X-Mas Tunneling

Definiere Anfangswellenpaket:

In[1]:=
```
ψ0[x_, x0_, a_, p_] := (π a^2)^(-1/4) Exp[- (x - x0)^2 / (2 a^2) + i p x]
```

Teste Wellenfunktion auf Normiertheit:

In[2]:=
```
Integrate[Abs[ψ0[x, 0, .5, 0]]^2, {x, -Infinity, Infinity}]
```

Out[2]=
```
1.
```

Definiere Funktion zum Lösen der Schrödingergleichung:

Input:

schrodinger	...	zu lösende Schrödingergleichung (Hamiltonian eingesetzt)
x0	...	Startposition des Wellenpakets
a	...	Startbreite des Wellenpakets
p	...	Startimpuls
xmax	...	Größe des Systems (in positive und negative Richtung)
tmax	...	Dauer der Simulation

Offene Randbedingungen:

In[3]:=
```
schrodingerSolve[schrodinger_, x0_, a_, p_, xmax_, tmax_] :=
  NDSolve[
   {schrodinger,
    ψ[x, 0] == ψ0[x, x0, a, p],
    ψ[xmax, t] == 0, ψ[-xmax, t] == 0},
   ψ,
   {x, -xmax, xmax},
   {t, 0, tmax}
  ];
```

Periodische Randbedingungen:

In[4]:=
```
schrSolvePeriodic[schrodinger_, x0_, a_, p_, xmax_, tmax_] :=
  NDSolve[
   {schrodinger,
    ψ[x, 0] == ψ0[x, x0, a, p],
    ψ[xmax, t] == ψ[-xmax, t]},
   ψ,
   {x, -xmax, xmax},
   {t, 0, tmax}
  ];
```

Schmelzende Weihnachtsmänner und Elfen ohne Potential

Definiere Systemparameter:

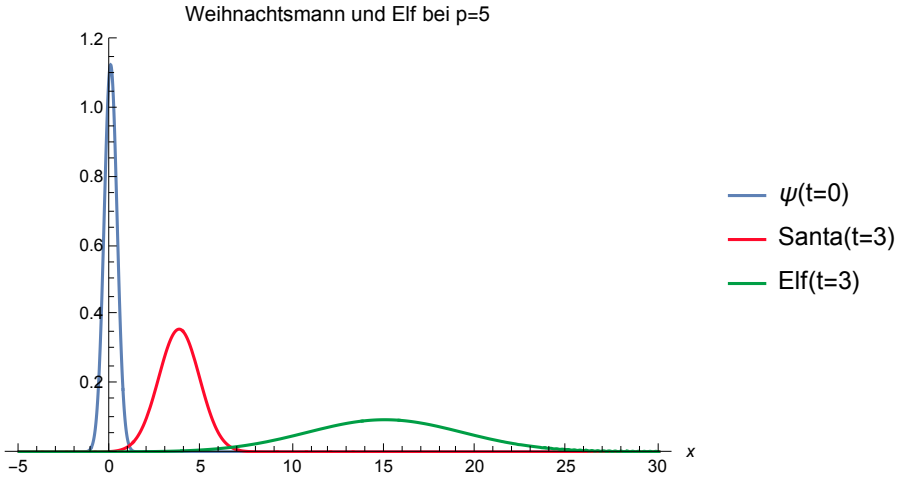
In[5]:=
```
(*Größe des Systems*)
size = 30;
(*Massen*)
santaMass = 4; elfMass = 1; rudolfMass = 1; ballMass = .2;
(*Dauer der Simulation*)
maxtimeSanta = 20; maxtimeElf = 4; maxtimeRudolf = 1; maxtimeBall = 1.5;
(*Anfangsbreite des Wellenpakets*)
initwidth = .5;
(*Anfangsimpuls*)
santaP = 5; rudolfP = 20;
(*Anfangsposition des Wellenpakets*)
start = 0;
```

Löse Schrödingergleichung:

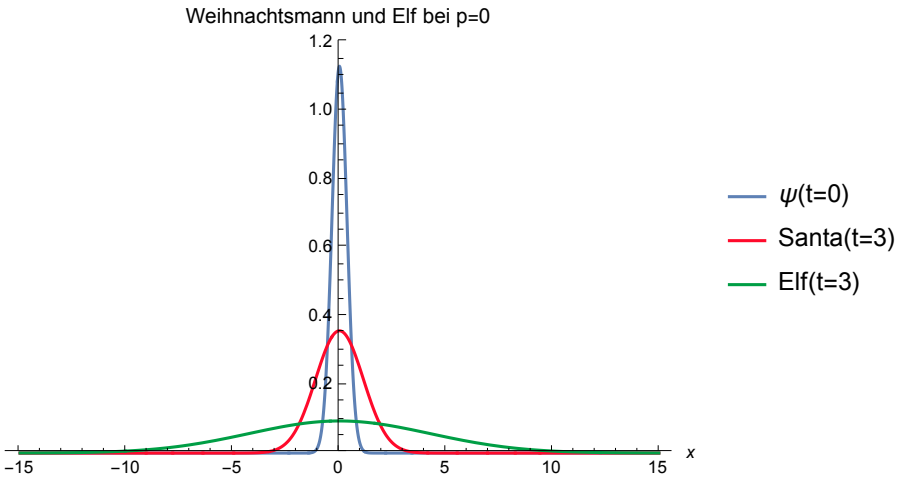
In[11]:=
```
schrFreeSanta = -1 / (2 santaMass) D[ψ[x, t], x, x] == i D[ψ[x, t], t];
schrFreeElf = -1 / (2 elfMass) D[ψ[x, t], x, x] == i D[ψ[x, t], t];

freeSanta =
  schrodingerSolve[schrFreeSanta, start, initwidth, santaP, size, maxtimeSanta];

freeElf =
  schrodingerSolve[schrFreeElf, start, initwidth, santaP, size, maxtimeElf];

freeSantaP0 =
  schrodingerSolve[schrFreeSanta, start, initwidth, 0, size, maxtimeSanta];

freeElfP0 = schrodingerSolve[schrFreeElf, start, initwidth, 0, size, maxtimeElf];
```

Darstellung der Lösung:

```
Plot[
 {Abs[ψ[x, 0]]^2 /. freeSanta,
  Abs[ψ[x, 3]]^2 /. freeSanta,
  Abs[ψ[x, 3]]^2 /. freeElf},
 {x, -5, 30},
 PlotRange → {0, 1.2},
 PlotStyle → {None, True, RGBColor[1., 0.04, 0.17], RGBColor[0., 0.62, 0.27]},
 PlotLegends → {"ψ(t=0)", "Santa(t=3)", "Elf(t=3)"}, AxesLabel → {x, None},
 PlotLabel → "Weihnachtsmann und Elf bei p=5"]
```




```
Plot[
 {Abs[ψ[x, 0]]^2 /. freeSantaP0,
  Abs[ψ[x, 3]]^2 /. freeSantaP0,
  Abs[ψ[x, 3]]^2 /. freeElfP0},
 {x, -15, 15},
 PlotRange → {0, 1.2},
 PlotStyle → {None, True, RGBColor[1., 0.04, 0.17], RGBColor[0., 0.62, 0.27]},
 PlotLegends → {"ψ(t=0)", "Santa(t=3)", "Elf(t=3)"}, AxesLabel → {x, None},
 PlotLabel → "Weihnachtsmann und Elf bei p=0"]
```



Zu beobachten ist, dass sowohl der Weihnachtsmann als auch der Elf mit der Zeit “zerfließen”, also immer unschärfer im Ort werden. Dabei bewegt sich der Elf bei gleichem Impuls schneller und schmilzt auch schneller dahin als der Weihnachtsmann. Das Zerfließen lässt sich dabei auch bei unbewegtem Weihnachtsmann und Elf beobachten. Die Geschwindigkeit der Unschärfezunahme im ist demnach abhängig von der Masse des Teilchens.

Bestimme die Geschwindigkeit des Weihnachtsmannes:
- durch Ermitteln des Maximums der Wellenfunktion an einem festen Zeitpunkt bei variablem Impuls

```
p1 = 1;
p2 = 3;
p3 = 5;
freeSanta1 =
  schrodingerSolve[schrFreeSanta, start, initwidth, p1, size, maxtimeSanta];
freeSanta2 = schrodingerSolve[schrFreeSanta,
   start, initwidth, p2, size, maxtimeSanta];
freeSanta3 = schrodingerSolve[schrFreeSanta, start,
   initwidth, p3, size, maxtimeSanta];
```

Auswertung des Maximums bei t=3:

```
x1 = x /. Last[FindMaximum[Abs[ψ[x, 3]]^2 /. freeSanta1, {x, -size, size}][[2]]];
x2 = x /. Last[FindMaximum[Abs[ψ[x, 3]]^2 /. freeSanta2, {x, -size, size}][[2]]];
x3 = x /. Last[FindMaximum[Abs[ψ[x, 3]]^2 /. freeSanta3, {x, -size, size}][[2]]];
```

Berechne Geschwindigkeiten:

```
v1 = x1 / 3;
v2 = x2 / 3;
v3 = x3 / 3;
```

Berechne Proportionalitätskonstante:

```
m1 = p1 / v1
m2 = p2 / v2
m3 = p3 / v3
mMean = (m1 + m2 + m3) / 3
```

```
3.99361
```

```
3.99753
```

```
3.99963
```

```
3.99692
```

Welch Überraschung!!! Die Proportionalitätskonstante zwischen Impuls und Geschwindigkeit scheint die Masse zu sein. Ungenauigkeiten durch das numerische Lösungsverfahren sind erwartbar.

Bewegung unter Einfluss eines Potentials:

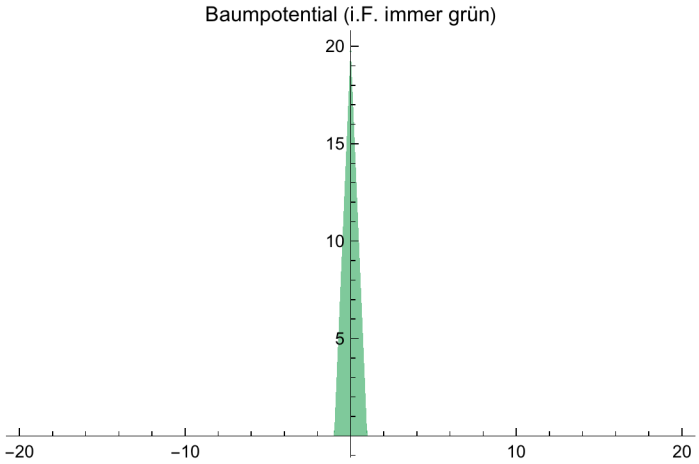
Definiere Potential:

In[13]:=
```
USingleTree[x_, x0_, h_, w_] :=
 Piecewise[{{0, x < x0 - w / 2}, {(x - x0) * 2 h / w + h, x0 - w / 2 ≤ x ≤ x0},
   {-(x - x0 - w / 2) * 2 h / w, x0 ≤ x ≤ x0 + w / 2}, {0, x > x0 + w / 2}}]
```

Parameter des Baumes:

In[90]:=
```
barrierpos = 0; treeheight = 20; treewidth = 2;
(*Ich hoffe du kannst mir verzeihen,
dass ich vergessen habe die Baumbreite zu
 verändern. Es ist schon recht spät und ich habe nicht vor
 die Wellenfunktionen nochmal neu berechnen zu lassen.*)
SingleTreePot = USingleTree[x, barrierpos, treeheight, treewidth];

Plot[SingleTreePot,
 {x, -20, 20},
 PlotRange → Full,
 Filling → Axis,
 FillingStyle → Directive[RGBColor[0., 0.58, 0.22], Opacity[.5]],
 PlotStyle → None,
 PlotLabel → "Baumpotential (i.F. immer grün)"]
```

Out[92]=



Schrödingergleichungen:

In[85]:=
```
schrTreeSanta =
  -1 / (2 santaMass) D[ψ[x, t], x, x] + SingleTreePot * ψ[x, t] == i D[ψ[x, t], t];
schrTreeElf = -1 / (2 elfMass) D[ψ[x, t], x, x] + SingleTreePot * ψ[x, t] ==
   i D[ψ[x, t], t];
schrTreeRudolf = -1 / (2 rudolfMass) D[ψ[x, t], x, x] +
    SingleTreePot * ψ[x, t] == i D[ψ[x, t], t];
schrTreeBall = -1 / (2 ballMass) D[ψ[x, t], x, x] + SingleTreePot * ψ[x, t] ==
   i D[ψ[x, t], t];
```

numerische Lösungen mit offenen Randbedingungen:

```
treeSanta =
  schrodingerSolve[schrTreeSanta, -10, initwidth, santaP, size, maxtimeSanta];
```

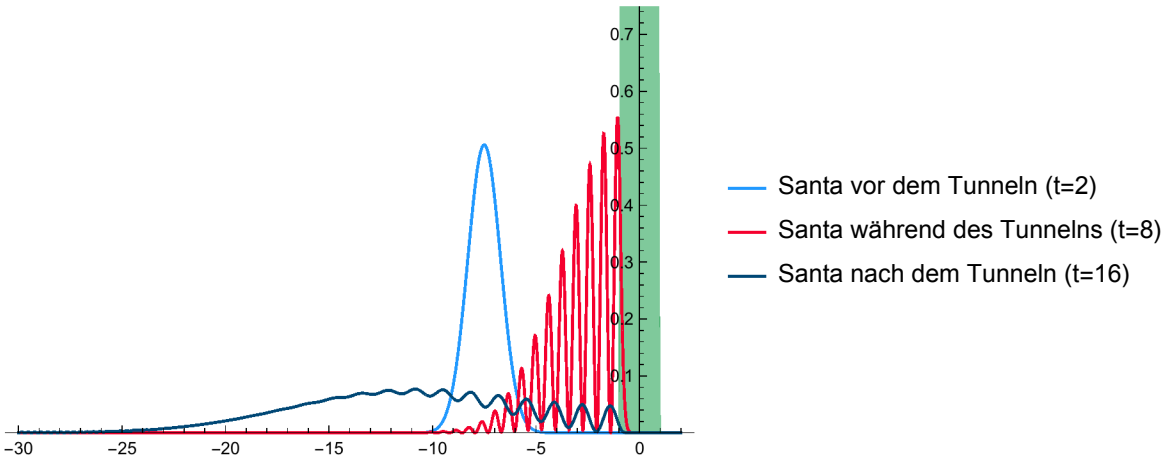
In[93]:=
```
treeElf = schrodingerSolve[schrTreeElf, -10, initwidth, santaP, size, maxtimeElf];
```

```
treeRudolf =
  schrodingerSolve[schrTreeRudolf, -10, initwidth, rudolfP, size, maxtimeRudolf];
```

```
treeBall =
  schrodingerSolve[schrTreeBall, -10, initwidth, santaP, 50, maxtimeBall];
```

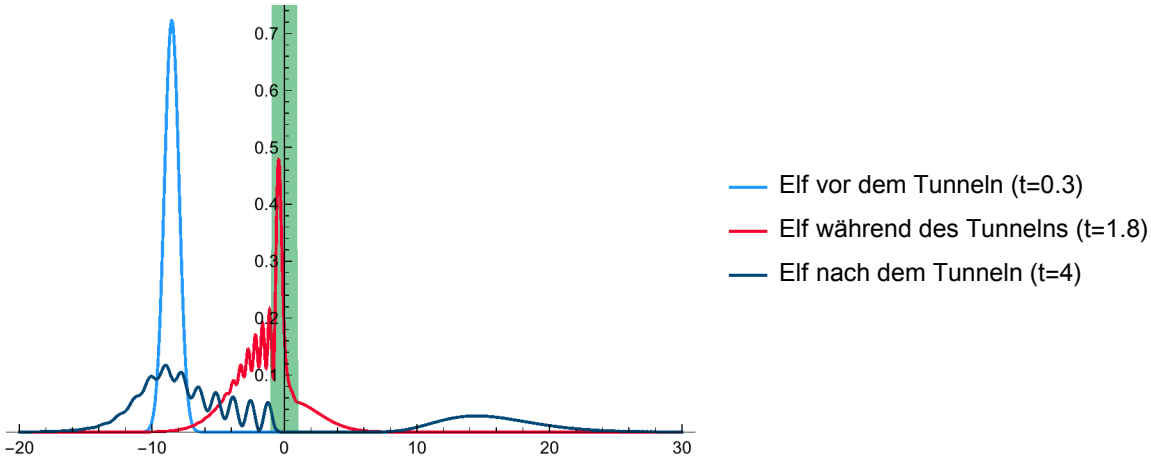
Numerische Lösungen mit periodischen Randbedinungen:

```
treeSantaPer =
  schrSolvePeriodic[schrTreeSanta, -10, initwidth, santaP, size, maxtimeSanta];

treeElfPer =
  schrSolvePeriodic[schrTreeElf, -10, initwidth, santaP, size, maxtimeElf];

treeRudolfPer = schrSolvePeriodic[
   schrTreeRudolf, -10, initwidth, rudolfP, size, maxtimeRudolf];

treeBallPer =
  schrSolvePeriodic[schrTreeBall, -10, initwidth, santaP, size, maxtimeBall];

Plot[
 {SingleTreePot,
  Abs[ψ[x, 2]]^2 /. treeSanta,
  Abs[ψ[x, 8]]^2 /. treeSanta,
  Abs[ψ[x, 16]]^2 /. treeSanta},
 {x, -30, 2},
 PlotRange → {0, .75},
 Filling → {1 → Axis},
 FillingStyle → Directive[RGBColor[0., 0.58, 0.22], Opacity[.5]],
 PlotStyle → {None, RGBColor[0.13, 0.6, 1.],
   RGBColor[0.96, 0., 0.18], RGBColor[0., 0.28, 0.46]},
 PlotLegends → {None, "Santa vor dem Tunneln (t=2)",
   "Santa während des Tunnelns (t=8)", "Santa nach dem Tunneln (t=16)"}]
```

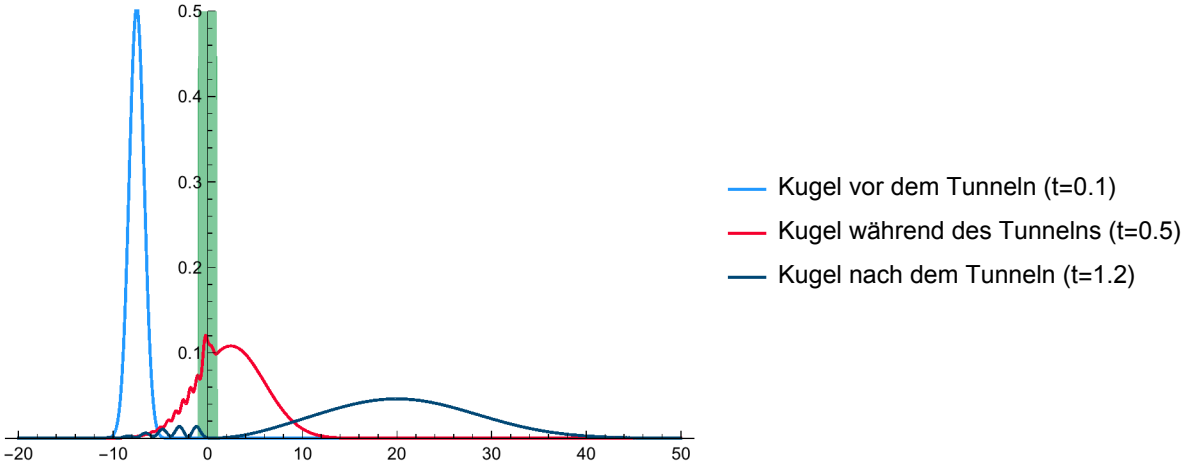


In[95]:=
```
Plot[
 {SingleTreePot,
  Abs[ψ[x, .3]]^2 /. treeElf,
  Abs[ψ[x, 1.8]]^2 /. treeElf,
  Abs[ψ[x, 4]]^2 /. treeElf},
 {x, -20, 30},
 PlotRange → {0, .75},
 Filling → {1 → Axis},
 FillingStyle → Directive[RGBColor[0., 0.58, 0.22], Opacity[.5]],
 PlotStyle → {None, RGBColor[0.13, 0.6, 1.],
   RGBColor[0.96, 0., 0.18], RGBColor[0., 0.28, 0.46]},
 PlotLegends → {None, "Elf vor dem Tunneln (t=0.3)",
   "Elf während des Tunnelns (t=1.8)", "Elf nach dem Tunneln (t=4)"}]
```

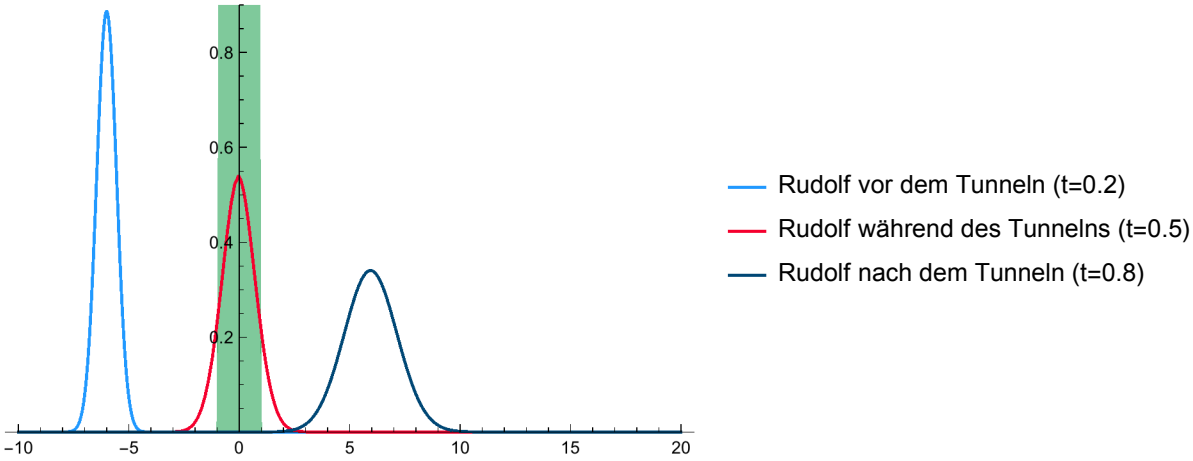
Out[95]=



```
Plot[
 {SingleTreePot,
  Abs[ψ[x, .1]]^2 /. treeBall,
  Abs[ψ[x, .5]]^2 /. treeBall,
  Abs[ψ[x, 1.2]]^2 /. treeBall},
 {x, -20, 50},
 PlotRange → {0, .5},
 Filling → {1 → Axis},
 FillingStyle → Directive[RGBColor[0., 0.58, 0.22], Opacity[.5]],
 PlotStyle → {None, RGBColor[0.13, 0.6, 1.],
   RGBColor[0.96, 0., 0.18], RGBColor[0., 0.28, 0.46]},
 PlotLegends → {None, "Kugel vor dem Tunneln (t=0.1)",
   "Kugel während des Tunnelns (t=0.5)", "Kugel nach dem Tunneln (t=1.2)"}]
```



```
Plot[
 {SingleTreePot,
  Abs[ψ[x, .2]]^2 /. treeRudolf,
  Abs[ψ[x, .5]]^2 /. treeRudolf,
  Abs[ψ[x, .8]]^2 /. treeRudolf},
 {x, -10, 20},
 PlotRange → {0, .9},
 Filling → {1 → Axis},
 FillingStyle → Directive[RGBColor[0., 0.58, 0.22], Opacity[.5]],
 PlotStyle → {None, RGBColor[0.13, 0.6, 1.],
   RGBColor[0.96, 0., 0.18], RGBColor[0., 0.28, 0.46]},
 PlotLegends → {None, "Rudolf vor dem Tunneln (t=0.2)",
   "Rudolf während des Tunnelns (t=0.5)", "Rudolf nach dem Tunneln (t=0.8)"}]
```



Ermittle Transmissionskoeffizienten:

- durch numerische Integration der Wellenfunktion nach dem Tunneln hinter der Potentialbarriere (x>0)

Weihnachtsmann (t=16):

```
Sqrt[NIntegrate[Abs[ψ[x, 16]]^2 /. treeSanta, {x, 0, size}]]
```

```
{0.00166422}
```

Elf (t=4):

```
Sqrt[NIntegrate[Abs[ψ[x, 4]]^2 /. treeElf, {x, 0, size}]]
```

```
{0.498803}
```

Christbaumkugel (t=1.2):

```
Sqrt[NIntegrate[Abs[ψ[x, 1.2]]^2 /. treeBall, {x, 0, 50}]]
```

```
{0.971312}
```

Rudolf (t=0.8):

```
Sqrt[NIntegrate[Abs[ψ[x, .8]]^2 /. treeRudolf, {x, 0, size}]]
```

```
{1.01205}
```

Die Transmissionskoeffizienten und die Graphen zeigen, dass die Christbaumkugel und Rudolf die besten Tunneleigenschaften haben. Der Weihnachtsmann tunnelt besser als der Elf. Daran erkennt

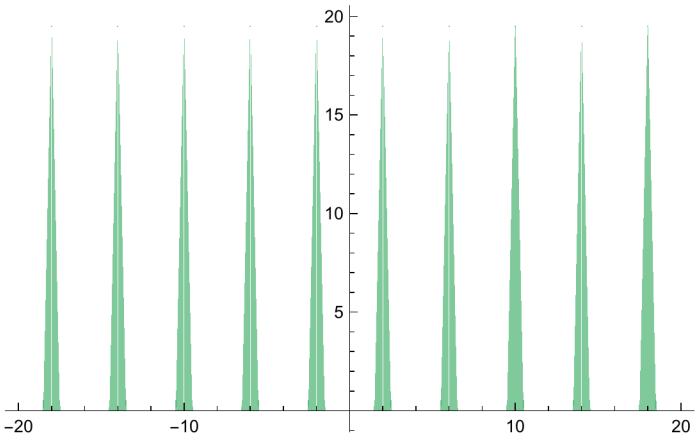
man, dass bei gleichem Impuls der Transmissionskoeffizient eines Teilchens mit kleinerer Masse größer ist. Rudolf, der die gleiche Masse hat, wie der Elf, tunnelt mit seinem größeren Impuls dabei besser durch die Potentialbarriere.
Wählt man periodische statt offene Randbedingungen, bewegt sich das Wellenpaket von ohne Interferenz am Rand von einem Ende des Systems zum anderen, während es bei offenen Randbedingungen am Rand reflektiert wird und mit sich selbst interferiert.

Rennt'n Elf inne Baumschule...

Definiere Baumschulenpotential:

In[22]:=
```
UBaumschule[x_, s_, n_, h_, w_] :=
  Sum[USingleTree[x, 2 * s / n * i, h, w], {i, -n / 2 + .5, n / 2 - .5}];
BaumschulePot = UBaumschule[x, 40, 20, 20, 1];
Plot[BaumschulePot,
 {x, -20, 20},
 PlotRange → Full,
 Filling → Axis,
 FillingStyle → Directive[RGBColor[0., 0.58, 0.22], Opacity[.5]],
 PlotStyle → None]
```

Out[24]=



Schrödingergleichung:

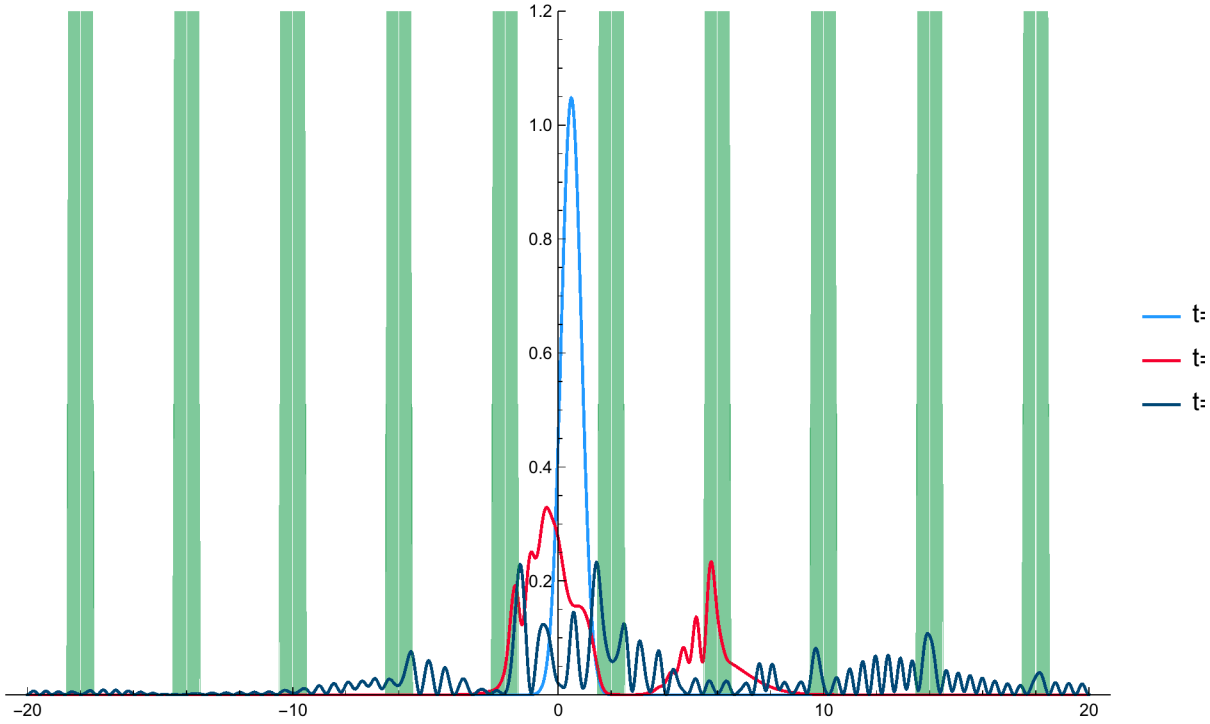
In[17]:=
```
schrBsElf =
  -1 / (2 elfMass) D[ψ[x, t], x, x] + BaumschulePot * ψ[x, t] == i D[ψ[x, t], t];
```

Lösung:

In[18]:=
```
BsElf = schrodingerSolve[schrBsElf, 0, initwidth, santaP, 20, maxtimeElf];
```

In[130]:=
```
Plot[
 {BaumschulePot,
  Abs[ψ[x, .1]]^2 /. BsElf,
  Abs[ψ[x, 1]]^2 /. BsElf,
  Abs[ψ[x, 4]]^2 /. BsElf},
 {x, -20, 20},
 PlotRange → {0, 1.2},
 Filling → {1 → Axis},
 FillingStyle → Directive[RGBColor[0., 0.58, 0.22], Opacity[.5]],
 PlotStyle → {None, RGBColor[0.13, 0.6, 1.], RGBColor[0.96, 0., 0.18],
   RGBColor[0., 0.28, 0.46](*,RGBColor[1.,0.55,0.19]*)},
 PlotLegends → {None, "t=0.1", "t=1", "t=4"},
 ImageSize → Large]
```

Out[130]=

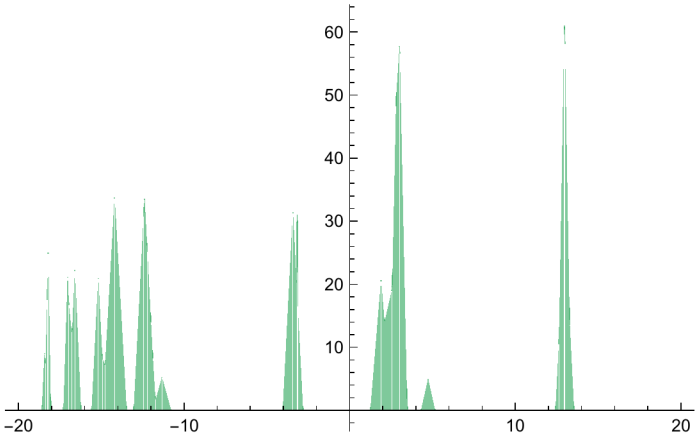


Rennt'n Elf innen Wald...

Definiere Waldpotential:

In[71]:=
```
PosList = RandomReal[{-1, 1}, 500];
HeightList = RandomReal[1, 500];
WidthList = RandomReal[1, 500];
UWald[x_, s_, n_, h_, w_] := Sum[USingleTree[x,
    s * PosList[[i]] , h * HeightList[[i]], w * WidthList[[i]]], {i, 1, n}];
WaldPot = UWald[x, 40, 2 * size / 2, 40, 1.5];
Plot[WaldPot,
 {x, -20, 20},
 PlotRange → Full,
 Filling → Axis,
 FillingStyle → Directive[RGBColor[0., 0.58, 0.22], Opacity[.5]],
 PlotStyle → None]
```

Out[76]=



Schrödingergleichung:

In[77]:=
```
schrWaldElf = -1 / (2 elfMass) D[ψ[x, t], x, x] + WaldPot * ψ[x, t] == i D[ψ[x, t], t];
```

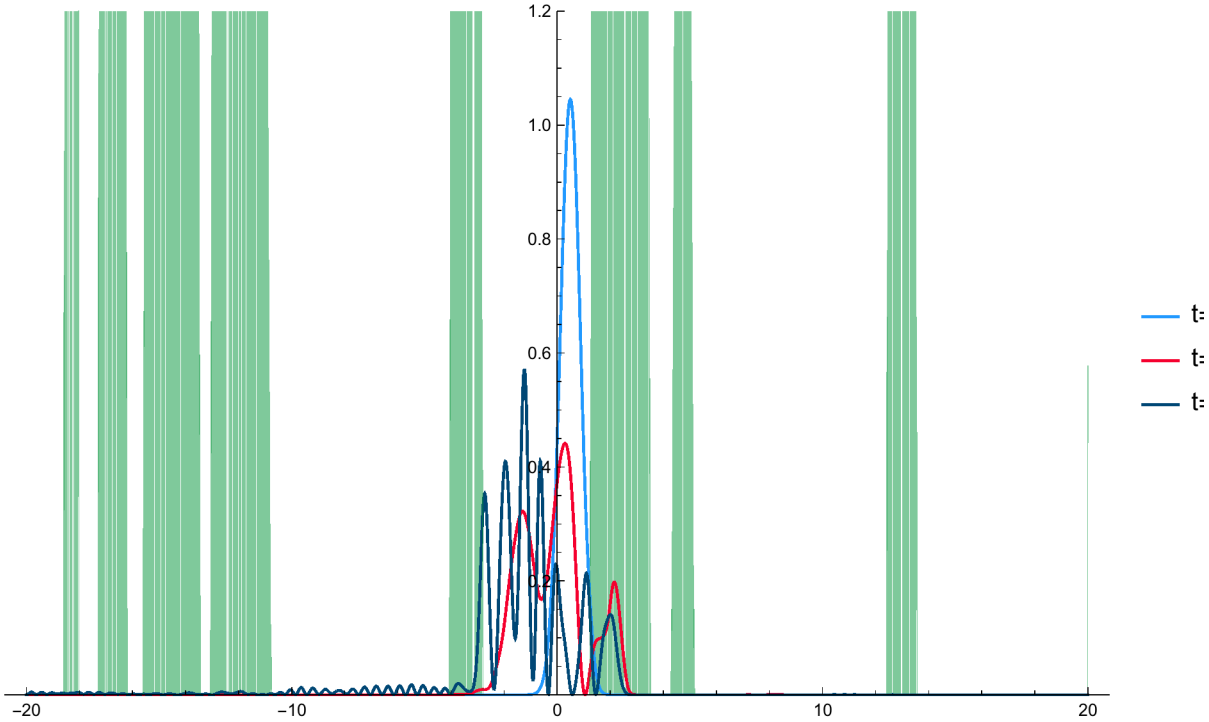
Lösung:

In[78]:=
```
waldElf = schrodingerSolve[schrWaldElf, 0, initwidth, santaP, 20, maxtimeElf];
```

••• NDSolve: Warning: scaled local spatial error estimate of 14.643765750425597` at t = 4.` in the direction of independent variable x is much greater than the prescribed error tolerance. Grid spacing with 649 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.

In[167]:=
```
Plot[
 {WaldPot,
  Abs[ψ[x, .1]]^2 /. waldElf,
  Abs[ψ[x, 1]]^2 /. waldElf,
  Abs[ψ[x, 4]]^2 /. waldElf},
 {x, -20, 20},
 PlotRange → {0, 1.2},
 Filling → {1 → Axis},
 FillingStyle → Directive[RGBColor[0., 0.58, 0.22], Opacity[.5]],
 PlotStyle → {None, RGBColor[0.13, 0.6, 1.], RGBColor[0.96, 0., 0.18],
   RGBColor[0., 0.28, 0.46](*,RGBColor[1.,0.55,0.19]*)},
 PlotLegends → {None, "t=0.1", "t=1", "t=4"},
 ImageSize → Large]
```

Out[167]=



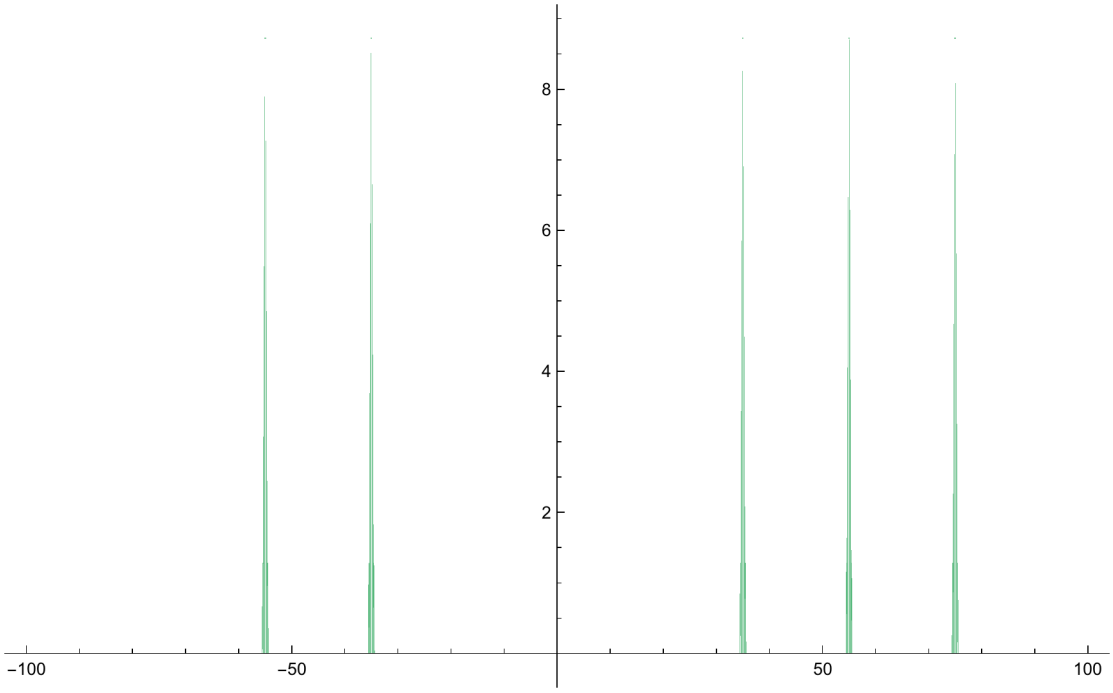
Vergleicht man die Wahrscheintlichkeitsverteilung des Elfen zu gleichen Zeitpunkten im regelmäßigen Baumschulenpotential mit der zufälligen Baumverteilung im Wald erkennt man, dass der Elf im Wald auch nach längerer Simulation eher um die Ausgangsposition verteilt ist.

Eigensystem einer Christbaumkugel in der Baumschule

Definiere neues Baumschulpotential

In[163]:=
```
GroßeBaumschulePot = UBaumschule[x, 100, 20, 10, 1];
GroßeBaumschulePot2 = UBaumschule[x, 100, 20, 10, 1];
Plot[GroßeBaumschulePot,
 {x, -100, 100},
 PlotRange → Full,
 Filling → Axis,
 FillingStyle → Directive[RGBColor[0., 0.58, 0.22], Opacity[.5]],
 PlotStyle → None,
 ImageSize → Large]
```

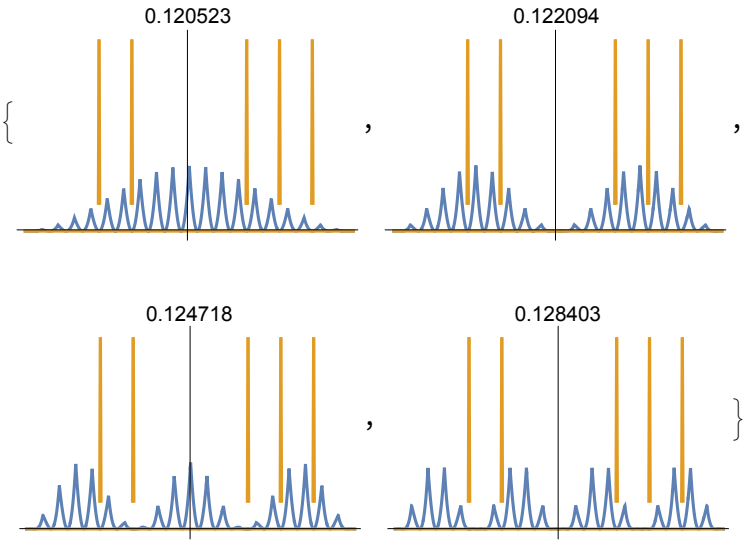
Out[165]=



In[146]:=
```
k = 4;
{vals, funs} = NDEigensystem[
    {-D[ϕ[x], x, x] + GroßeBaumschulePot * ϕ[x],
     DirichletCondition[ϕ[x] == 0, True]},
    ϕ,
    {x, -100, 100},
    k];
```

In[156]:=
```
Table[
 Plot[
  {(Evaluate[ funs[[l]] ][x] )^2,
   .01 GroßeBaumschulePot},
  {x, -100, 100},
  PlotRange → Full, PlotLabel → vals[[l]], PlotTheme → "Minimal"],
 {l, Length[vals]}]
```

Out[156]=



Leider habe ich es nicht mehr geschafft mehrere Potentialparameter zu testen. Die Eigenfunktionen der zeitunabhängigen Schrödingergleichung zeigen sinusförmige Hüllkurven mit größerer Frequenz bei höheren Energiewerten. Ich habe es auch nicht mehr geschafft, die Darstellungsprobleme des Potentials zu lösen.