Quantum Walks

Author: Enrique Arroyo Moro.

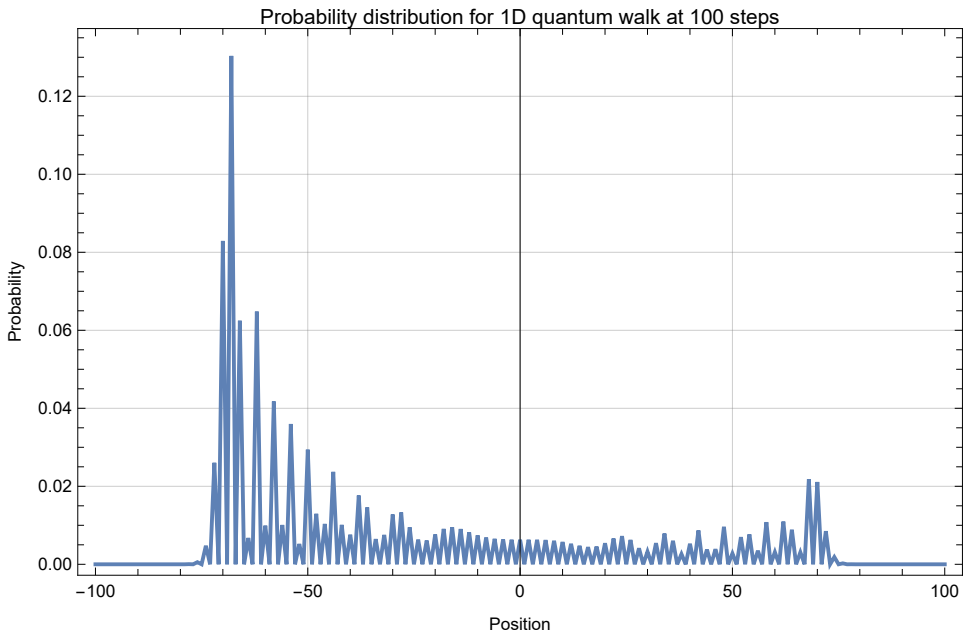
Simulation 1: Probability distribution for 1D quantum walk at 100 steps

We plot the position probability distribution of a 1D discrete-time quantum walk [1]. The walker has spin $s = 1/2$, so the coin space is two-dimensional, with basis states $|0\rangle$ (spin up) and $|1\rangle$ (spin down). The walk takes place on a 1D lattice with $n = 201$ sites, and we evolve the system for 100 steps. The initial state is chosen as $|0\rangle$ in the coin (spin) space and $|0\rangle$ in position space, i.e., the walker is fully localized at position $x = 0$. Both the number of lattice sites n and the number of steps can be easily modified at the beginning of the code.

We use the Hadamard operator as the coin, yielding the standard Hadamard walk. The conditional shift is defined as follows: if the spin is $|0\rangle$, the walker moves one site to the left; if the spin is $|1\rangle$, it moves one site to the right. Instead of explicitly constructing the full shift operator (which would be a $2n \times 2n$ matrix), we implement the shift by directly re-indexing (rotating) the components of the wavefunction. This produces the same result while being computationally more efficient.

In[1]:=
```
ClearAll["Global`*"]

s = 1 / 2; (*spin of the walker*)
n = 201; (*number of allowed positions for the walked*)
steps = 100; (*maximum steps the walker takes*)
spinini = SparseArray[{1} → 1, {2 s + 1}]; (*initial conditions for the spin,
note that the element 1 refers to |0> (spin up)*)
positionini = SparseArray[{101} → 1, {n}]; (*initial conditions for the position*)

wavefunction = Flatten[KroneckerProduct[spinini, positionini]];
(*the initial conditions define the initial wavefunction*)

H = KroneckerProduct[1 / √2 {{1, 1}, {1, -1}}, IdentityMatrix[n]];
(*our coin operator is the Hadamard transform*)
shift[wavefunction_] := Join[RotateLeft[wavefunction[[1 ;; n]]],
   RotateRight[wavefunction[[n + 1 ;; 2 n]]]]; (*shift operation encoded*)
coinandshift[wavefunction_] := shift[H.wavefunction];
(*a combination of applying the coin operator and then the shift*)

wavefunction = Nest[coinandshift, wavefunction, steps]; (*coin and shift
 are applied a number of times to the wavefunction defined by the steps*)

probability = Abs[wavefunction[[1 ;; n]]]^2 + Abs[wavefunction[[n + 1 ;; 2 n]]]^2;
(*the probability is computed as the sum of the components squared*)

ListPlot[probability, DataRange → {-100, 100},
 PlotRange → Full, Joined → True, GridLines → Automatic,
 PlotLabel → "Probability distribution for 1D quantum walk at 100 steps",
 Frame → True, FrameLabel → {"Position", "Probability"}, ImageSize → 500]
```

Out[13]=

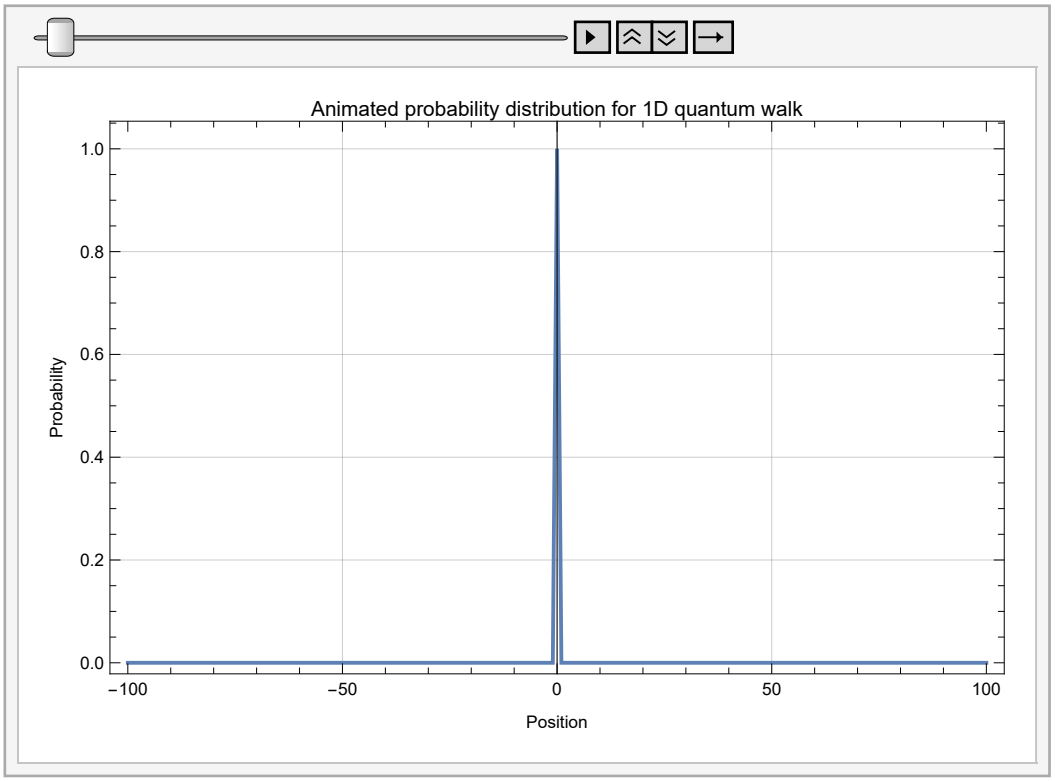


Simulation 2: Animated probability distribution for 1D quantum walk

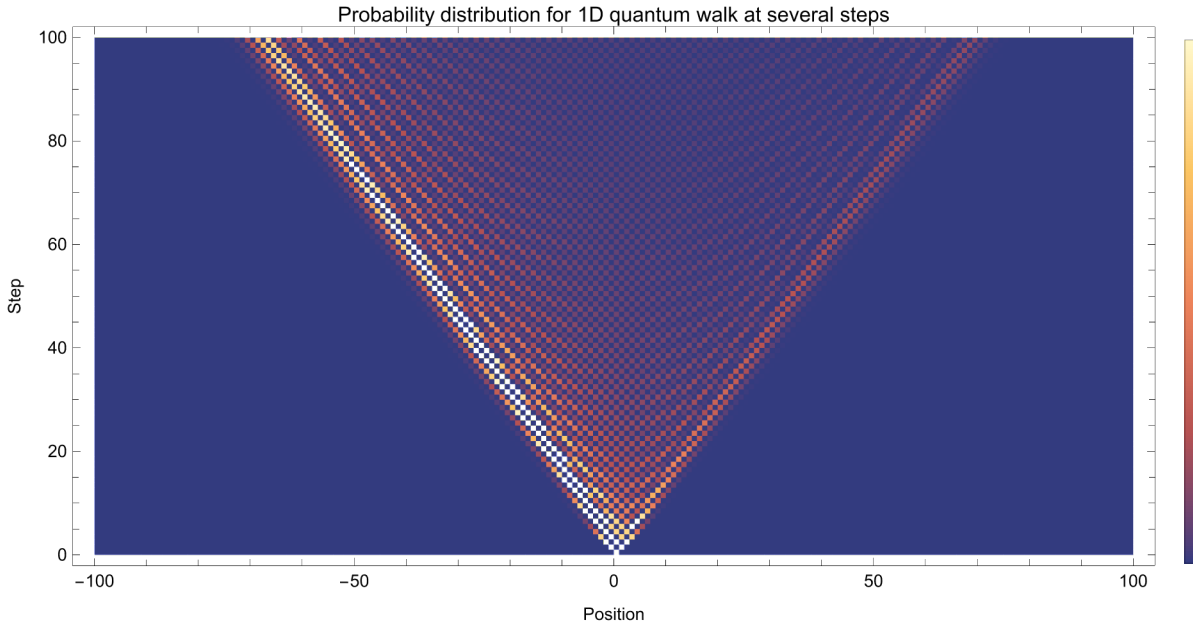
With a small modification to the code, we animate the position probability distribution over several time steps [1]. We consider the same spin-1/2 walker (two spin states) on a 1D lattice with $n = 201$ sites, using the same coin, shift rule, and initial state as in Simulation 1.

In[14]:=
```
ClearAll["Global`*"]

s = 1 / 2;
n = 201;
steps = 100;
spinini = SparseArray[{1} → 1, {2 s + 1}];
positionini = SparseArray[{101} → 1, {n}];

wavefunction = Flatten[KroneckerProduct[spinini, positionini]];

H = KroneckerProduct[1 / √2 {{1, 1}, {1, -1}}, IdentityMatrix[n]];
shift[wavefunction_] :=
  Join[RotateLeft[wavefunction[[1 ;; n]]], RotateRight[wavefunction[[n + 1 ;; 2 n]] ]];
coinandshift[wavefunction_] := shift[H.wavefunction];

wavefunctions = Transpose[NestList[coinandshift, wavefunction, steps]];
(*the key is using NestList[] instead of Nest[] to store intermediate results*)
probabilities = Abs[wavefunctions[[1 ;; n]]^2 + Abs[wavefunctions[[n + 1 ;; 2 n]]^2;
(*we store the probabilities for every step taken*)

frames = Table[ListPlot[probabilities[[All, 1 + i]],
    DataRange → {-100, 100}, PlotRange → Full, Joined → True, GridLines → Automatic,
    PlotLabel → "Animated probability distribution for 1D quantum walk", Frame → True,
    FrameLabel → {"Position", "Probability"}, ImageSize → 500], {i, 0, steps, 1}];

ListAnimate[frames, AnimationRunning → False]

ListDensityPlot[Transpose[probabilities], DataRange → {{-100, 100}, {0, steps}},
 PlotRange → {0, 0.15}, AspectRatio → steps / n, InterpolationOrder → 0,
 PlotLabel → "Probability distribution for 1D quantum walk at several steps",
 PlotLegends → Automatic, Frame → True,
 FrameLabel → {"Position", "Step"}, ImageSize → 600]
```

Out[27]=



Out[28]=



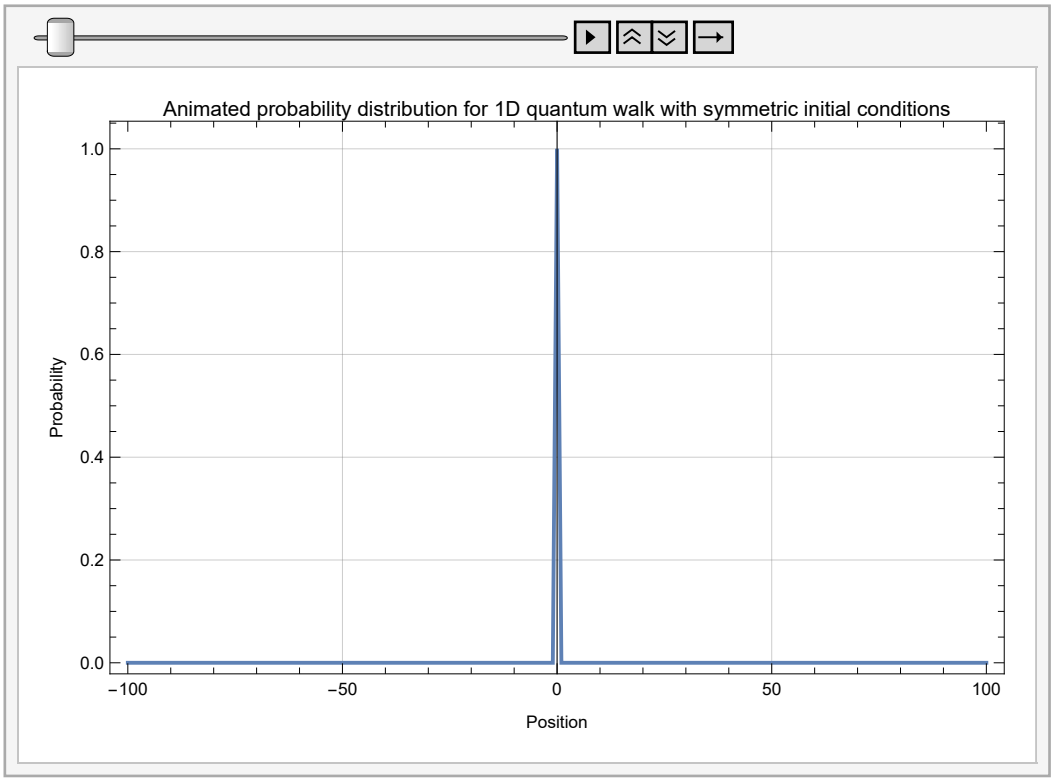
Simulation 3: Animated probability distribution for 1D quantum walk with symmetric initial conditions

We modify the initial coin (spin) state to show how it affects the probability distribution [1].

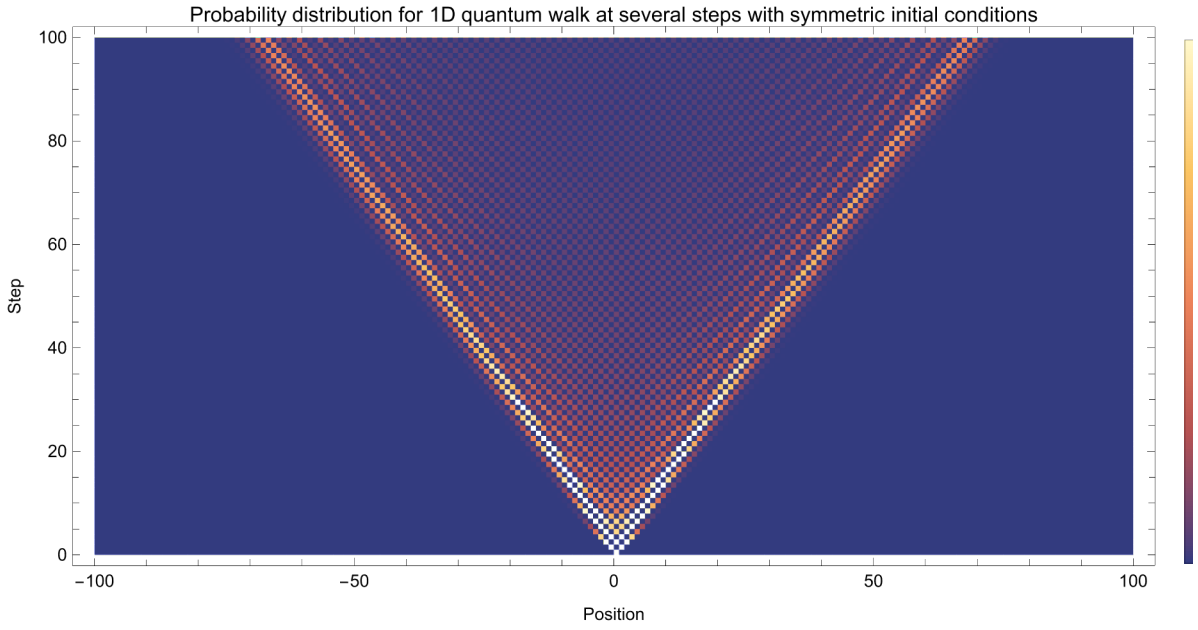
Starting from $|0\rangle$ biases the walk to the left because the conditional shift sends spin-up left and spin-down right, even if after one Hadamard step the spin measurement probabilities are 1/2 and 1/2. A symmetric distribution is obtained by choosing the initial coin state 1/Sqrt[2] ($|0\rangle - i\,|1\rangle$), i.e., by setting an appropriate relative phase between $|0\rangle$ and $|1\rangle$.

In[29]:=
```
ClearAll["Global`*"]

s = 1 / 2;
n = 201;
steps = 100;
spinini = SparseArray[{{1} → 1 / Sqrt[2], {2} → I / Sqrt[2]}, {2 s + 1}];
(*symmetric initial conditions*)
positionini = SparseArray[{101} → 1, {n}];

wavefunction = Flatten[KroneckerProduct[spinini, positionini]];

H = KroneckerProduct[1 / Sqrt[2] {{1, 1}, {1, -1}}, IdentityMatrix[n]];
shift[wavefunction_] :=
  Join[RotateLeft[wavefunction[[1 ;; n]]], RotateRight[wavefunction[[n + 1 ;; 2 n]]]];
coinandshift[wavefunction_] := shift[H.wavefunction];

wavefunctions = Transpose[NestList[coinandshift, wavefunction, steps]];
probabilities = Abs[wavefunctions[[1 ;; n]]^2 + Abs[wavefunctions[[n + 1 ;; 2 n]]]^2;

frames =
  Table[ListPlot[probabilities[[All, 1 + i]], DataRange → {-100, 100}, PlotRange → Full,
    Joined → True, GridLines → Automatic, PlotLabel → "Animated probability
       distribution for 1D quantum walk with symmetric initial conditions",
    Frame → True, FrameLabel → {"Position", "Probability"},
    ImageSize → 500], {i, 0, steps, 1}];

ListAnimate[frames, AnimationRunning → False]

ListDensityPlot[Transpose[probabilities], DataRange → {{-100, 100}, {0, steps}},
 PlotRange → {0, 0.15}, AspectRatio → steps / n, InterpolationOrder → 0,
 PlotLabel → "Probability distribution for 1D quantum walk at several
    steps with symmetric initial conditions", PlotLegends → Automatic,
 Frame → True, FrameLabel → {"Position", "Step"}, ImageSize → 600]
```

Out[42]=



Out[43]=

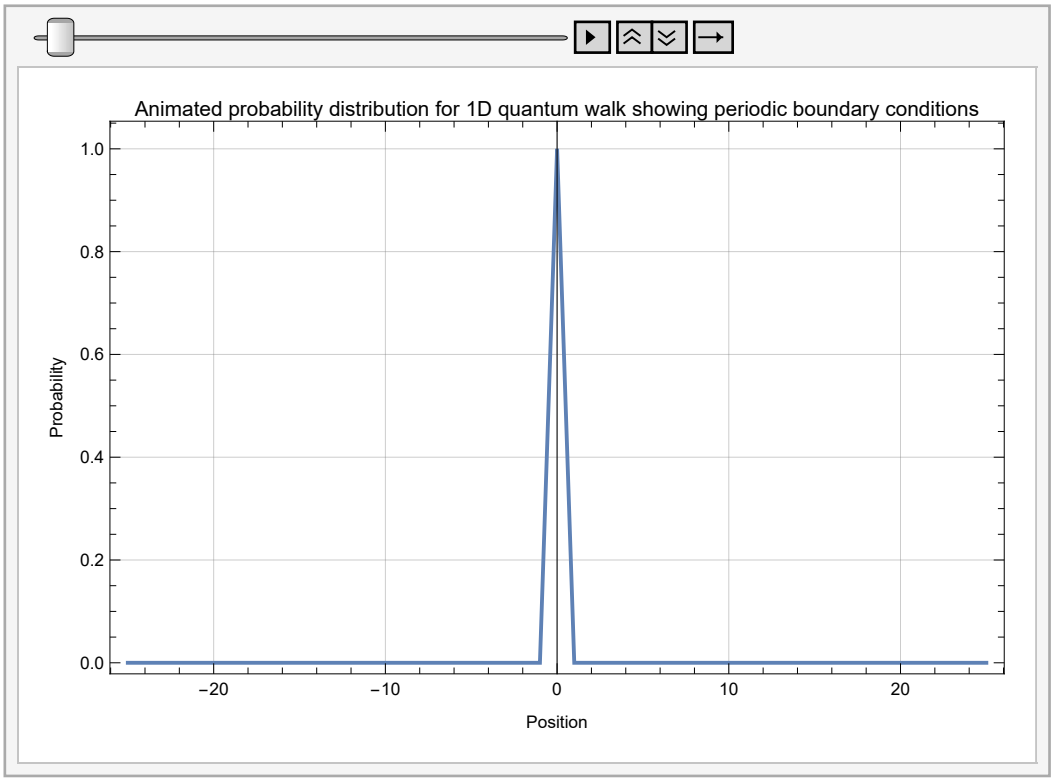


Simulation 4: Periodic boundary conditions

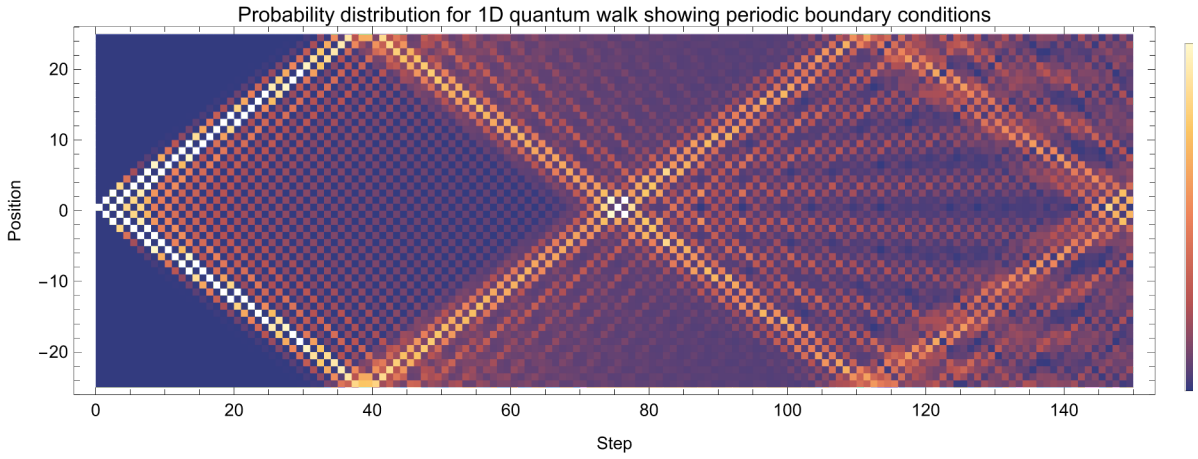
Keeping the symmetric initial coin state, we verify that periodic (wrap-around) boundary conditions are correctly implemented [1]. With periodic boundaries, probability that reaches one end of the lattice reappears at the opposite end after the shift operation. To make this behavior easy to see, we reduce the lattice to $n = 51$ sites and plot the distribution near the boundaries.

In[44]:=
```
ClearAll["Global`*"]

s = 1 / 2;
n = 51;
steps = 150;
spinini = SparseArray[{{1} → 1 / √2 , {2} → I / √2 }, {2 s + 1}];
(*symmetric initial conditions*)
positionini = SparseArray[{26} → 1, {n}];

wavefunction = Flatten[KroneckerProduct[spinini, positionini]];

H = KroneckerProduct[1 / √2 {{1, 1}, {1, -1}}, IdentityMatrix[n]];
shift[wavefunction_] :=
  Join[RotateLeft[wavefunction[[1 ;; n]]], RotateRight[wavefunction[[n + 1 ;; 2 n]] ]];
coinandshift[wavefunction_] := shift[H.wavefunction];

wavefunctions = Transpose[NestList[coinandshift, wavefunction, steps]];
probabilities = Abs[wavefunctions[[1 ;; n]]^2 + Abs[wavefunctions[[n + 1 ;; 2 n]]^2;

frames = Table[ListPlot[probabilities[[All, 1 + i]],
    DataRange → {-25, 25}, PlotRange → Full, Joined → True, GridLines → Automatic,
    PlotLabel → "Animated probability distribution for 1D quantum
       walk showing periodic boundary conditions", Frame → True,
    FrameLabel → {"Position", "Probability"}, ImageSize → 500], {i, 0, steps, 1}];

ListAnimate[frames, AnimationRunning → False]

ListDensityPlot[probabilities, DataRange → {{0, steps}, {-25, 25}},
 PlotRange → {0, 0.15}, AspectRatio → n / steps, InterpolationOrder → 0,
 PlotLabel → "Probability distribution for 1D quantum walk
    showing periodic boundary conditions", PlotLegends → Automatic,
 Frame → True, FrameLabel → {"Step", "Position"}, ImageSize → 600]
```

Out[57]=



Out[58]=



In[59]:=
```
Export["C:\\Users\\enriq\\OneDrive\\MSc\\1. Programming in Mathematica
    and Python (PMP)\\FinalProject\\GitHub sharing\\.png", %619, "PNG"]
```

Out[59]=
```
C:\Users\enriq\OneDrive\MSc\1. Programming in
  Mathematica and Python (PMP)\FinalProject\GitHub sharing\.png
```

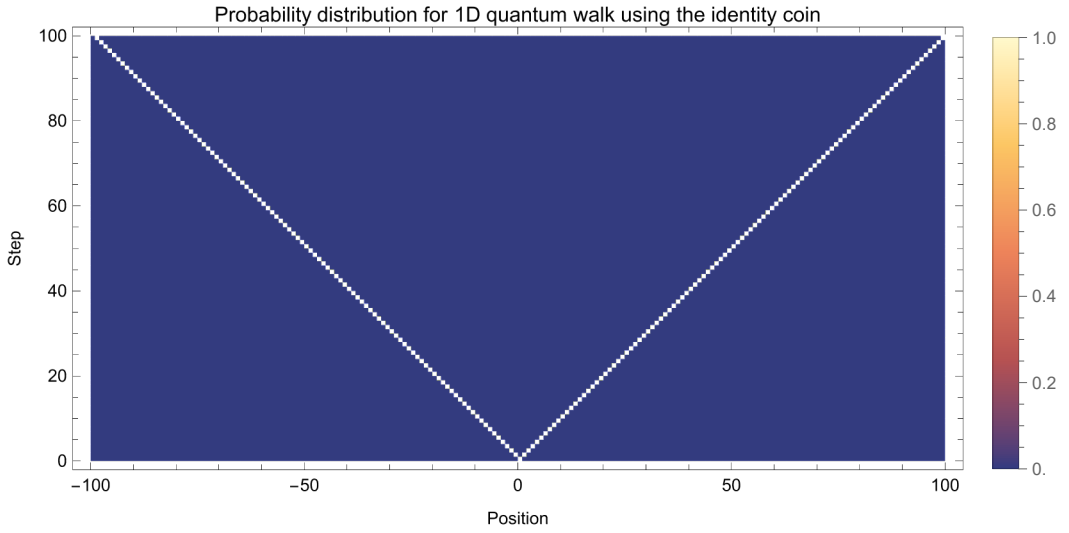
Simulation 5: Different coins

So far, we have studied the walk using the Hadamard coin. Here we illustrate how the choice of coin [1] affects the dynamics by comparing two additional coins: the identity coin (no mixing, yielding a trivial quantum walk) and the Pauli-X coin (bit-flip). We use a lattice with n = 201 sites, run 100 steps, and use the symmetric initial coin state, then plot the resulting probability

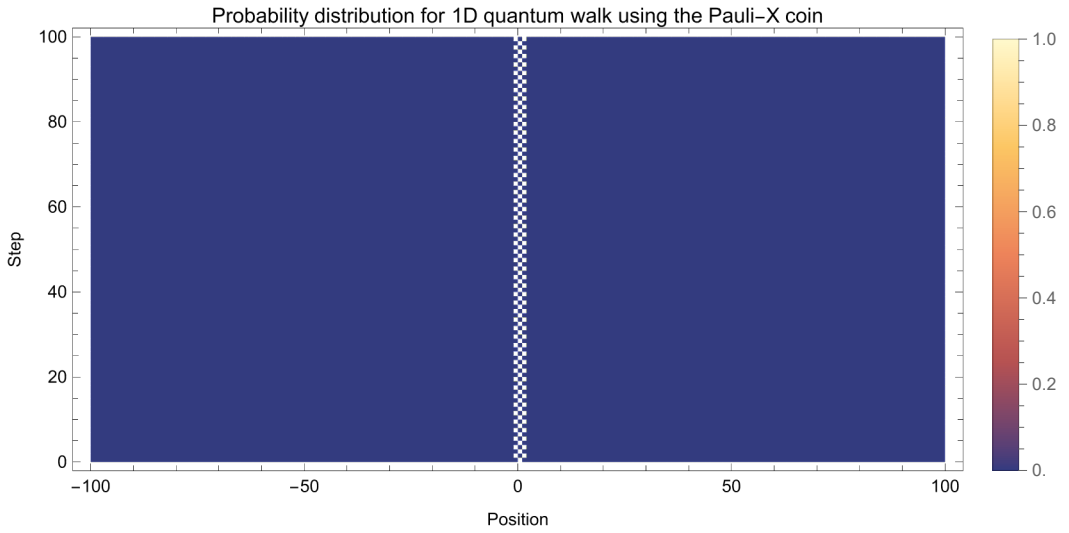
distributions.

In[60]:=
```
ClearAll["Global`*"]

s = 1 / 2;
n = 201;
steps = 100;
spinini = SparseArray[{{1} → 1 / Sqrt[2], {2} → I / Sqrt[2]}, {2 s + 1}];
(*symmetric initial conditions*)
positionini = SparseArray[{101} → 1, {n}];

wavefunction = Flatten[KroneckerProduct[spinini, positionini]];

Id = IdentityMatrix[2 * n];(*Identity coin*)
X = KroneckerProduct[{{0, 1}, {1, 0}}, IdentityMatrix[n]];(*Pauli-X coin*)
shift[wavefunction_] :=
  Join[RotateLeft[wavefunction[[1 ;; n]]], RotateRight[wavefunction[[n + 1 ;; 2 n]]]];
coinandshiftId[wavefunction_] := shift[Id.wavefunction];
coinandshiftX[wavefunction_] := shift[X.wavefunction];

wavefunctionsId = Transpose[NestList[coinandshiftId, wavefunction, steps]];
probabilitiesId = Abs[wavefunctionsId[[1 ;; n]]]^2 + Abs[wavefunctionsId[[n + 1 ;; 2 n]]]^2;
wavefunctionsX = Transpose[NestList[coinandshiftX, wavefunction, steps]];
probabilitiesX = Abs[wavefunctionsX[[1 ;; n]]]^2 + Abs[wavefunctionsX[[n + 1 ;; 2 n]]]^2;

ListDensityPlot[Transpose[probabilitiesId], DataRange → {{-100, 100}, {0, steps}},
 PlotRange → {0, 0.15}, AspectRatio → steps / n, InterpolationOrder → 0,
 PlotLabel → "Probability distribution for 1D quantum walk using the identity coin",
 PlotLegends → Automatic, Frame → True,
 FrameLabel → {"Position", "Step"}, ImageSize → 500]
ListDensityPlot[Transpose[probabilitiesX], DataRange → {{-100, 100}, {0, steps}},
 PlotRange → {0, 0.15}, AspectRatio → steps / n, InterpolationOrder → 0,
 PlotLabel → "Probability distribution for 1D quantum walk using the Pauli-X coin",
 PlotLegends → Automatic, Frame → True,
 FrameLabel → {"Position", "Step"}, ImageSize → 500]
```

Out[76]=



Out[77]=

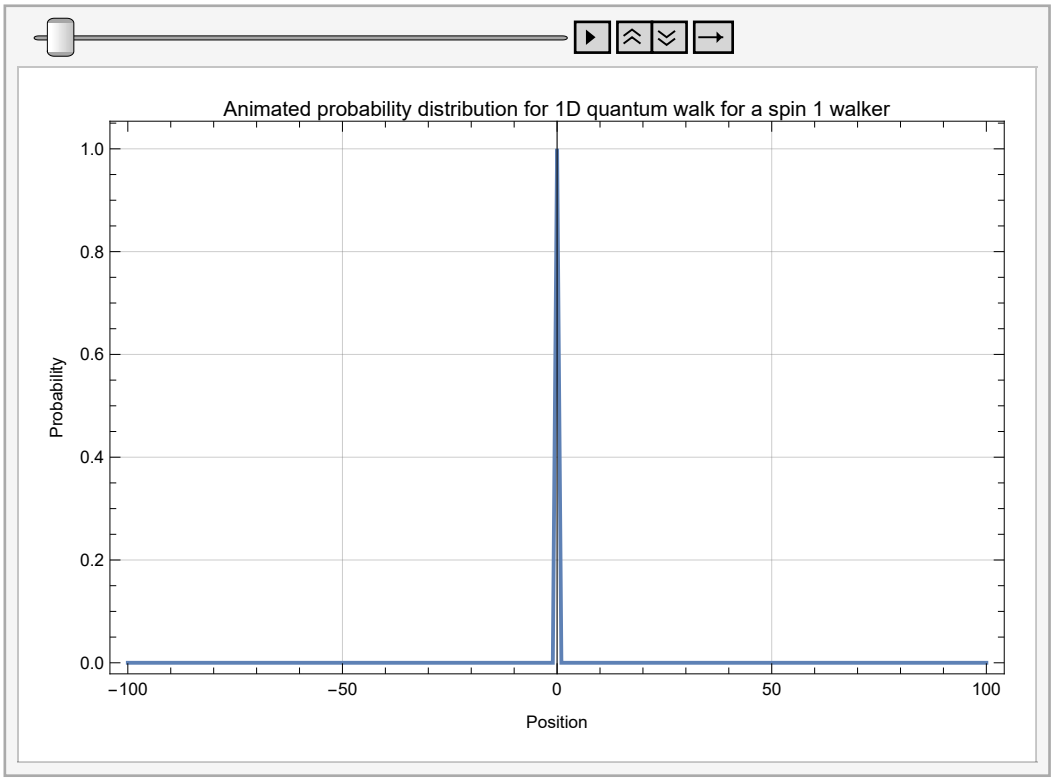


Simulation 6: 1D quantum walk for a spin 1 walker

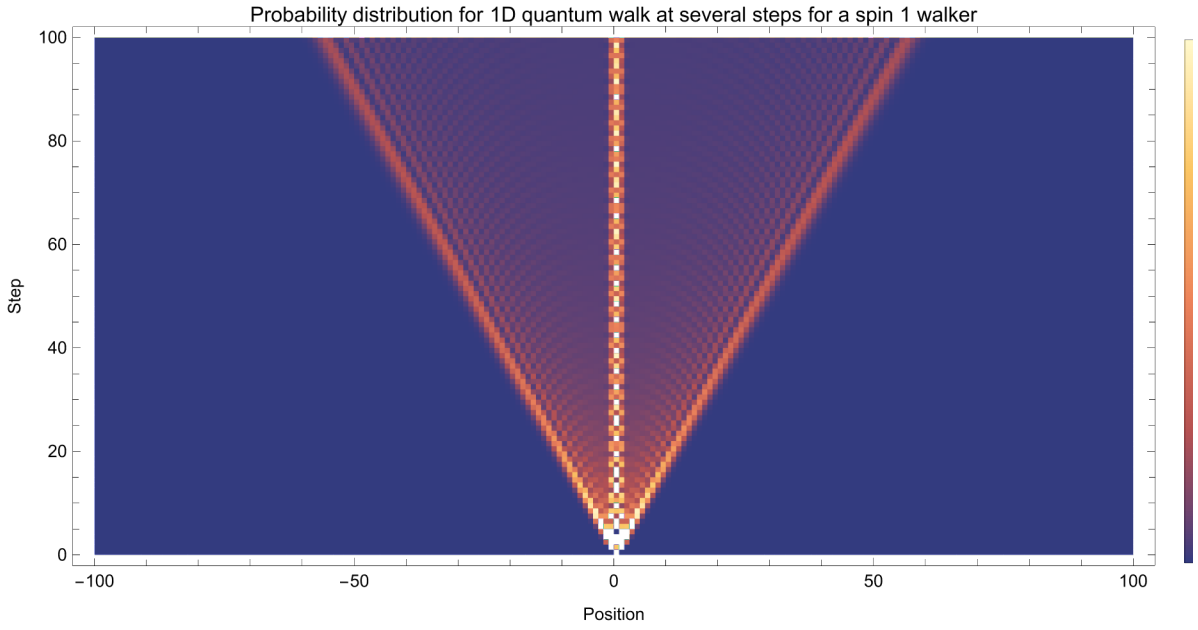
We simulate a spin-1 (three-level) quantum walk. In this case, since the spin is a superposition of three states, we need a coin defined by a 3 x3 unitary matrix. One commonly used is the Grover coin, which is defined for any dimension [2, 3]. We also need to redefine the shifting operation: now, when the spin of the walker is $|-1\rangle$ or $|1\rangle$, the walker will turn left or right, respectively; in the other case, if the spin is $|0\rangle$, it will remain in its position. We simulate a lattice with $n = 201$ points for 100 steps.

In[78]:=
```
ClearAll["Global`*"]

s = 1; (*now, the spin is 1*)
n = 201;
steps = 100;
spinini = SparseArray[{2} → 1, {2 s + 1}]; (*initially, the particle has spin |0>*)
positionini = SparseArray[{101} → 1, {n}];

wavefunction = Flatten[KroneckerProduct[spinini, positionini]];

G3aux = 1 / 3 {{-1, 2, 2}, {2, -1, 2}, {2, 2, -1}}; (*Grover coin in three dimensions*)
G3 = KroneckerProduct[G3aux , IdentityMatrix[n]];
shift[wavefunction_] :=
  Join[RotateLeft[wavefunction[[1 ;; n]]], wavefunction[[n + 1 ;; 2 n]],
   RotateRight[wavefunction[[2 n + 1 ;; 3 n]]]]; (*new shift operation*)
coinandshift[wavefunction_] := shift[G3.wavefunction];

wavefunctions = Transpose[NestList[coinandshift, wavefunction, steps]];
probabilities = Abs[wavefunctions[[1 ;; n]]^2 +
   Abs[wavefunctions[[n + 1 ;; 2 n]]]^2 + Abs[wavefunctions[[2 n + 1 ;; 3 n]]]^2;(*tweak*)

frames = Table[ListPlot[probabilities[[All, 1 + i]], DataRange → {-100, 100},
    PlotRange → Full, Joined → True, GridLines → Automatic, PlotLabel →
     "Animated probability distribution for 1D quantum walk for a spin 1 walker",
    Frame → True, FrameLabel → {"Position", "Probability"},
    ImageSize → 500], {i, 0, steps, 1}];

ListAnimate[frames, AnimationRunning → False]

ListDensityPlot[Transpose[probabilities], DataRange → {{-100, 100}, {0, 100}},
 PlotRange → {0, 0.15}, AspectRatio → steps / n, InterpolationOrder → 0,
 PlotLabel → "Probability distribution for 1D quantum walk
    at several steps for a spin 1 walker", PlotLegends → Automatic,
 Frame → True, FrameLabel → {"Position", "Step"}, ImageSize → 600]
```

Out[92]=



Out[93]=



References:
[1] J. Kempe, “Quantum random walks – an introductory overview,” Contemporary Physics 44(4), 307–327 (2003). arXiv:quant-ph/0303081. DOI: 10.1080/00107151031000110776. Available at: https://arxiv.org/abs/quant-ph/0303081.
[2] L. K. Grover, “A fast quantum mechanical algorithm for database search,” in Proceedings of the 28th Annual ACM Symposium on Theory of Computing (STOC ’96), pp. 212–219 (1996). DOI: 10.1145/237814.237866. Available at: https://arxiv.org/abs/quant-ph/9605043.
[3] A. Saha, D. Saha, and A. Chakrabarti, “Discrete-time Quantum Walks in Qudit Systems,”

arXiv:2207.04319 (2022). DOI: 10.48550/arXiv.2207.04319. Available at: https://arxiv.org/abs/2207.04319.