decrease through time

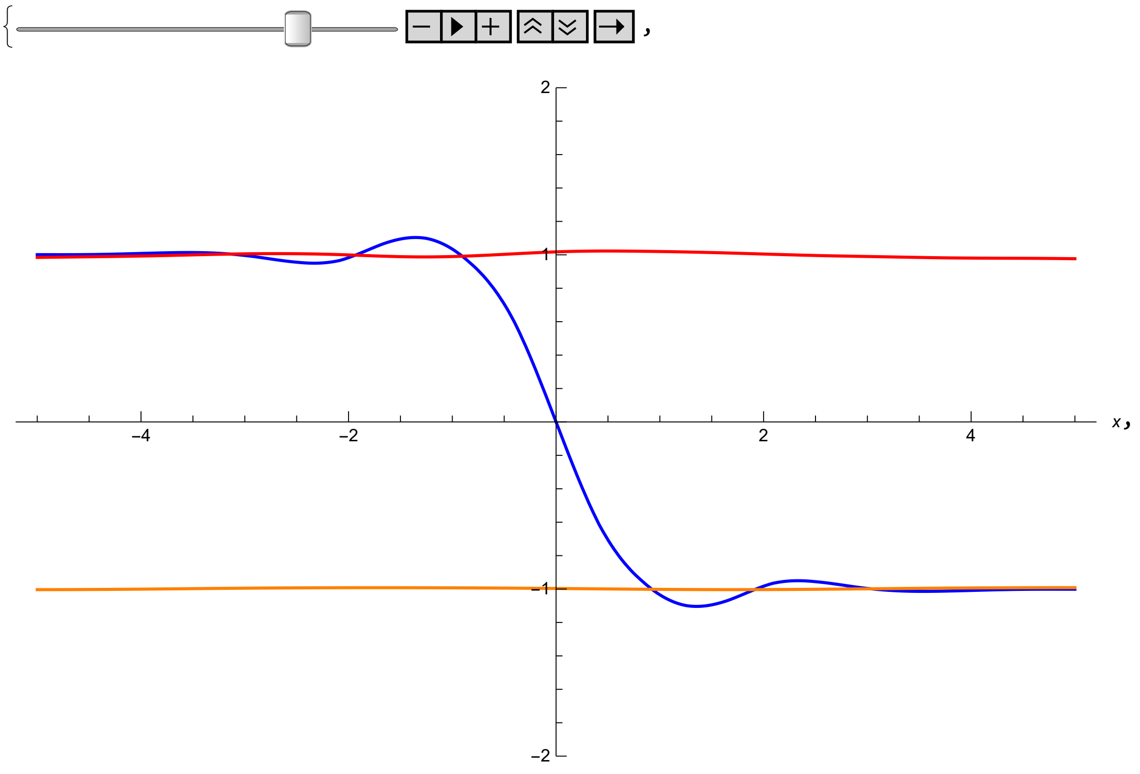
(Energy should be taken away by something external. this shows up when you try to write Lorentz-scalar friction terms involving the field only, which fails. E.g. a term $\pm\partial\phi^{*}\partial\phi$ in e.o.m. does not obstruct/slow down ϕ's motion.)
(! reminder: if you dont put the friction term: energy is conserved as long as you correctly set up

bdy/initial conditions. otherwise boundary terms dont cancel out when you integrate by parts)

```
In[61]:= At = +0.3;
Ax = 0;
FrEquation[ϕ_][t_, x_] :=
 Equation[ϕ][t, x] + At (Derivative[1, 0][ϕ][t, x]) + Ax (Derivative[0, 1][ϕ][t, x])
aafr = NDSolve[{FrEquation[ϕ][t, x] == 0, ϕ[0, x] == perturbb[x],
     Derivative[1, 0][ϕ][0, x] == 0, Derivative[0, 1][ϕ][t, -spaceend] == 0,
     Derivative[0, 1][ϕ][t, spaceend] == 0},
   ϕ, {t, 0, newtimeend}, {x, -spaceend, spaceend}];
bbfr = NDSolve[{FrEquation[ϕ][t, x] == 0, ϕ[0, x] == perturbb[x - 3],
     Derivative[1, 0][ϕ][0, x] == 0, Derivative[0, 1][ϕ][t, -spaceend] == 0,
     Derivative[0, 1][ϕ][t, spaceend] == 0},
   ϕ, {t, 0, newtimeend}, {x, -spaceend, spaceend}];
ccfr = NDSolve[{FrEquation[ϕ][t, x] == 0, ϕ[0, x] == 3 perturbb[x] - 0.1,
     Derivative[1, 0][ϕ][0, x] == 0, Derivative[0, 1][ϕ][t, -spaceend] == 0,
     Derivative[0, 1][ϕ][t, spaceend] == 0},
   ϕ, {t, 0, newtimeend}, {x, -spaceend, spaceend}];

DynamicModule[{t},
 {Animator[Dynamic[t], {0, newtimeend}], Dynamic[Plot[{Evaluate[ϕ[t, x] /. aafr],
      Evaluate[ϕ[t, x] /. bbfr], Evaluate[ϕ[t, x] /. ccfr]}, {x, -5, 5}, PlotRange → 2,
    ImageSize → Large, AxesLabel → Automatic, PlotStyle → colors]], Dynamic[t]}]
frsolutionscolors = {{aafr, colors[[1]]}, {bbfr, colors[[2]]}, {ccfr, colors[[3]]}};
```

```
··· NDSolve: Warning: scaled local spatial error estimate of 12.119331667611176` at t = 40.` in the direction of independent
    variable x is much greater than the prescribed error tolerance. Grid spacing with 25 points may be too large to achieve
    the desired accuracy or precision. A singularity may have formed or a smaller grid spacing can be specified using the
    MaxStepSize or MinPoints method options.

··· NDSolve: Warning: boundary and initial conditions are inconsistent.
```

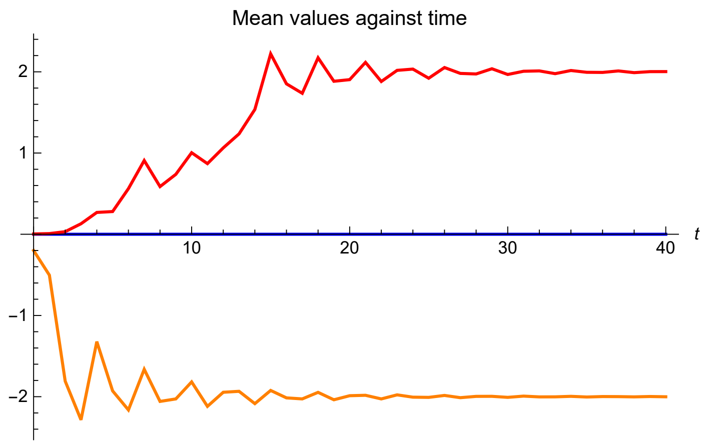
Out[67]= {



```
30.8584}
```

```
In[69]:= frplots = Map[meanvaluesplot[First[#], 0, newtimeend, Last[#]] &, frsolutionscolors];

In[70]:= Show[frplots, PlotRange → All]
```

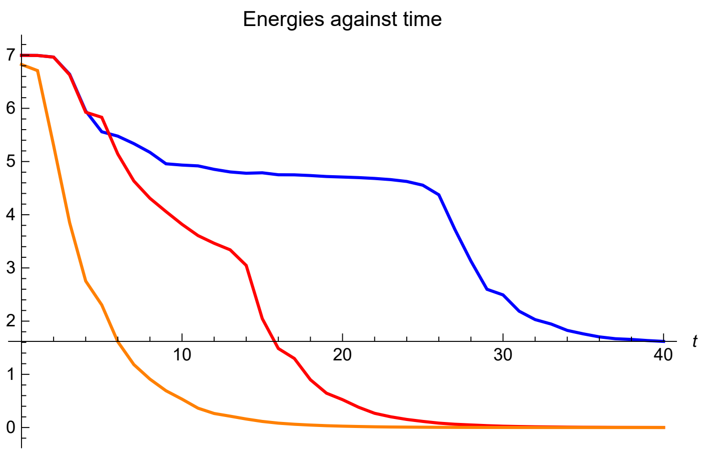
Out[70]=



Notice the spacially odd solution is prevented from falling in one of the two minima so it becomes a kink/antikink! (with non zero energy)
Now plot energy against time.

```
In[71]:= energies[t_, solution_] := NIntegrate[
           Evaluate[0.5 ((Derivative[1, 0][ϕ][t, x])^2 + (Derivative[0, 1][ϕ][t, x])^2) +
              V[ϕ[t, x]] /. solution], {x, -spaceend, spaceend}]
         energiesplot[solution_, integerstarttime_, integerendtime_, color_] :=
          DiscretePlot[energies[t, solution], {t, integerstarttime, integerendtime},
           Joined → True, PlotRange → All, Filling → None,
           PlotLabel → "Energies against time", AxesLabel → Automatic, PlotStyle → color]

         frenergyplots =
           Map[energiesplot[First[#], 0, newtimeend, Last[#]] &, frsolutionscolors];
         Show[frenergyplots, PlotRange → All]
```

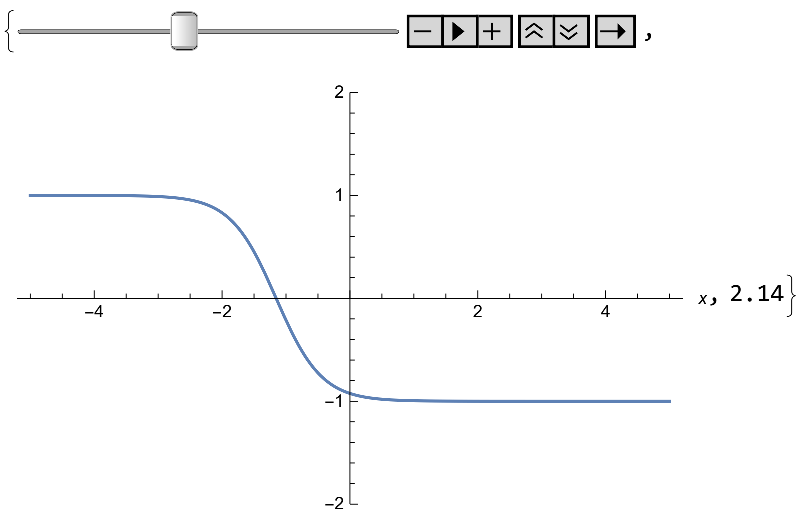
Out[74]=



10) antikink

```
In[75]:= ϕ0anti[x_] := -Tanh[Sqrt[2 λ] × γ[v] x];
         NDSolve[{Equation[ϕ][t, x] == 0,
            ϕ[0, x] == ϕ0anti[x], Derivative[1, 0][ϕ][0, x] == v ϕ0anti'[x],
            ϕ[t, -spaceend] == ϕ0anti[-spaceend], ϕ[t, spaceend] == ϕ0anti[spaceend]},
           ϕ, {t, 0, timeend}, {x, -spaceend, spaceend}];

         timeplot[Evaluate[ϕ[#1, #2] /. %] &, 0, timeend, -spaceend, spaceend]
```

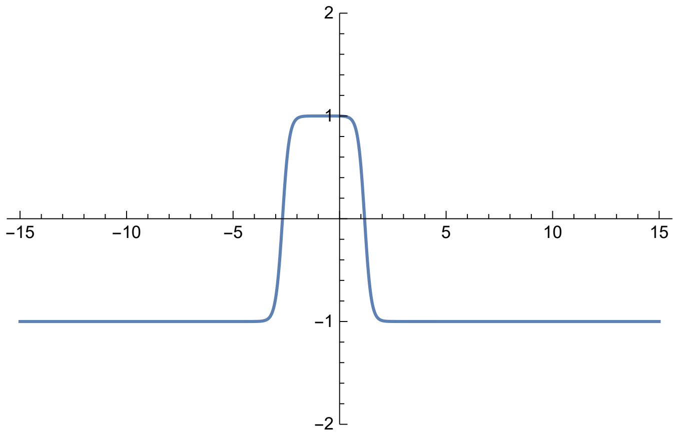
Out[77]= {



11) Scattering of kink coming from the left vs antikink coming from the right

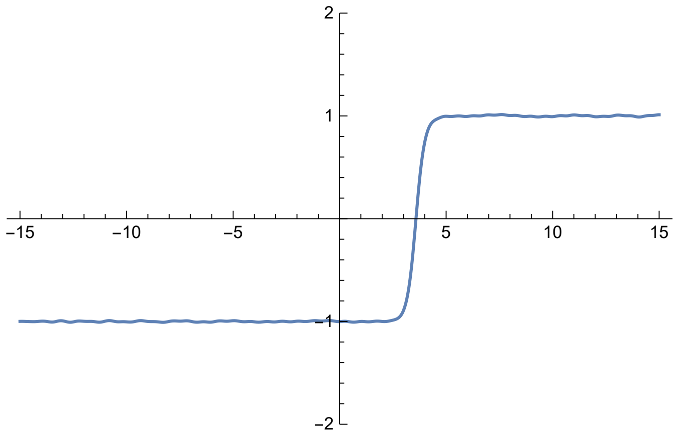
```
In[78]:= l = 5;
         newspaceend = 30;
         x0kink = -4;
         x0anti = 4;
         ϕ0kink[x_, vkink_] := Tanh[Sqrt[2 l] × γ[vkink] (x - x0kink)];
         ϕ0anti[x_, vanti_] := -Tanh[Sqrt[2 l] × γ[vanti] (x - x0anti)];
         ϕ0scatter[x_, vkink_, vanti_] := ϕ0kink[x, vkink] + ϕ0anti[x, vanti] - 1;

In[85]:= vkink = {0.1, 0.7, 0.7};
         vanti = {0.2, 0.1, 0.8};

In[87]:= solscatter = Table[NDSolve[{Equationcoup[ϕ, l][t, x] == 0 ,
              ϕ[0, x] == ϕ0scatter[x, vkink[[i]], vanti[[i]]], Derivative[1, 0][ϕ][0, x] ==
               -vkink[[i]] × Derivative[1, 0][ϕ0kink][x, vkink[[i]]] +
                vanti[[i]] × Derivative[1, 0][ϕ0anti][x, vanti[[i]]],
              Derivative[0, 1][ϕ][t, -newspaceend] == 0, Derivative[0, 1][ϕ][t, newspaceend] ==
               0}, ϕ, {t, 0, newtimeend}, {x, -newspaceend, newspaceend}], {i, 1, 3}];

     ··· NDSolve: Warning: scaled local spatial error estimate of 206.07725111937714` at t = 40.` in the direction of independent
           variable x is much greater than the prescribed error tolerance. Grid spacing with 1379 points may be too large to
           achieve the desired accuracy or precision. A singularity may have formed or a smaller grid spacing can be specified
           using the MaxStepSize or MinPoints method options.

     ··· NDSolve: Warning: scaled local spatial error estimate of 46.65988699011651` at t = 40.` in the direction of independent
           variable x is much greater than the prescribed error tolerance. Grid spacing with 1435 points may be too large to
           achieve the desired accuracy or precision. A singularity may have formed or a smaller grid spacing can be specified
           using the MaxStepSize or MinPoints method options.

     ··· NDSolve: Warning: scaled local spatial error estimate of 196.43333514796825` at t = 40.` in the direction of independent
           variable x is much greater than the prescribed error tolerance. Grid spacing with 1599 points may be too large to
           achieve the desired accuracy or precision. A singularity may have formed or a smaller grid spacing can be specified
           using the MaxStepSize or MinPoints method options.

     ··· General: Further output of NDSolve::eerr will be suppressed during this calculation.

In[88]:= command[j_] := {Print["vkink ", vkink[[j]]], Print["vantikink ", vanti[[j]]],
           timeplot[Evaluate[ϕ[#1, #2] /. solscatter[[j]] &],
            0, newtimeend, -newspaceend / 2, newspaceend / 2]}

In[89]:= command[1]
         command[2]
          command[3]

         vkink 0.1

         vantikink 0.2
```

```
Out[89]= {Null, Null, {
```



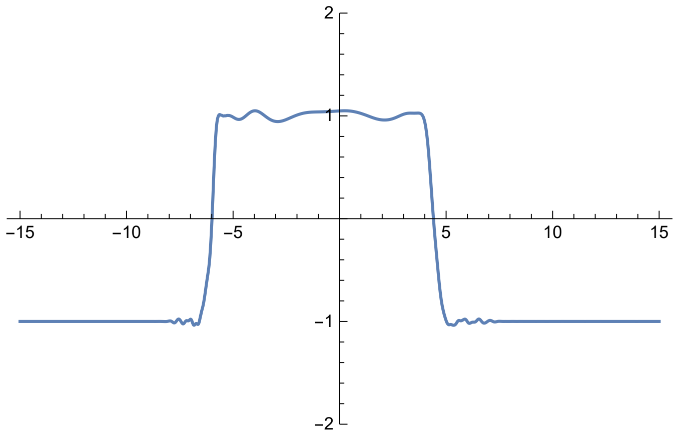
```
x, 13.0878}}
```

```
vkink 0.7
vantikink 0.1
Out[90]= {Null, Null, {
```



```
x, 32.5441}}
```

```
vkink 0.7
vantikink 0.8
Out[91]= {Null, Null, {
```



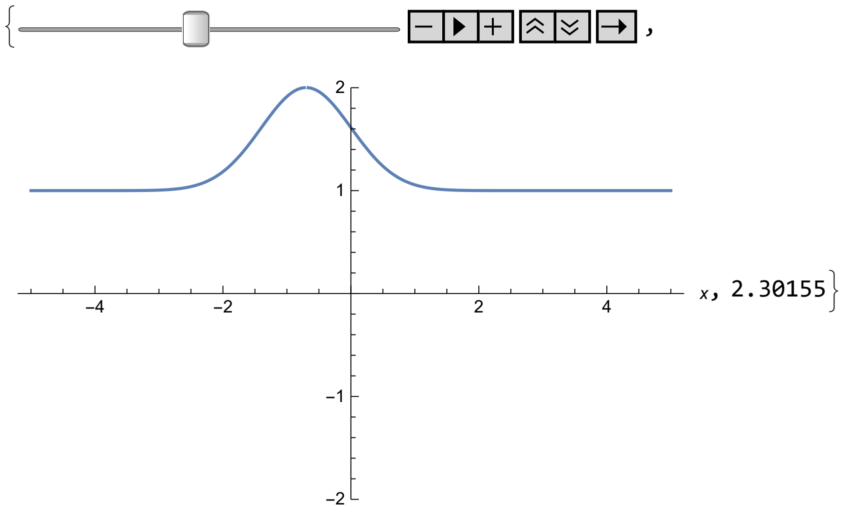
```
x, 13.3962}}
```

Outlook : study “out” velocities as functions of “in” velocities (also analytically, e.g. what is momentum conservation?)

12)packet traveling at speed of light, free theory

```
In[92]:= gauss[x_, c_] := c Exp[- (x + 3) ^2] + 1;
         freegauss = NDSolve[{Equationcoup[ϕ, 0][t, x] == 0 , ϕ[0, x] == gauss[x, 1],
              Derivative[1, 0][ϕ][0, x] == -1 Derivative[1, 0][gauss][x, 1],
              Derivative[0, 1][ϕ][t, -newspaceend] == 0, Derivative[0, 1][ϕ][t, newspaceend] == 0},
            ϕ, {t, 0, timeend}, {x, -newspaceend, newspaceend}];
         timeplot[Evaluate[ϕ[#1, #2] /. freegauss &], 0, timeend, -spaceend, spaceend]
```

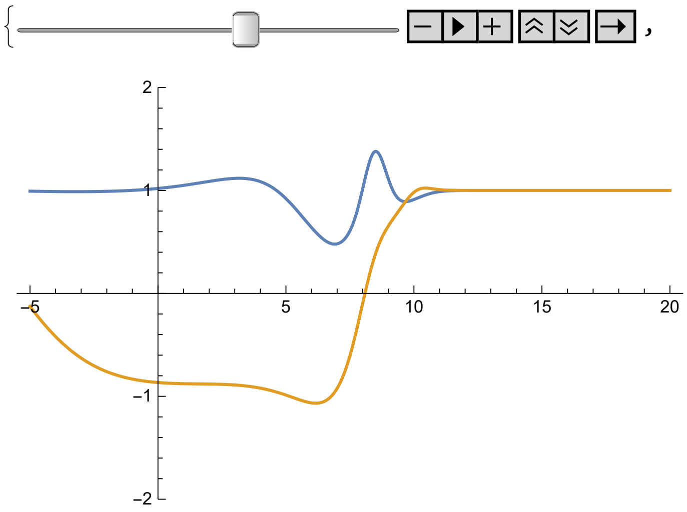
Out[94]= {



In[95]:=

13) packet starting at speed of light, interacting theory

```
In[96]:= c = -1;
         c1 = 1;
         λgauss = 0.1;
         coupledgauss = NDSolve[{Equationcoup[ϕ, λgauss][t, x] == 0 , ϕ[0, x] == gauss[x, c],
              Derivative[1, 0][ϕ][0, x] == -1 Derivative[1, 0][gauss][x, c],
              Derivative[0, 1][ϕ][t, -newspaceend] == 0, Derivative[0, 1][ϕ][t, 2 newspaceend] ==
               0}, ϕ, {t, 0, 4 timeend}, {x, -newspaceend, 2 newspaceend}];
         coupledgauss1 = NDSolve[{Equationcoup[ϕ, λgauss][t, x] == 0 , ϕ[0, x] == gauss[x, c1],
              Derivative[1, 0][ϕ][0, x] == -1 Derivative[1, 0][gauss][x, c1],
              Derivative[0, 1][ϕ][t, -newspaceend] == 0, Derivative[0, 1][ϕ][t, 2 newspaceend] ==
               0}, ϕ, {t, 0, 4 timeend}, {x, -newspaceend, 2 newspaceend}];
         doubletimeplot[{Evaluate[ϕ[#1, #2] /. coupledgauss &],
            Evaluate[ϕ[#1, #2] /. coupledgauss1 &]}, 0, 4 timeend, -spaceend, 4 spaceend]

Out[101]= {
```



```
x, 12.1921}
```

outlooks :

make two packets scatter

study spacial fourier trasform, ϕ(t,k)

14) 2 + 1 dimensions

!! ϕ(t,x,y) = f(x-t, y) is NOT solution for the free theory!! unless $\partial_{yy} f = 0$

So the free equation is interesting in its own right.

Problem: big spaceends require much computation and results don’t look good. Bdy conditions must then be chosen carefully. I try periodic bdy conditions.

Next and last input may require some time to run

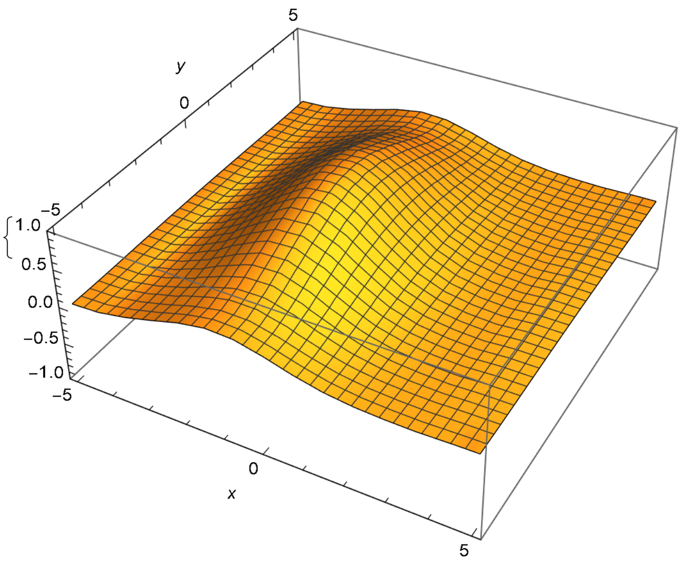
```
In[102]:= λ2D = 0;

spaceendx = 5;
spaceendy = 5;
timeend2D = 10;
xstart = -2;
spreadx = 2;
spready = 4;
gauss2D[x_, y_] := Exp[-(((x - xstart) / spreadx) ^2 + (y / spready) ^2)];
"Plot3D[gauss2D[x,y],{x,-spaceendx,spaceendx},{y,-spaceendy,spaceendy},PlotRange→{-
  0.2,1.2},ImageSize→Large,AxesLabel → Automatic,Mesh→30]"
Equation3D[ϕ_][t_, x_, y_] :=
 Derivative[2, 0, 0][ϕ][t, x, y] - Derivative[0, 2, 0][ϕ][t, x, y] -
  Derivative[0, 0, 2][ϕ][t, x, y] + Derivative[1, 0][Vcoup][ϕ[t, x, y], λ2D]
Clear[sol3D];
"VON NEUMANN sol3D=NDSolve[{Equation3D[ϕ][t,x,y]==0
   ,ϕ[0,x,y]==gauss2D[x,y],Derivative[1,0,0][ϕ][0,x,y]==-1
   Derivative[1,0][gauss2D][x,y],
   Derivative[0,1,0][ϕ][t,-spaceendx,y]==0,Derivative[0,1,0][ϕ][t,spaceendx,y]==0,
   Derivative[0,0,1][ϕ][t,x,-spaceendy]==0,Derivative[0,0,1][ϕ][t,x,spaceendy]==0},ϕ
   ,{t,0,timeend2D},{x,-spaceendx,spaceendx},{y,-spaceendy,spaceendy}];";
sol3Dperiodic = NDSolve[{Equation3D[ϕ][t, x, y] == 0 , ϕ[0, x, y] == gauss2D[x, y],
    Derivative[1, 0, 0][ϕ][0, x, y] == -1 Derivative[1, 0][gauss2D][x, y],
    ϕ[t, -spaceendx, y] == ϕ[t, spaceendx, y], ϕ[t, x, -spaceendy] == ϕ[t, x, spaceendy]},
   ϕ, {t, 0, timeend2D}, {x, -spaceendx, spaceendx}, {y, -spaceendy, spaceendy}];

Out[110]= Plot3D[gauss2D[x,y],{x,-spaceendx,spaceendx},{y,-spaceendy,spaceendy},PlotRange→{-0.2
  ,1.2},ImageSize→Large,AxesLabel → Automatic,Mesh→30]

••• NDSolve: Using maximum number of grid points 100 allowed by the MaxPoints or MinStepSize options for independent
      variable x.

••• NDSolve: Using maximum number of grid points 100 allowed by the MaxPoints or MinStepSize options for independent
      variable y.

••• NDSolve: Using maximum number of grid points 100 allowed by the MaxPoints or MinStepSize options for independent
      variable x.

••• General: Further output of NDSolve::mxsst will be suppressed during this calculation.

••• NDSolve: Warning: boundary and initial conditions are inconsistent.

••• NDSolve: Warning: scaled local spatial error estimate of 139.09960174861925` at t = 10.` in the direction of independent
      variable x is much greater than the prescribed error tolerance. Grid spacing with 101 points may be too large to
      achieve the desired accuracy or precision. A singularity may have formed or a smaller grid spacing can be specified
      using the MaxStepSize or MinPoints method options.
```

Leaving the time-dependent 3D plot animation in comment bc it doesn' t work well:
ANIMATIONPrint[Style[“YOUR MATHEMATICA IS PROBABLY SLOWING DOWN A LOT BECAUSE YOU ARE SEEING THIS ANIMATION”,24,Red]]
DynamicModule[{t},{Animator[Dynamic[t],{0,timeend2D,0.2}],
Dynamic[Plot3D[Evaluate[ϕ[t,x,y]/.sol3D],{x,-spaceendx,spaceendx},{y,-spaceendy,spaceendy},-PlotRange→{-1,1},ImageSize→Large,AxesLabel → Automatic,Mesh→30]],Dynamic[N[t]]}]
Print[Style[“YOUR MATHEMATICA IS PROBABLY SLOWING DOWN A LOT BECAUSE YOU ARE SEEING THIS ANIMATION”,24,Red]]

```
In[115]:= timestep = 1;
          command3D[j_] :=
           Plot3D[Evaluate[ϕ[j * timestep, x, y] /. sol3Dperiodic], {x, -spaceendx, spaceendx},
             {y, -spaceendy, spaceendy}, PlotRange → {-1, 1}, ImageSize → Medium,
            AxesLabel → Automatic, Mesh → 30, PlotLabel → {"time", j * timestep}]
          Table[command3D[j], {j, 1, 10}]
```

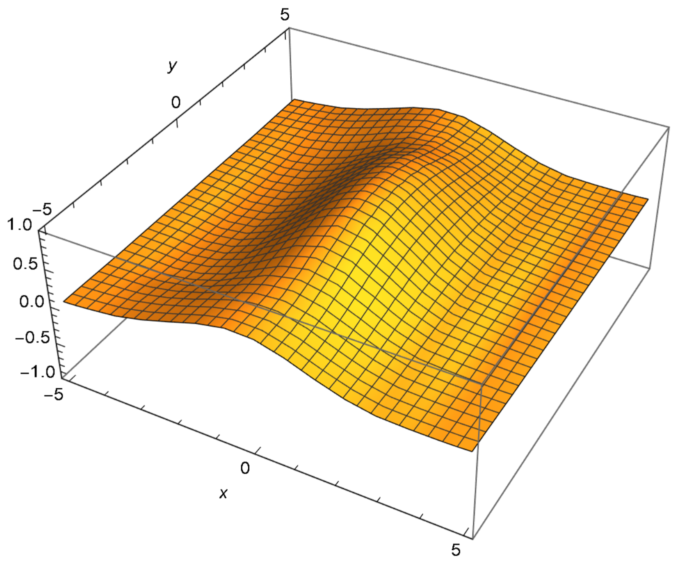
Out[117]= {

{time, 1}



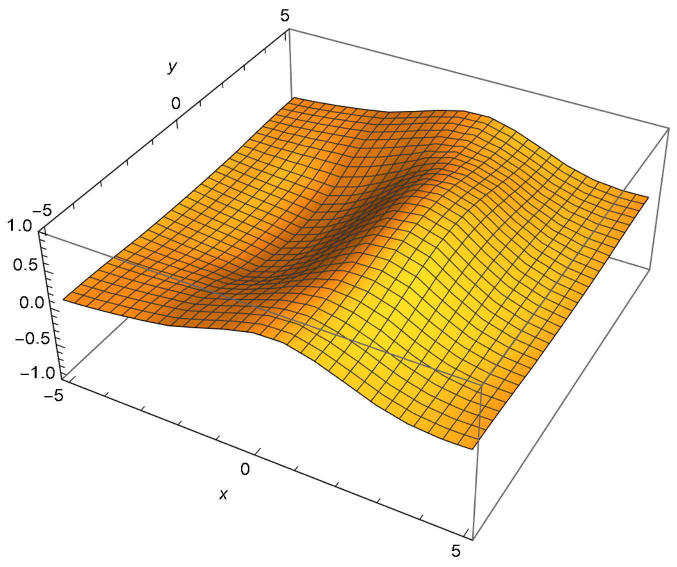
,

{time, 2}



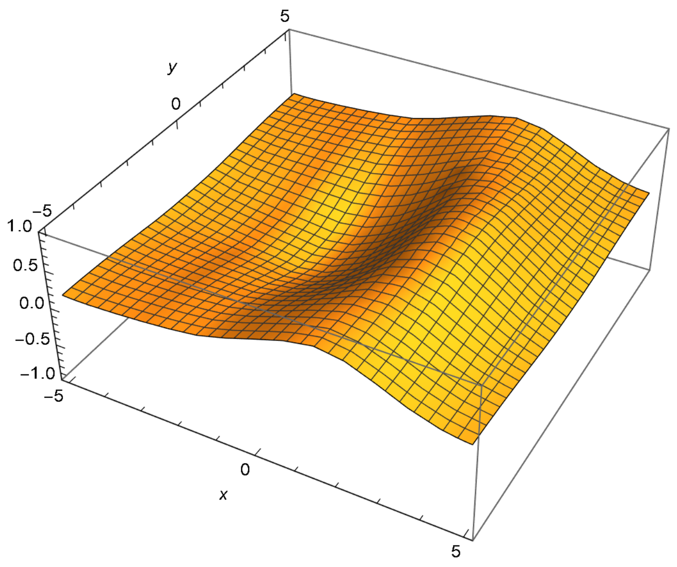
,

{time, 3}



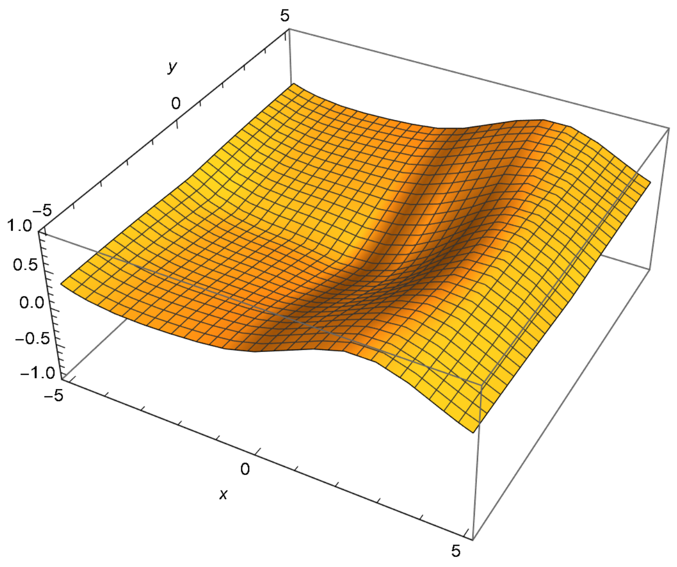
,

{time, 4}



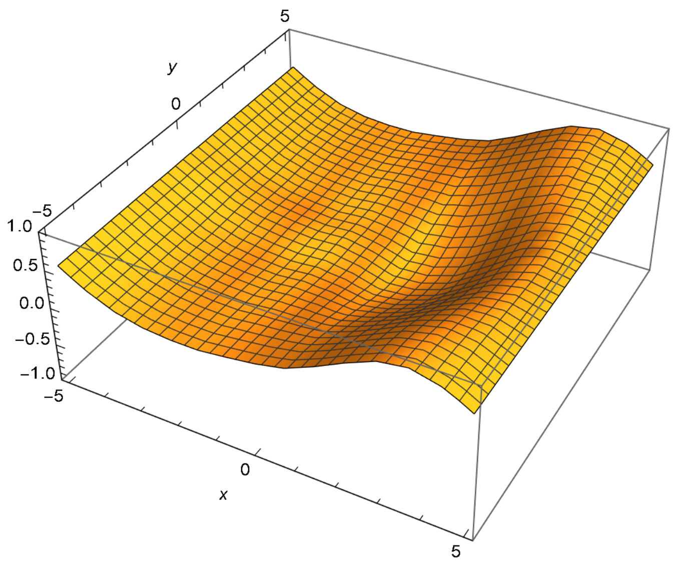
,

{time, 5}



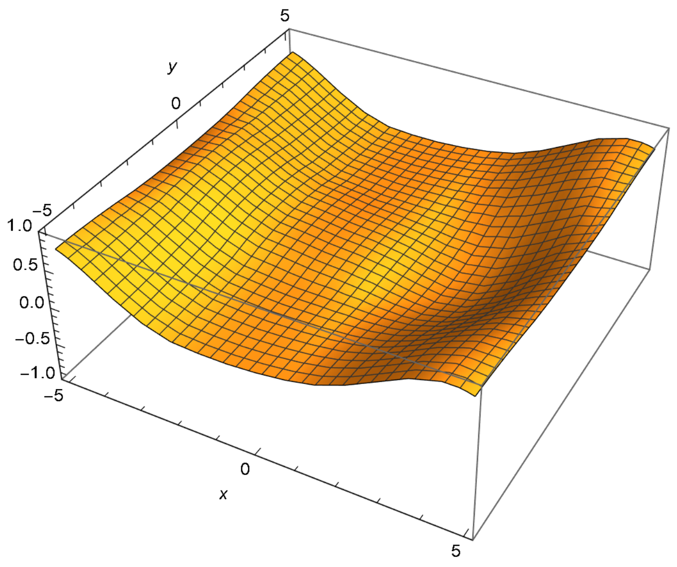
,

{time, 6}



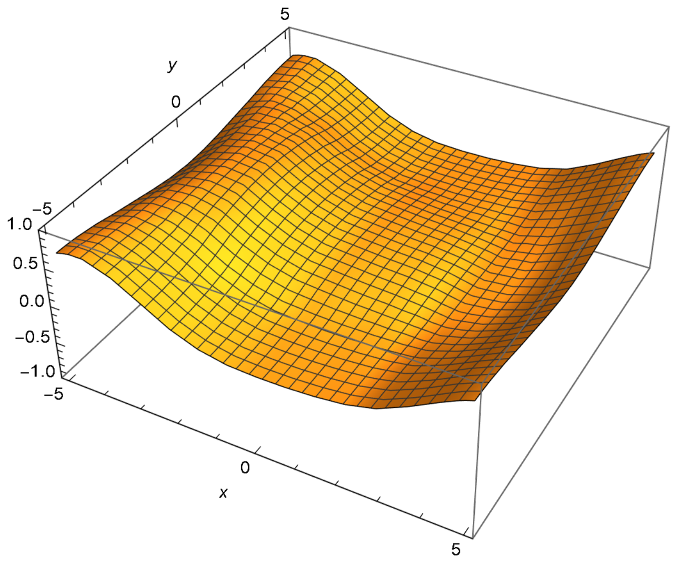
,

{time, 7}



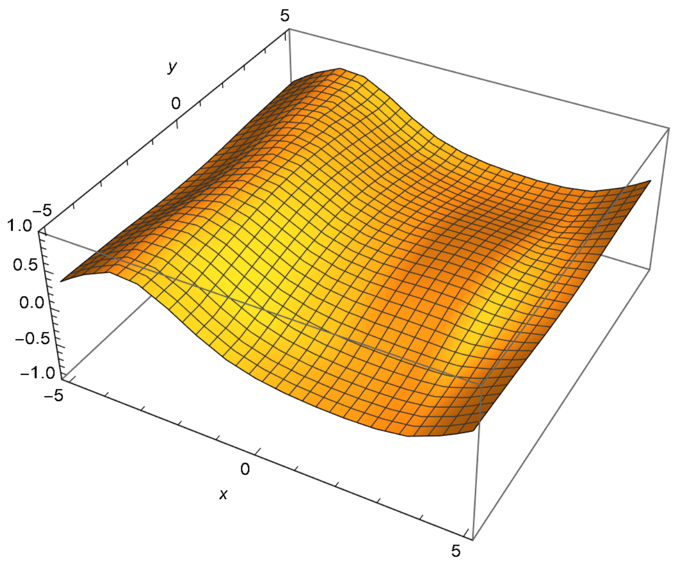
,

{time, 8}



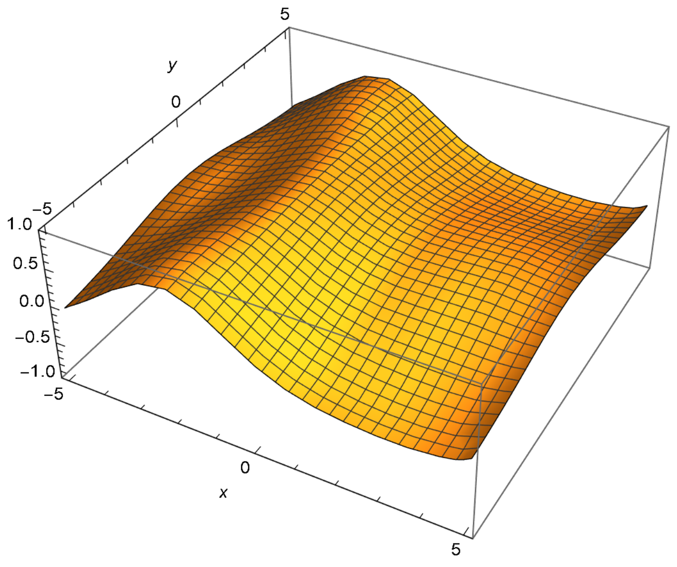
,

{time, 9}



,

{time, 10}



}