OLLSCOIL NA HÉIREANN, GAILLIMH

Fairness in Qutrit Walks

David F. Dolphin
May 2011

Thesis submitted in partial fulfilment of the requirements for a Master's of Science.

In classical random walks moving from a two state walk to a three state (lazy) walk only affects the scale of the probability distribution. In quantum walks the introduction of a third (lazy) state results in a fundamentally different probability distribution. We investigate the properties of Qutrit (three state quantum) walks and compare them to Qubit (two state quantum) walks. We also provide a computation tool for calculating n-dimensional quantum walks with arbitrary coins on a cycle or line.

Table of Contents

Table of Figures

1 Introduction

Random walks are a statistical tool, used to study patterns in randomness. They can be applied over a finite space (typically a graph) or an infinite continuum.

Quantum (random) walks are the quantum equivalent of classical random walks. They are studied to observe the statistical properties of quantum systems. These results aid in the general design of randomised quantum algorithms, particularly regarding efficiency concerns for those algorithms (Kempe, 2003).

Much of the work concerning discrete quantum walks deals with two-state bits, known as qubits. At each time step in a qubit system the particle must move. Our work looks at three-state systems, whose particles are known as qutrits. In a qutrit system the particle is not forced to move at each time step; there is a possibility that it can remain in the same location.

A work in progress of this work was presented at the 14th Workshop on Quantum Information Processing (Dolphin & McGettrick, 2011).

2 Theory

To observe a random walk we repeated perform some statistically random event. The outcome of this event moves a particle (in our case) on a line or cycle. We repeat this event x times to obtain a single walk.

To observe patterns in the walk we repeat it y times. The final position after x steps for each of the y walks can be plotted on a histogram. As the number of walks (y) approaches infinity the histogram of positions will tend toward the probability distribution of the random event being observed.

2.1 Classical Walks

The most approachable application of a discrete classical random walk is a fair coin toss, the result of which moves a particle left or right on an infinite line. After this experiment has been run a number of times, the distance from the origin is recorded. This series of experiments is then run a number of times, and the distance from the origin is recorded each time. When a histogram of these results is plotted, we see that the distribution of distances from the origin is approximately normal (Figure 2-1). As the particle can only land on an odd numbered space after an odd number of steps and an even numbered space after an even number of steps, over a larger number of steps every second value will be zero.

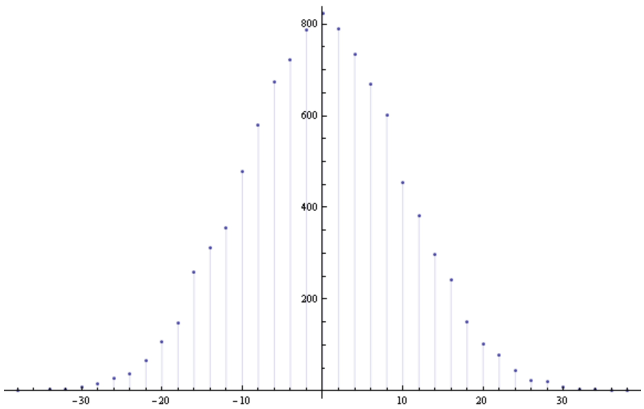


Figure 2-1: Histogram of results from 10,000 repetitions of a 100 step fair calssical two state random walk

This can be generalised to show the probability of the particle being at a certain position after a certain number of steps (Kempe, 2003).

T/i	-5	-4	-3	-2	-1	0	1	2	3	4	5
0						1					
1					$1/2$		$1/2$				
2				$1/4$		$1/2$		$1/4$			
3			$1/8$		$3/8$		$3/8$		$1/8$		
4		$1/16$		$1/4$		$3/8$		$1/4$		$1/16$	
5	$1/32$		$5/32$		$5/16$		$5/16$		$5/32$		$1/32$

Figure 2-2: The probability of being at node *i* after *T* steps of the classical random walk on the line starting at 0.

However, on a closed graph the probabilities converge over time. On a cyclic graph with 4 nodes, where we can move clockwise or counter-clockwise after each time-step, the probabilities converge to $1/4$ for each of the four nodes.

We now consider a three state system on the integers. A particle can move left, right or remain stationary at each time step. Figure 2-3 shows the generalised probability distributions of a lazy classical walk.

$T/_i$	-5	-4	-3	-2	-1	0	1	2	3	4	5
0						1					
1					$^1/_3$	$^1/_3$	$^1/_3$				
2				$^1/_9$	$^2/_9$	$^1/_3$	$^2/_9$	$^1/_9$			
3			$^1/_{27}$	$^1/_9$	$^2/_9$	$^2/_{27}$	$^2/_9$	$^1/_9$	$^1/_{27}$		
4		$^1/_{81}$	$^4/_{81}$ 1/	$^{10}/_{81}$	$^{16}/_{81}$	$^{19}/_{81}$	$^{16}/_{81}$	$^{10}/_{81}$	$^4/_{81}$	$^1/_{81}$	
5	$^1/_{243}$	$^5/_{243}$	$^5/_{81}$	$^{10}/_{81}$	$^5/_{27}$	$^{17}/_{81}$	$^5/_{27}$	$^{10}/_{81}$	$^5/_{81}$	$^5/_{243}$	$^1/_{243}$

Figure 2-3: The probability of being at position *i* after *T* steps of the lazy classical random walk on the line starting at 0.

The lazy classical walk has a normal distribution; the introduction of the lazy step removes the odd/even restriction. Like the standard classical walk, the lazy classical walk converges to equal values on a closed graph.

It is possible to vary the lazy bias of a classical walk. A common bias is to choose a probability per iteration of $1/2$ for the lazy move, $1/4$ for the clockwise move, and $1/4$ for the counter-clockwise move. This bias can be modelled as two coin tosses, where one coin signals movement or its absence, the second signals the direction of movement (if the first coin signals movement).

However, we have chosen to give each movement possibility a probability of $1/4$. This is for comparison with the quantum randomization function we will be examining.

2.2 Quantum Walks

Quantum mechanics is different from most other branches of science in that it uses complex numbers in a fundamental way. Whenever we measure a quantum state we collapse it from a superposition of all states to a well defined state. In this process of measurement we lose the imaginary component of the states' probability (Yanofsky & Mannucci, 2008).

While other branches of science explain deterministic systems, where every unique input maps to a unique output, quantum systems are by their very nature probabilistic systems. As such, a study of the probabilities of a quantum system is vital in understanding how that system will perform. If we want to use a quantum system for computation we must we well acquainted with the probable outcomes of any given operation. With knowledge of the fundamental probabilities we can design our algorithms to take advantage of states with relatively certain outcomes.

Quantum algorithms depend on superposition. Measurement of a quantum computer in this superposition of states will collapse it to a single observable state. As such, operations are performed on a quantum computer without measurement. Only when all operations have been performed do we measure the system and observe the outcome.

Quantum physics requires that the evolution of quantum states is unitary and reversible (Aharonov, Ambainis, Kempe, & Vazirani, 2001). The randomising function we apply to our state must satisfy these constrains.

To obtain a fair (symmetric) probability distribution it is often necessary to use many coins (Brun, Carteret, & Ambainis, 2003) (Kempe, 2003). We represent multiple coins as an initial superposition of states, with different spins and sometimes complex amplitudes.

Quantum walks can be used to perform database search with quadratic speedup over the most efficient classical database search algorithm (Grover, 1996) (Shenvi, Kempe, & Whaley, 2003).

2.2.1 Hadamard Gate

The quantum randomizing function we are examining is known as a Hadamard Gate (also known as a Hadamard Coin). A two-state (qubit) Hadamard gate is given by H_2:

$$H_2 = \frac{1}{\sqrt{2}}\begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}$$

It is unitary and has been shown to be fair (Kempe, 2003). The Hadamard gate has been generalised to H_m, where m is the number of states in the system, the logical order (Marttala, 2007).

$$\sigma = e^{\frac{i2\pi}{m}}$$

$$H_m = \frac{1}{\sqrt{m}}\begin{pmatrix} 1 & 1 & 1 & \cdots & 1 & 1 \\ 1 & \sigma^{(m-1)} & \sigma^{2(m-1)} & \cdots & \sigma^2 & \sigma \\ 1 & \sigma^{(m-2)} & \sigma^{2(m-2)} & \cdots & \sigma^4 & \sigma^2 \\ 1 & \sigma^{(m-3)} & \sigma^{2(m-3)} & \cdots & \sigma^6 & \sigma^3 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 1 & \sigma & \sigma^2 & \cdots & \sigma^{(m-2)} & \sigma^{(m-1)} \end{pmatrix}$$

As we are interested in a lazy walk we want a three-state gate H_3, where the three states represent a left movement, a right movement, and no movement.

$$H_3 = \frac{1}{\sqrt{3}}\begin{pmatrix} 1 & 1 & 1 \\ 1 & -(-1)^{1/3} & (-1)^{2/3} \\ 1 & (-1)^{2/3} & -(-1)^{1/3} \end{pmatrix}$$

We are satisfied that H_3 is unitary as $H_3 {H_3}^{\dagger} = I_3$:

$$\frac{1}{3}\left(\begin{pmatrix} 1 & 1 & 1 \\ 1 & -(-1)^{1/3} & (-1)^{2/3} \\ 1 & (-1)^{2/3} & -(-1)^{1/3} \end{pmatrix} . \begin{pmatrix} 1 & 1 & 1 \\ 1 & (-1)^{2/3} & -(-1)^{1/3} \\ 1 & -(-1)^{1/3} & (-1)^{2/3} \end{pmatrix}\right) = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

2.2.2 Quantum Walks on a Graph

A Lazy One-Dimensional Discrete Quantum Walk takes place on the state space spanned by vectors

$$|n, p\rangle$$

where $n \in \mathbb{Z}$ and $p \in \{0,1,2\}$ is a three-state variable. n represents the position of a particle on the walk and is the walks classical component. p is the quantum component; for qubits this represents two spin states. In our work we look at qutrits[1] and as such p has three possible values which represent left movement, right movement and non-movement.

One time step of the qutrit walk is given by the translations:

$$|n, 0\rangle \longrightarrow a|n, 0\rangle + b|n + 1,1\rangle + c|n - 1,2\rangle$$

$$|n, 1\rangle \longrightarrow d|n, 0\rangle + e|n + 1,1\rangle + f|n - 1,2\rangle$$

$$|n, 2\rangle \longrightarrow g|n, 0\rangle + h|n + 1,1\rangle + i|n - 1,2\rangle$$

where

$$\begin{pmatrix} a & b & c \\ d & e & f \\ g & h & i \end{pmatrix} = H_3$$

This has been expanded from the two-state walk equations given elsewhere (Kempe, 2003) (McGettrick, 2010).

[1] Three-state quantum particles

2.3 Fairness

A fair classical coin is one which is described as producing a 50% chance of coming up heads after a single flip. After a large number of flips we will see that the number of times the coin comes up head is likely to be approximately equal to the number of times the coin comes up tails. It is highly unlikely that after a large number of coin flips there will all show heads. A symmetric probability distribution is produced, the highest point of which represents a trial where an equal number of head and tails results were observed.

While randomising a quantum particle the same results are not obtained. This can be thought of as firing a single photon through a prism which has a 50% chance of passing though and 50% chance of being refracted.

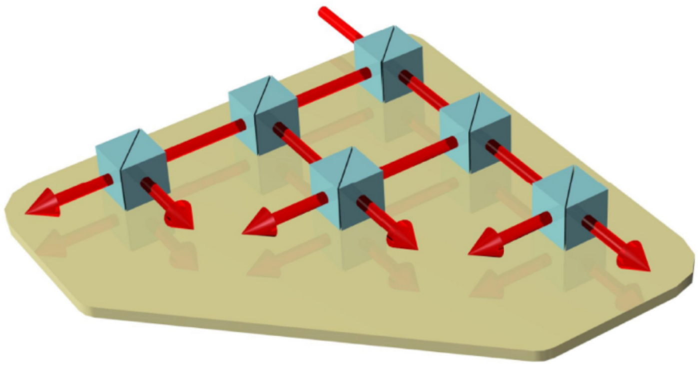


Figure 2-4: Laser bounding through prisms

While the particle will have a 50/50 chance of passing or reflecting after one step, after multiple steps the particle has a higher probability of continuing in the direction it was initially moving in. This asymmetry can be countered if we use two particles with opposite spin, one of which has imaginary amplitude. The difference between the asymmetric and symmetric results after 100 steps can be found in Figure 3-13.

For quantum walks we propose a comparitive measurement of fairness. For walk A to be considered more fair than walk B three considions must be satisfied:

1. Both walks must be of the same order (qubit, qutrit, &c.).
2. Both walks must produce symetric result graphs.
3. Walk A must have a lower maximum probability than Walk B after an equal number of steps.

3 Results

3.1 Walks on the Line

Qubit and qutrit walks were performed on the line. In each case, probabilities for the progression of the walk over its first one hundred (100) steps are shown. Probabilities are limited to a maximum of 0.2 in each graph for the sake of clarity.

For qubit (non lazy) walks odd numbered time steps only have non-zero probabilities on odd numbered positions, and even numbered time steps only have non-zero probabilities on even numbered positions. For the sake of clarity in the graphs below, qubit walks only show the even numbered positions on the even numbered time steps. A graph showing all steps in the balanced non lazy walk can be seen in the Appendix (Figure 5-1).

The following qubit starting states were used:

$$B_1 = |0,1\rangle$$
$$B_4 = \frac{1}{\sqrt{2}}(|0,0\rangle + |0,1\rangle)$$
$$B_5 = \frac{1}{\sqrt{2}}(|0,0\rangle + i|0,1\rangle)$$

The following qutrit starting states were used:

$$T_1 = |0,2\rangle$$
$$T_2 = |0,0\rangle$$
$$T_3 = i|0,0\rangle$$

$$T_4 = \frac{1}{\sqrt{2}}(|0,1\rangle + |0,2\rangle)$$

$$T_5 = \frac{1}{\sqrt{2}}(|0,1\rangle + i|0,2\rangle)$$

$$T_6 = \frac{1}{\sqrt{3}}(|0,0\rangle + |0,1\rangle + |0,2\rangle)$$

$$T_7 = \frac{1}{\sqrt{3}}(i|0,0\rangle + |0,1\rangle + |0,2\rangle)$$

The starting states represent the following:

1. Moving left
2. Non-moving
3. Imaginary non-moving
4. Equal combination of left and right moving
5. Equal combination of left and right moving, with imaginary right moving
6. Equal combination of left, right and non-moving
7. Equal combination of left, right and non-moving, with imaginary non-moving

It is noted that results from starting state T_2 and T_3 on a qutrit walk are equal. Also, states T_1 and T_5 yield unsymmetrical results for the qutrit walk, while states B_1 and B_4 yield unsymmetrical results for the qubit walk. All other starting states yield symmetric results.

3.1.1 Probability distribution evolution

Graphs in this section represent an evolution of the probability distribution over time.

All results have been generated using our Mathematica Module (included in the Appendix).

Initial states are listed on page 11.

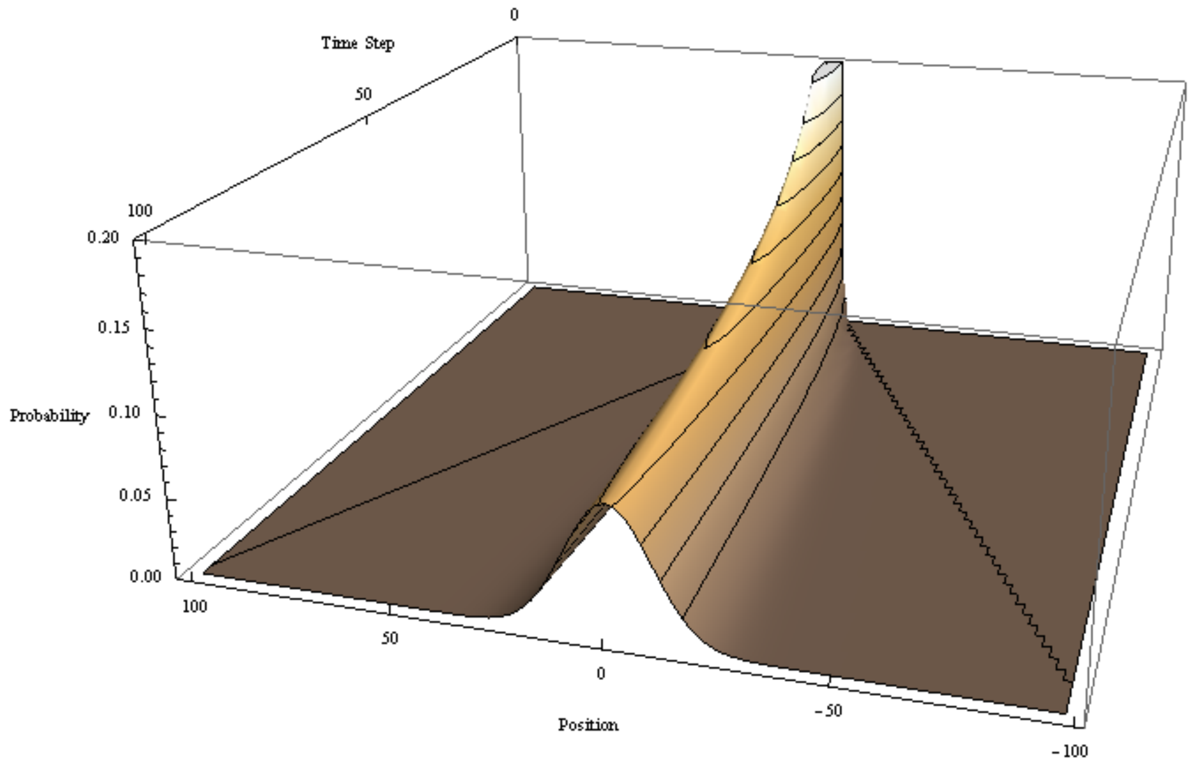


Figure 3-1: Non-lazy classical walk

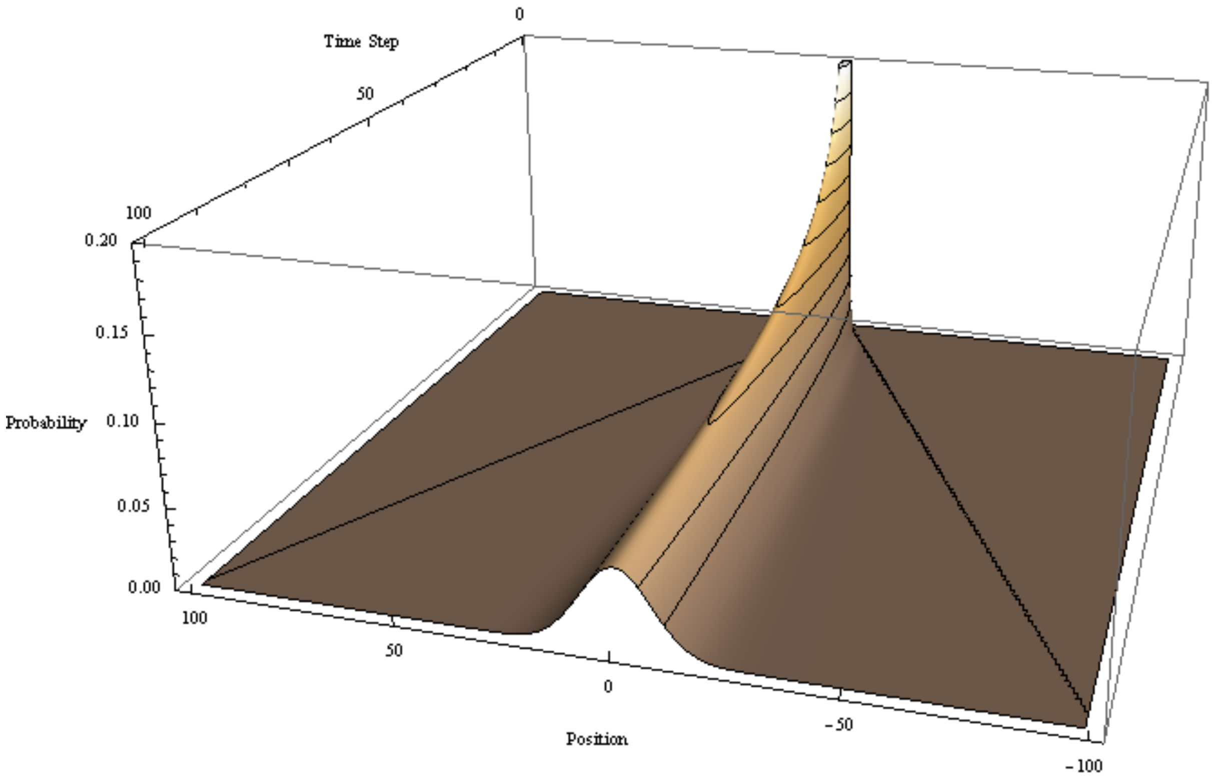


Figure 3-2: Lazy classical walk

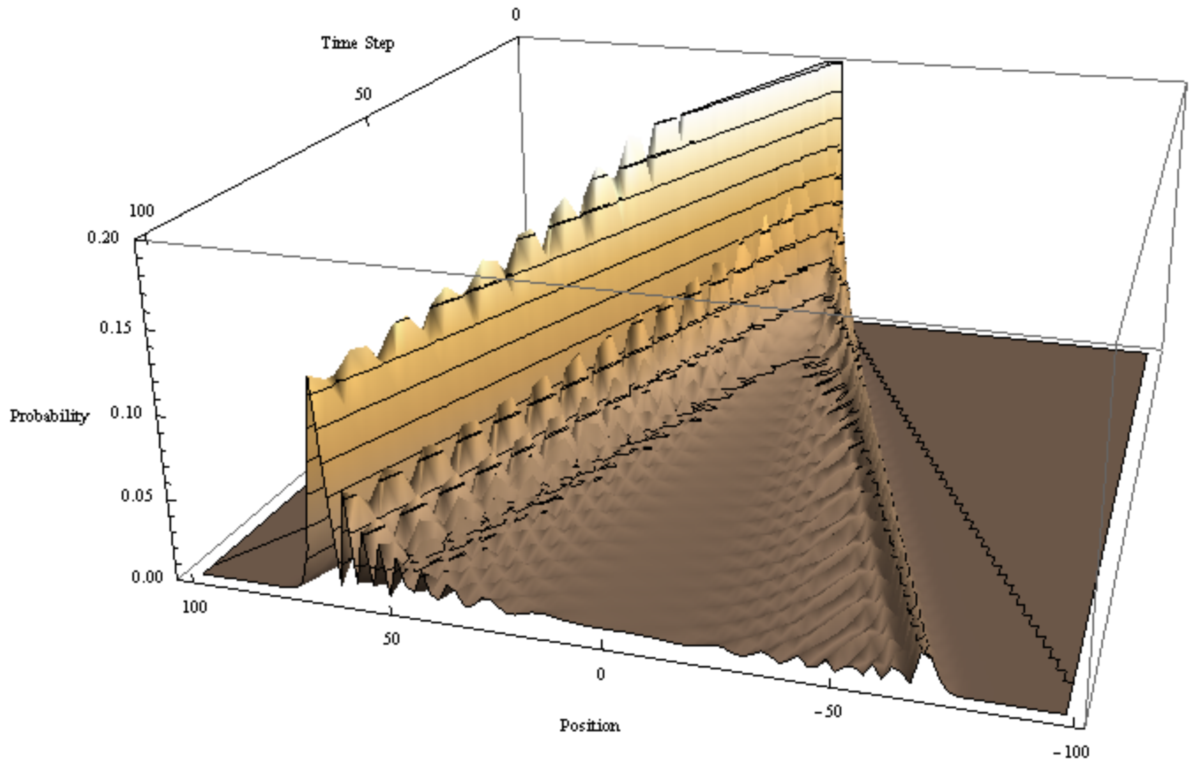


Figure 3-3: Qubit walk, B1 starting state

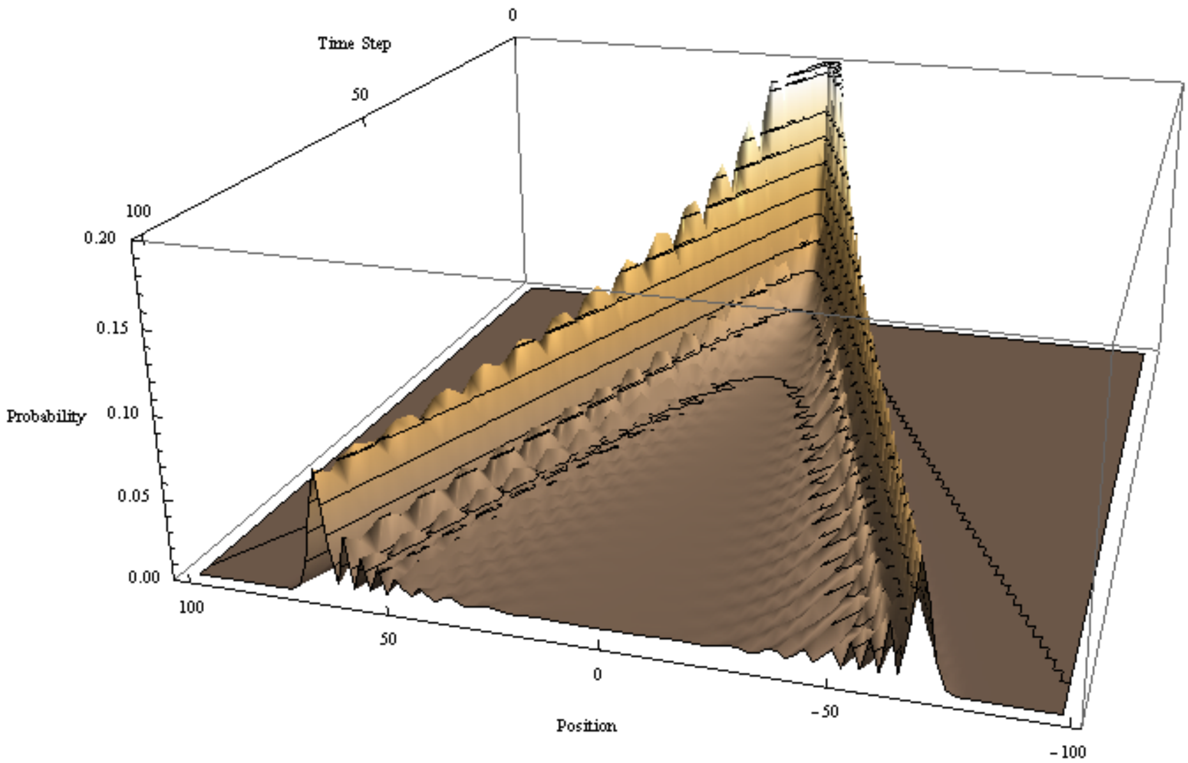


Figure 3-4: Qubit walk, B5 starting state

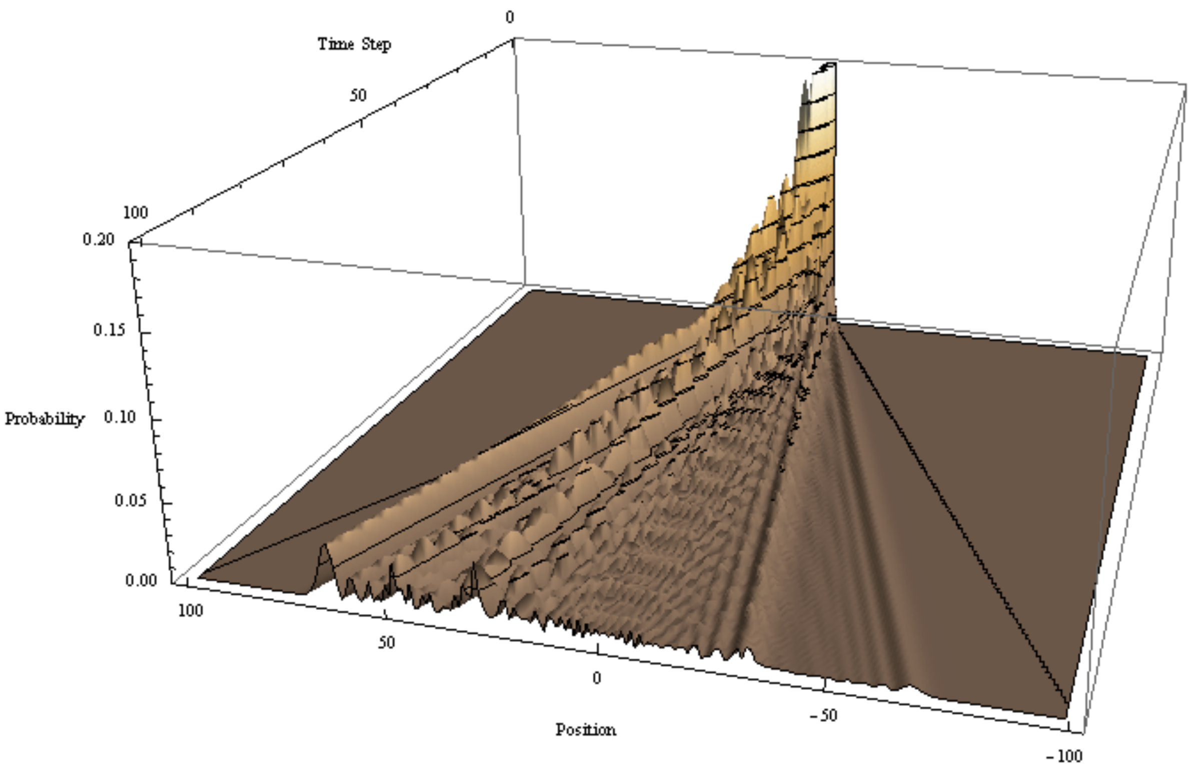


Figure 3-5: Qutrit walk, T1 starting state

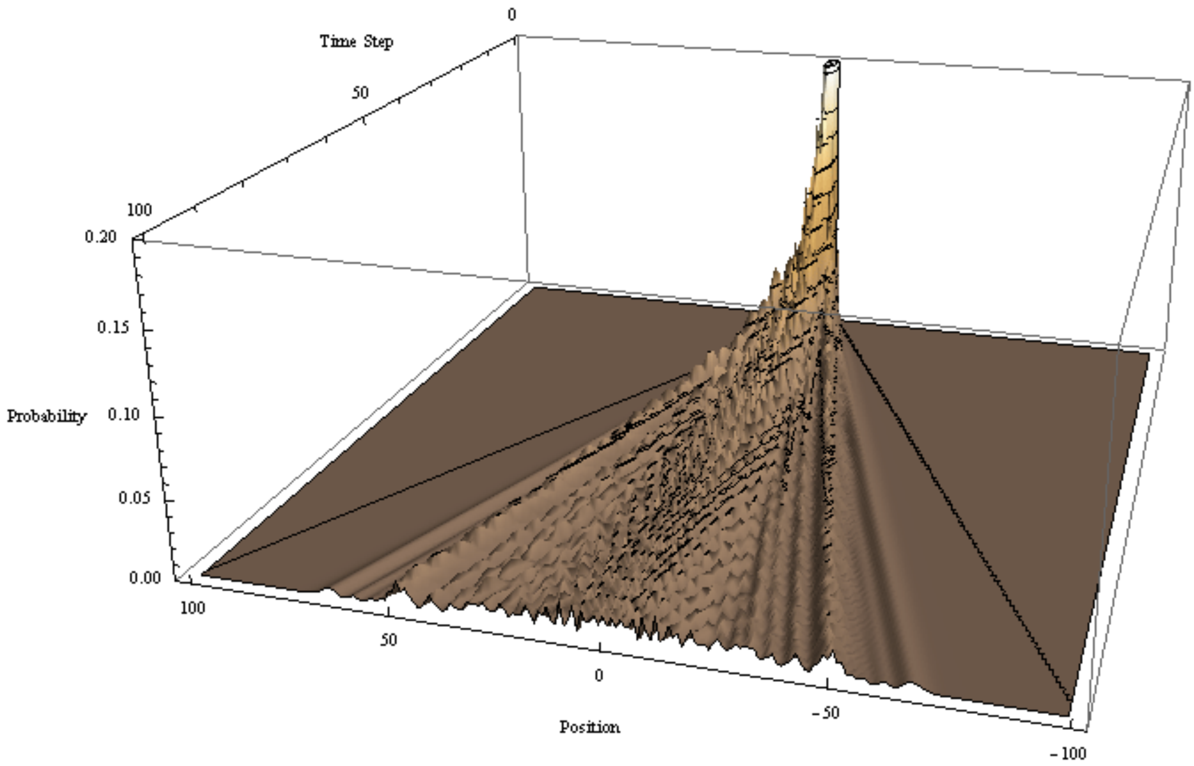


Figure 3-6: Qutrit walk, T2 / T3 starting state

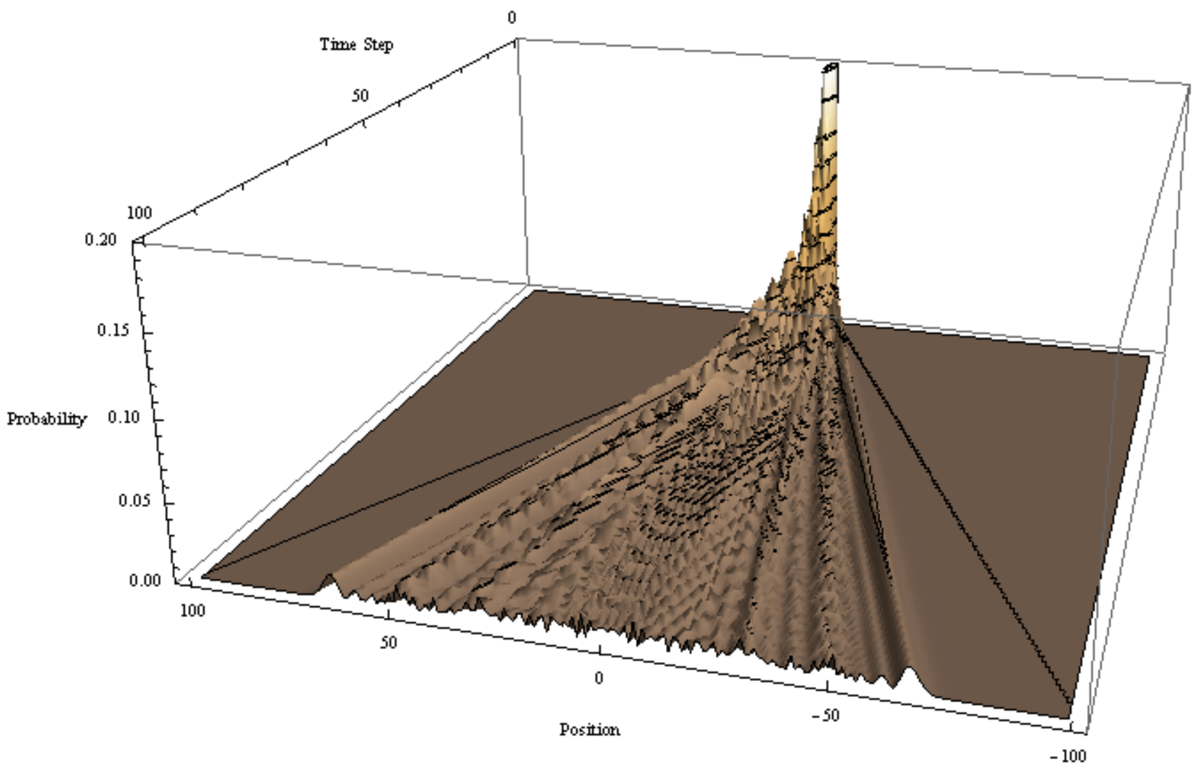


Figure 3-7: Qutrit walk, T4 starting state

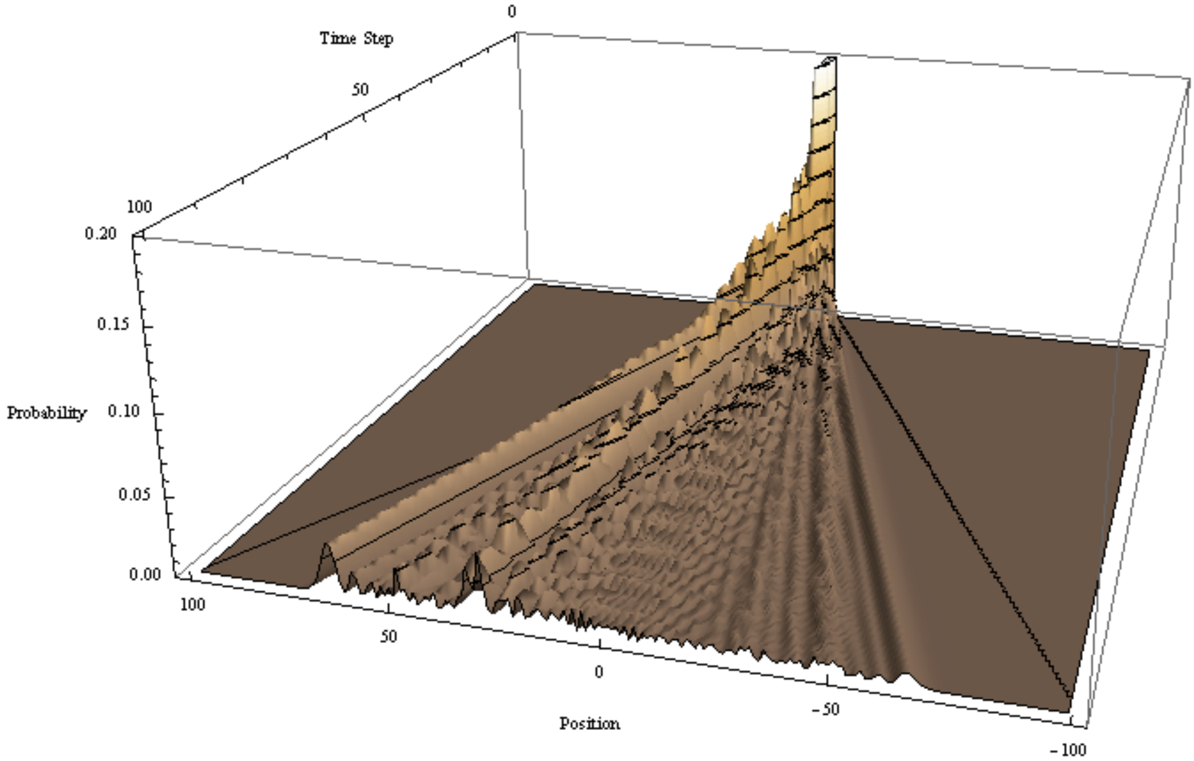


Figure 3-8: Qutrit walk, T5 starting state

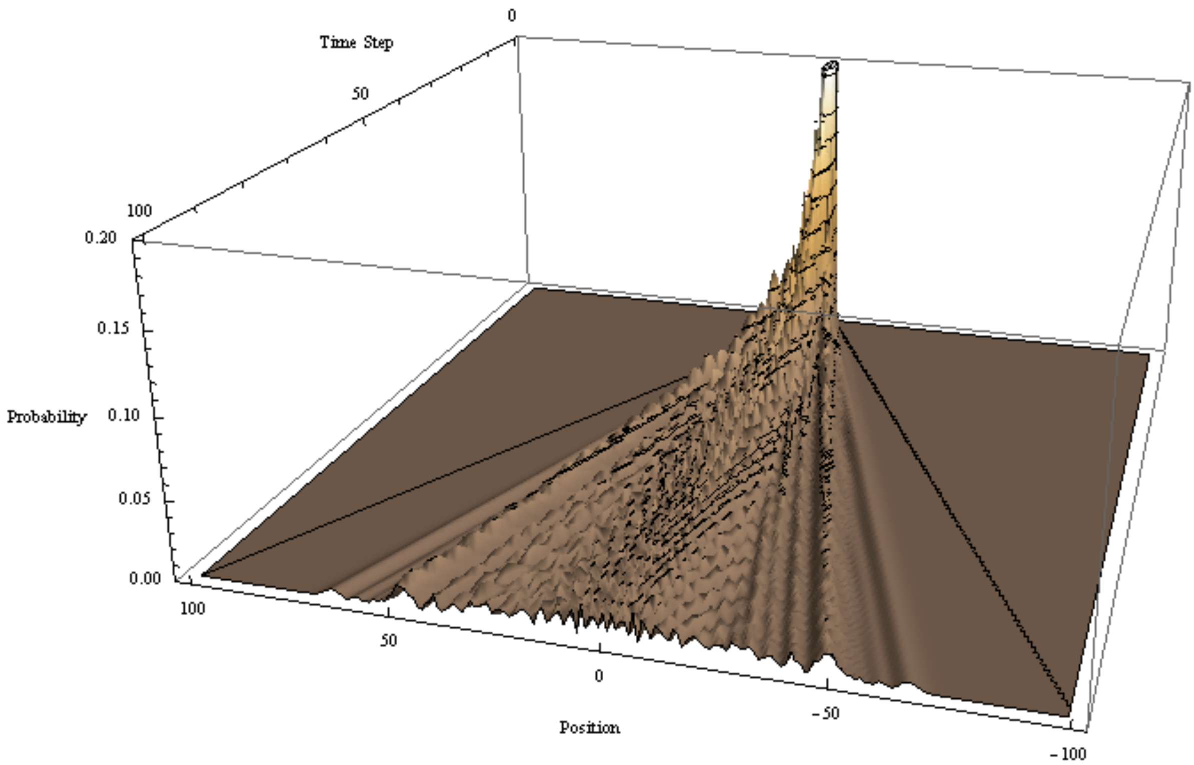


Figure 3-9: Qutrit walk, T6 starting state

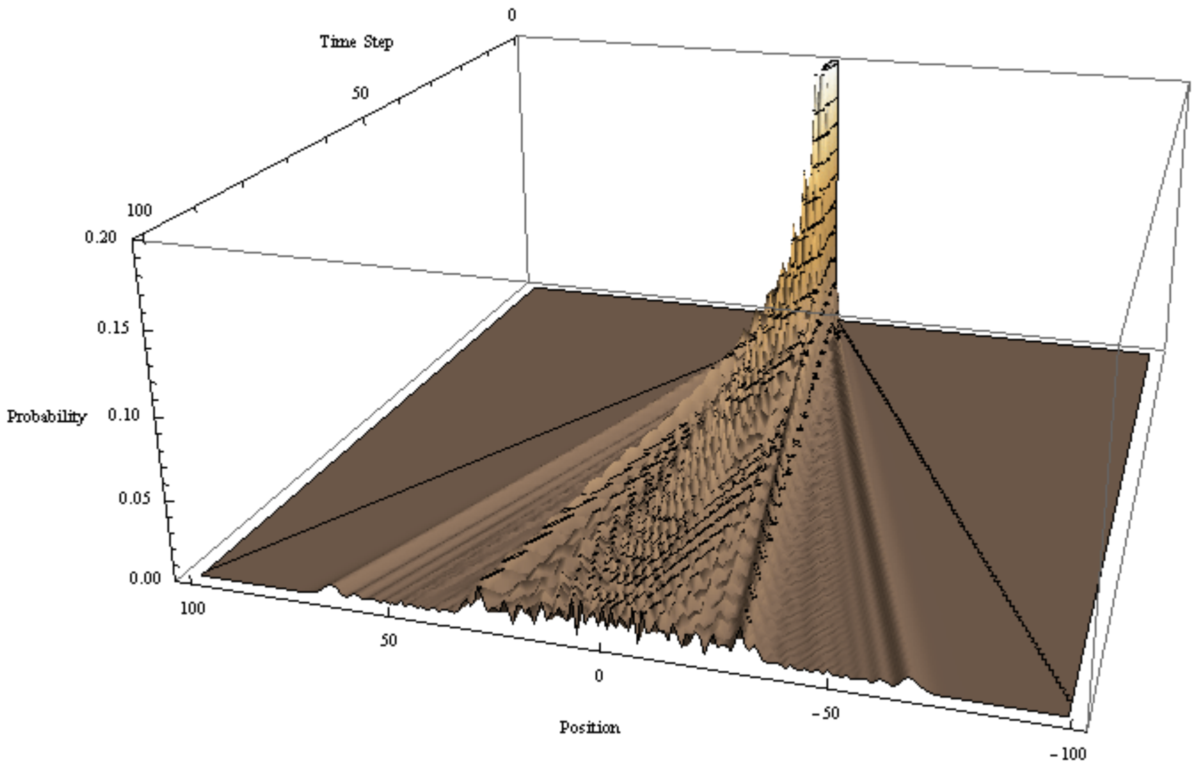
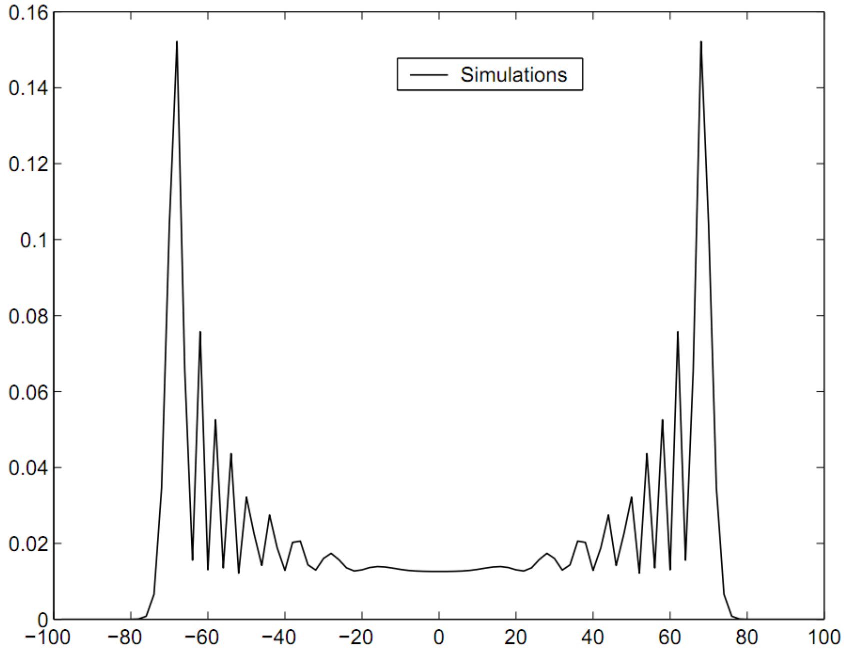


Figure 3-10: Qutrit walk, T7 starting state

3.1.2 Probability distributions

Graphs in this section show probability distributions with various starting states after 100 time steps. All graphs were generated with our Mathematica module. Initial states are listed on page 11.

Our graphs match those found elsewhere (Kempe, 2003).



3-11: Figure 6 from Kempe, 2003. The probability distribution obtained from a computer simulation of the Hadamard walk with a symmetric initial condition. The number of steps in the walk was taken to be 100. Only the probability at the even points is plotted, since the odd points have probability zero.

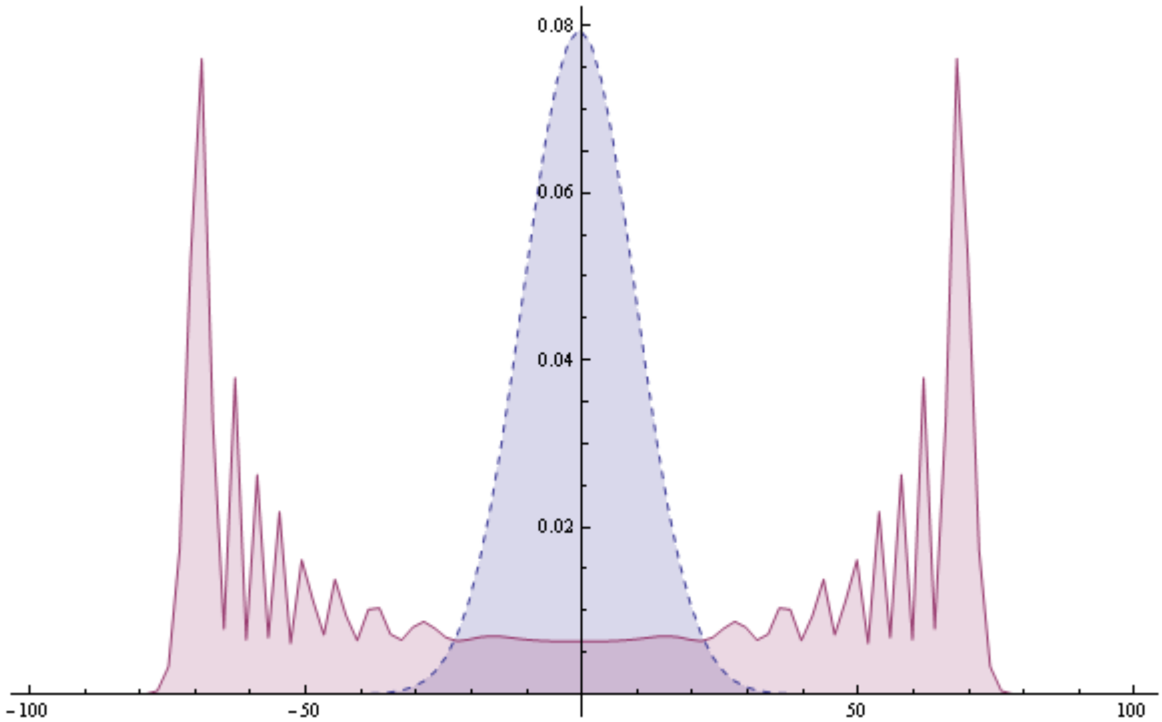


Figure 3-12: Non-lazy classical walk (dashed) and qubit walk, B5 starting state (solid)

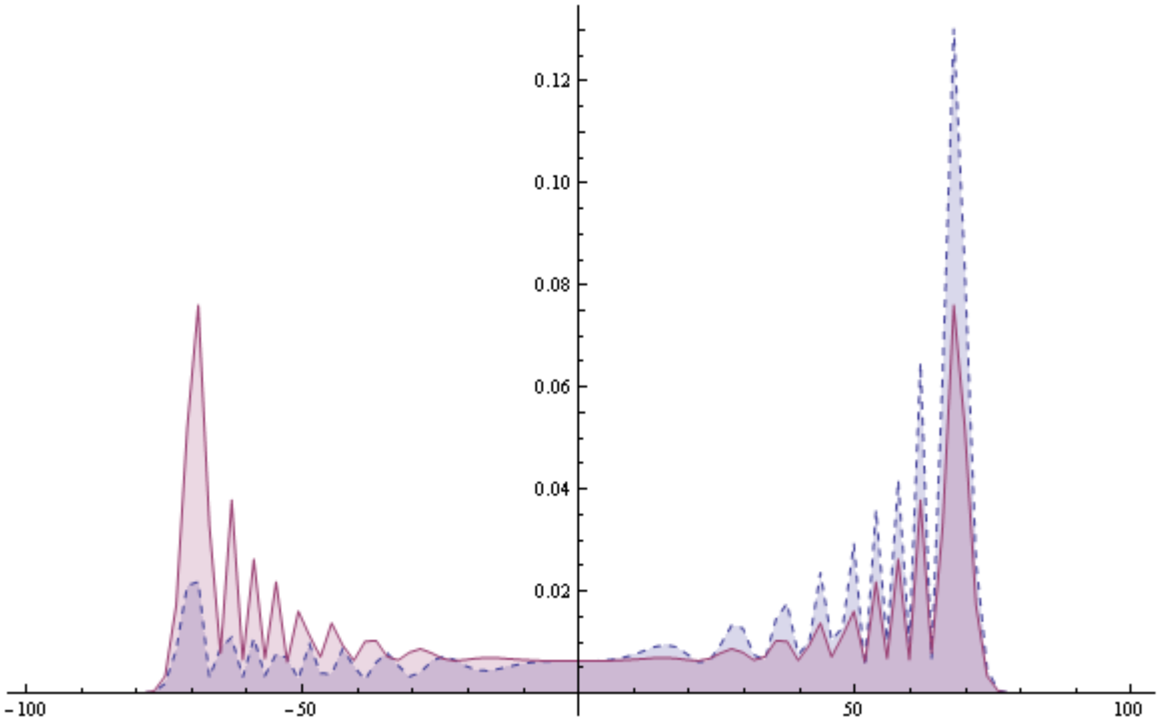


Figure 3-13: Qubit walks, starting state B1 (dashed) and B5 (solid)

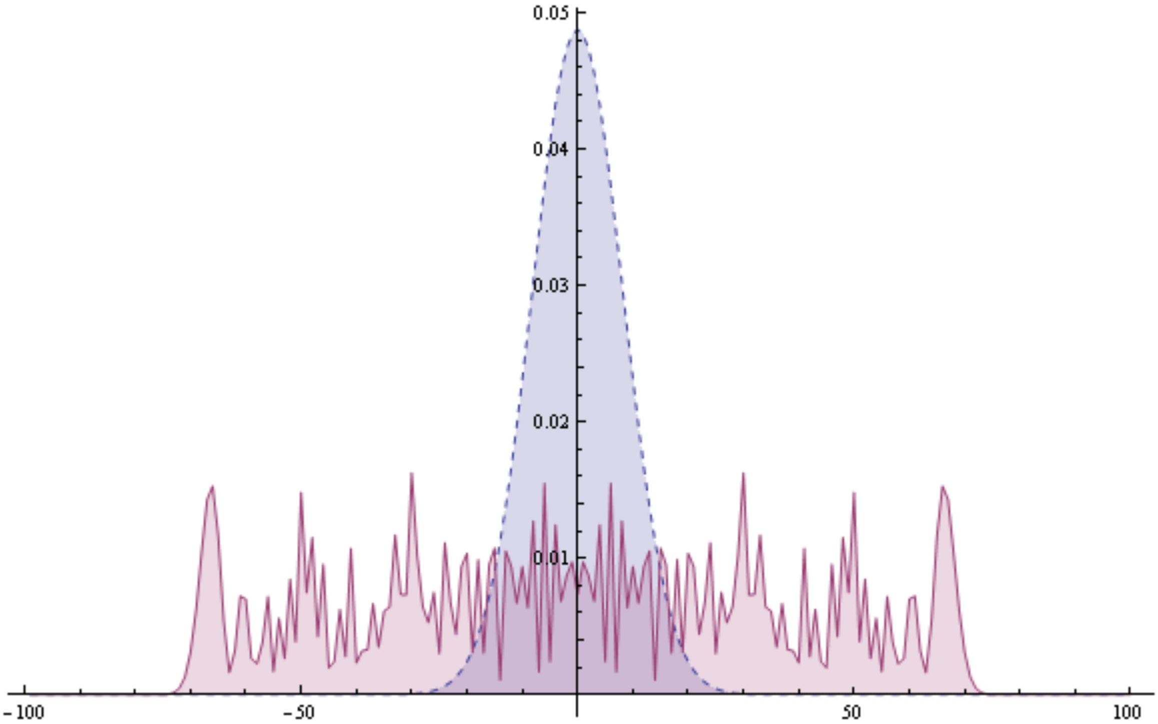


Figure 3-14: Lazy classical walk (dashed) and qutrit walk, T4 starting state (solid)

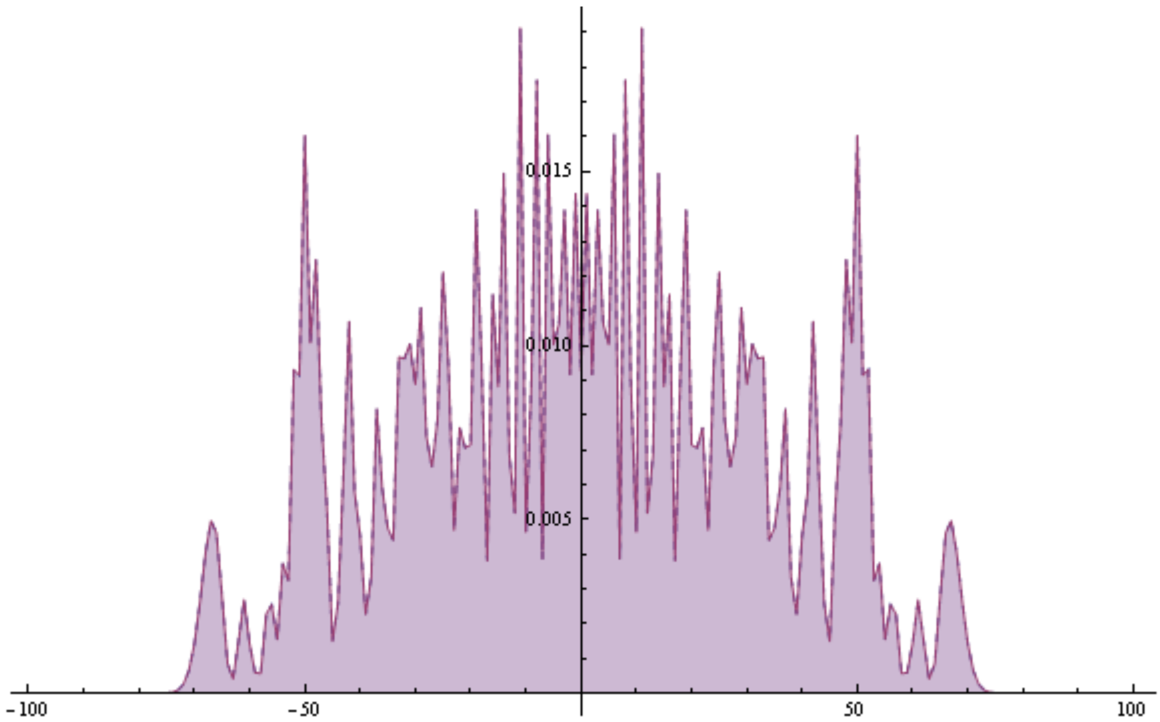


Figure 3-15: Qutrit walks, starting state T2 (dashed) and T3 (solid)

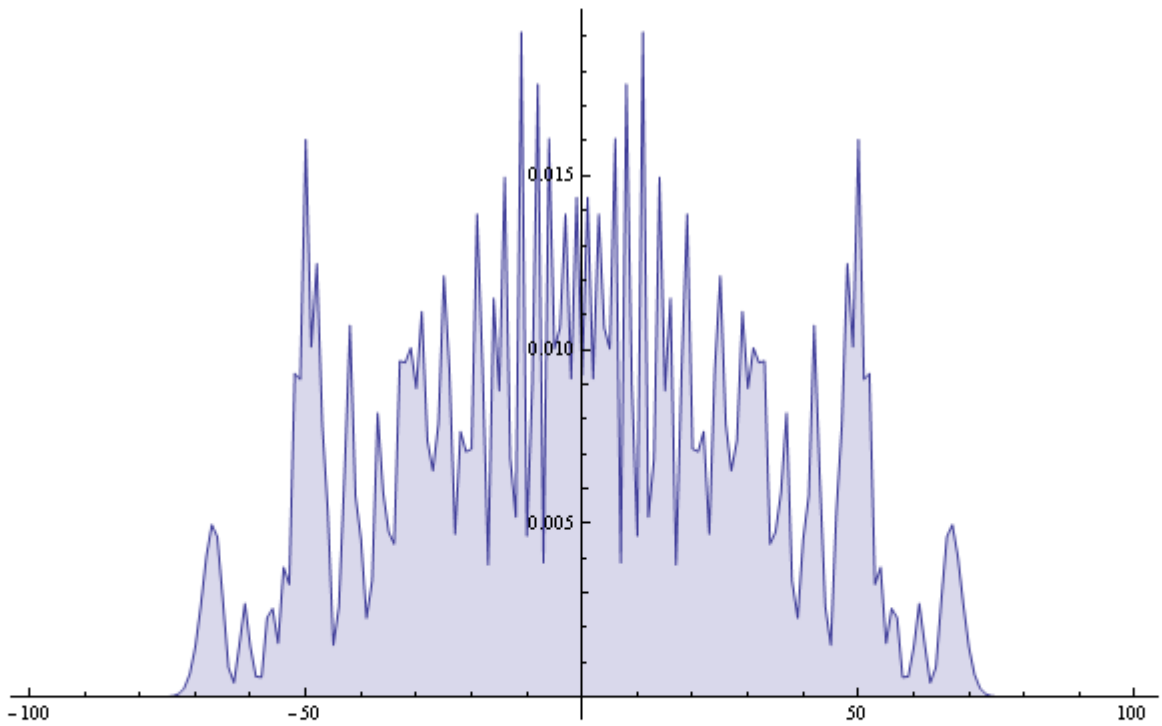


Figure 3-16: Qutrit walk, T2 / T3 starting state

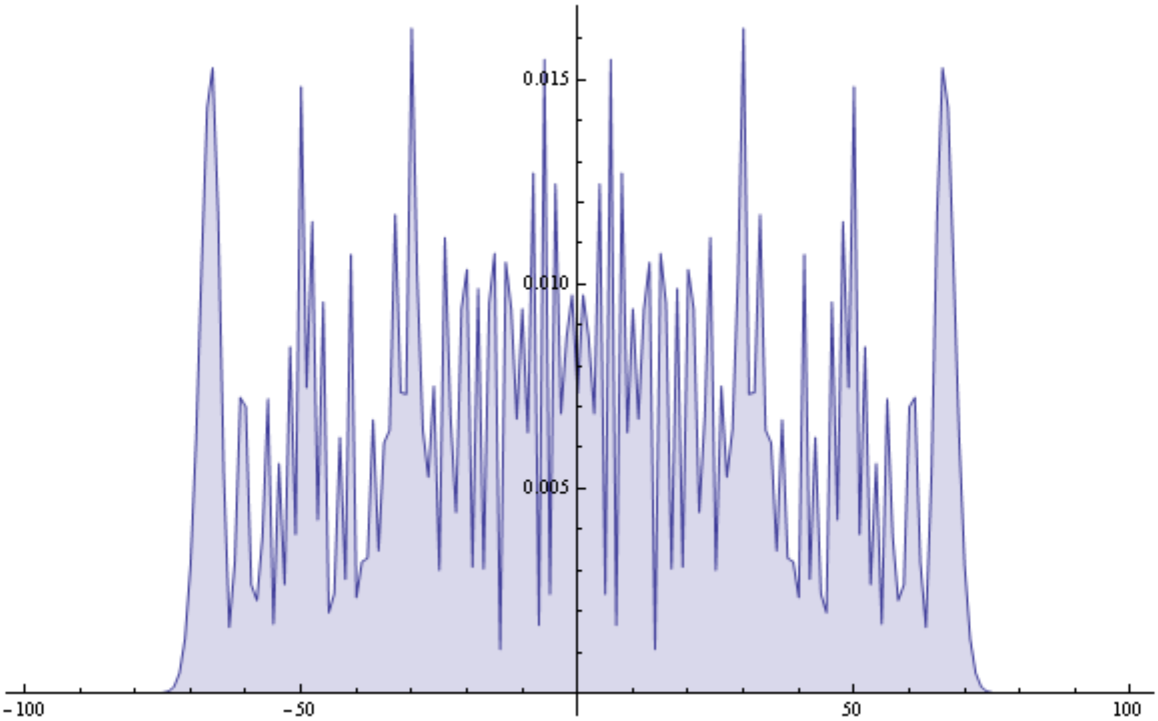


Figure 3-17: Qutrit walk, T4 starting state

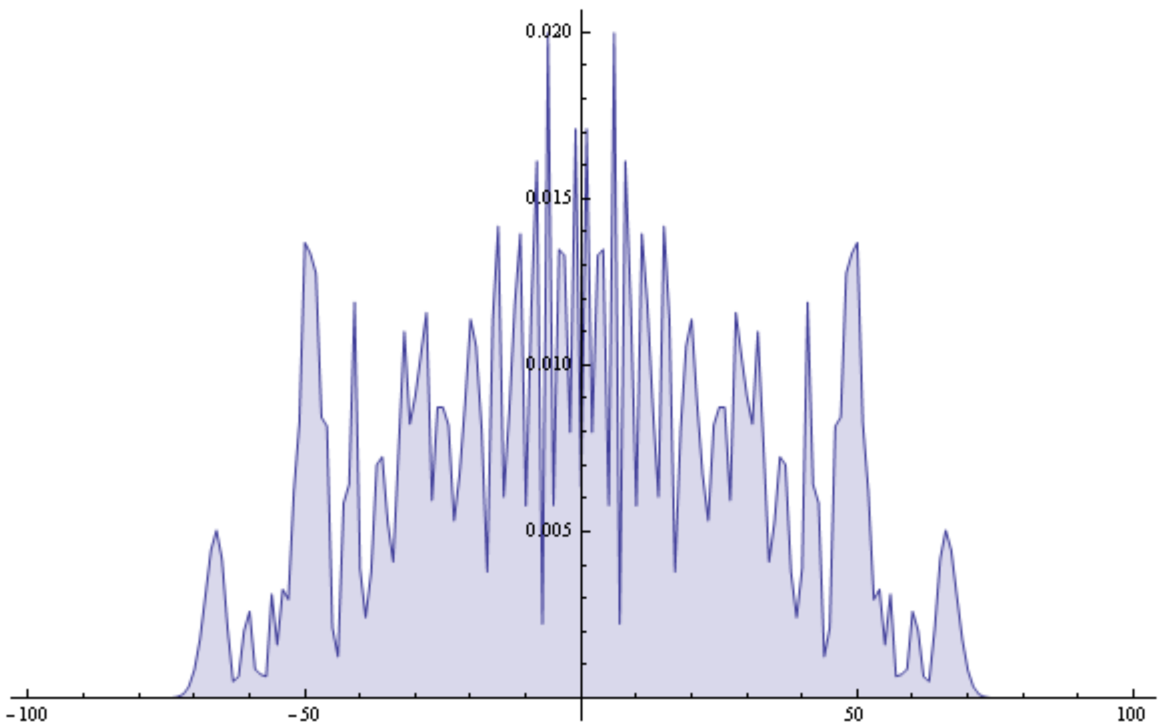


Figure 3-18: Qutrit walk, T6 starting state

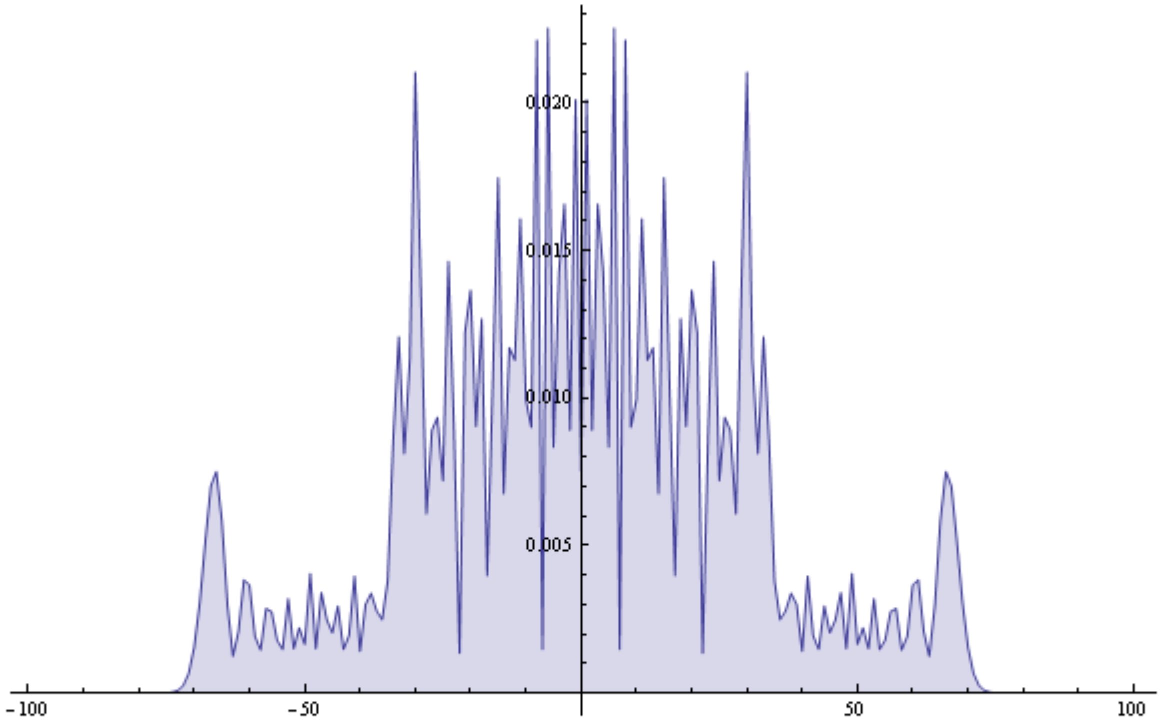


Figure 3-19: Qutrit walk, T7 starting state

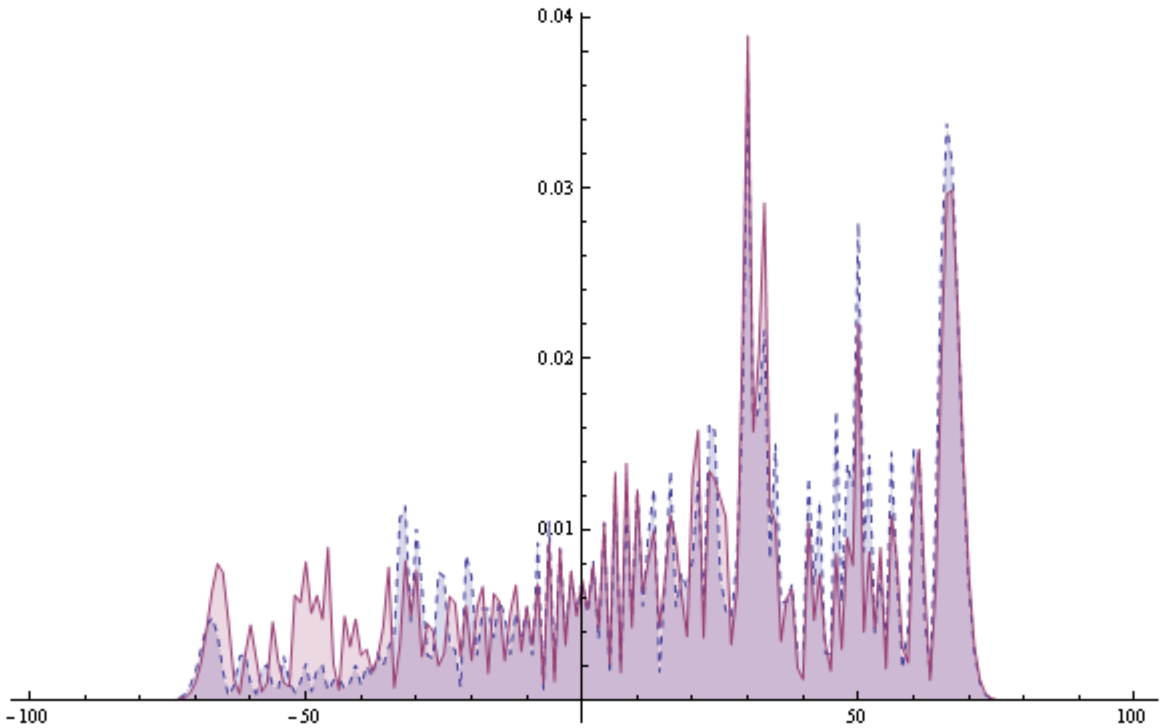


Figure 3-20: Qutrit walks, starting state T1 (dashed) and T5 (solid)

3.2 Walks on the Cycle

For walks on a cycle we take the cumulative mean of the nodal probabilities. After a large number of time steps certain patterns emerge. For cycles of even length the starting node and the node opposite the starting node have highest cumulative probabilities, whose values are approximately equal. For cycles of odd length the starting node always has the highest cumulative probability.

All graphs show the cumulative probabilities over the first 500 time steps.

3.2.1 Cumulative probability distribution evolution

Graphs in this section show how the probability distribution changes over time. Results have been shown with the origin position facing the viewer.

Some graphs show a probability of zero at time step zero on the starting node. This is an artefact in Mathematica's graphing library.

Initial states are listed on page 11.

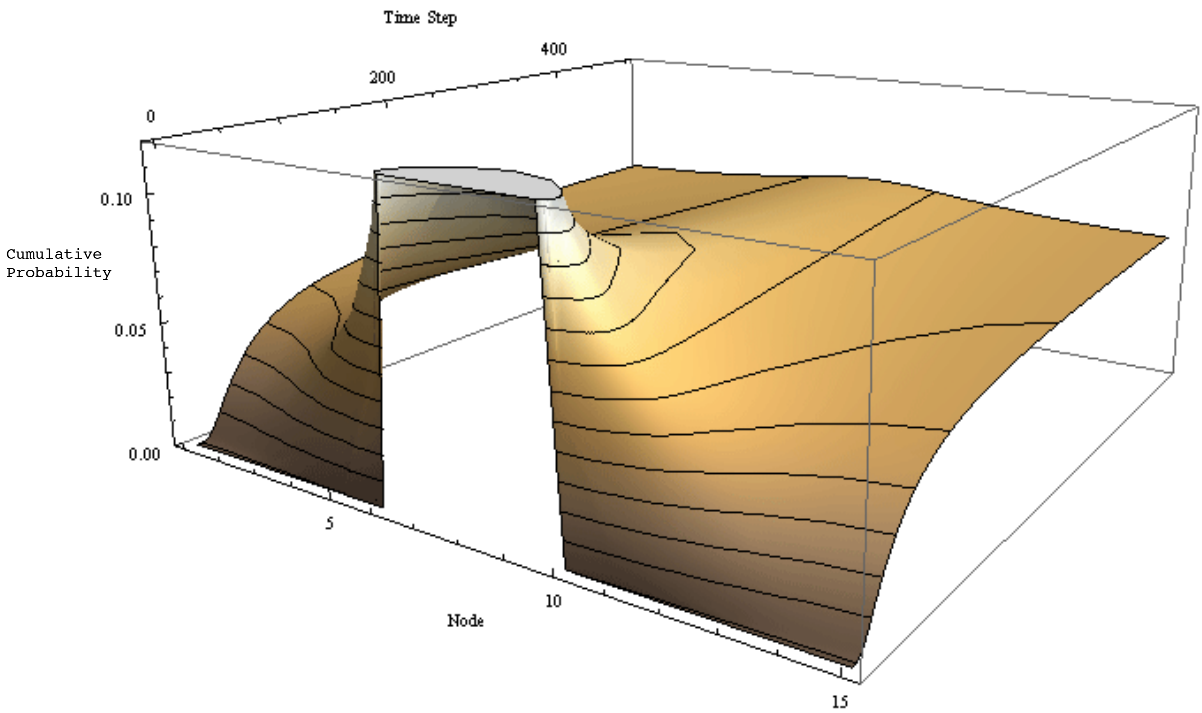


Figure 3-21: Non-lazy classical walk on a 15 node cycle

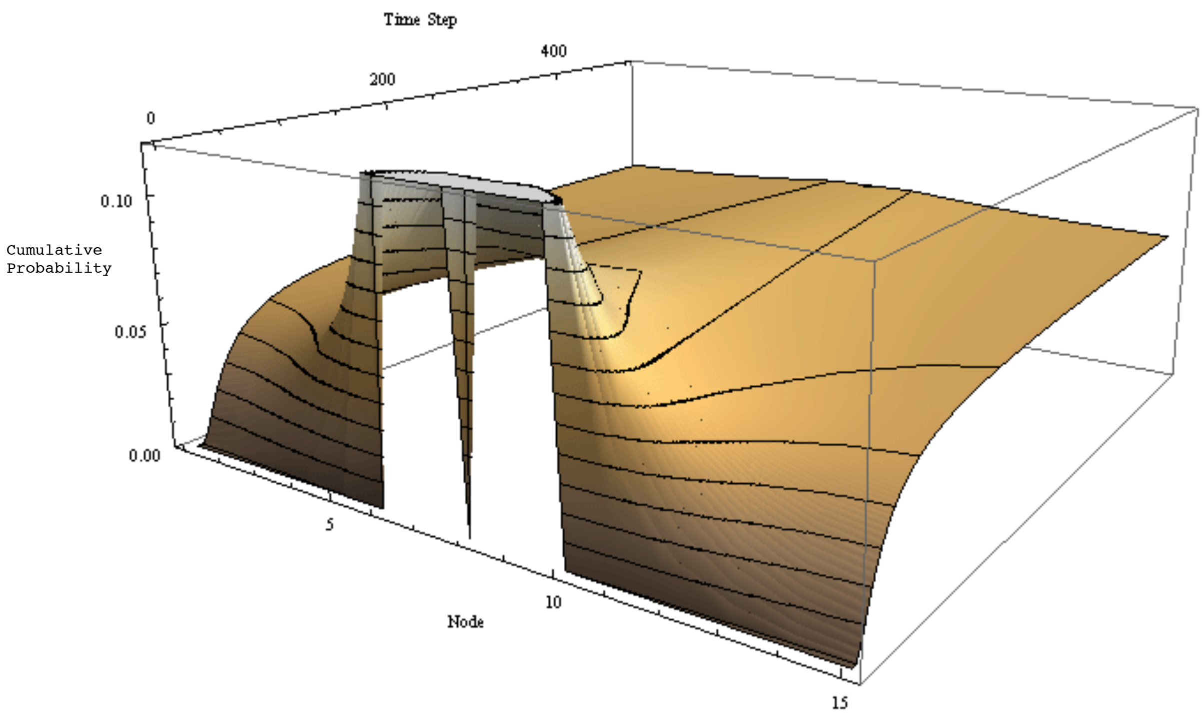


Figure 3-22: Lazy classical walk on a 15 node cycle

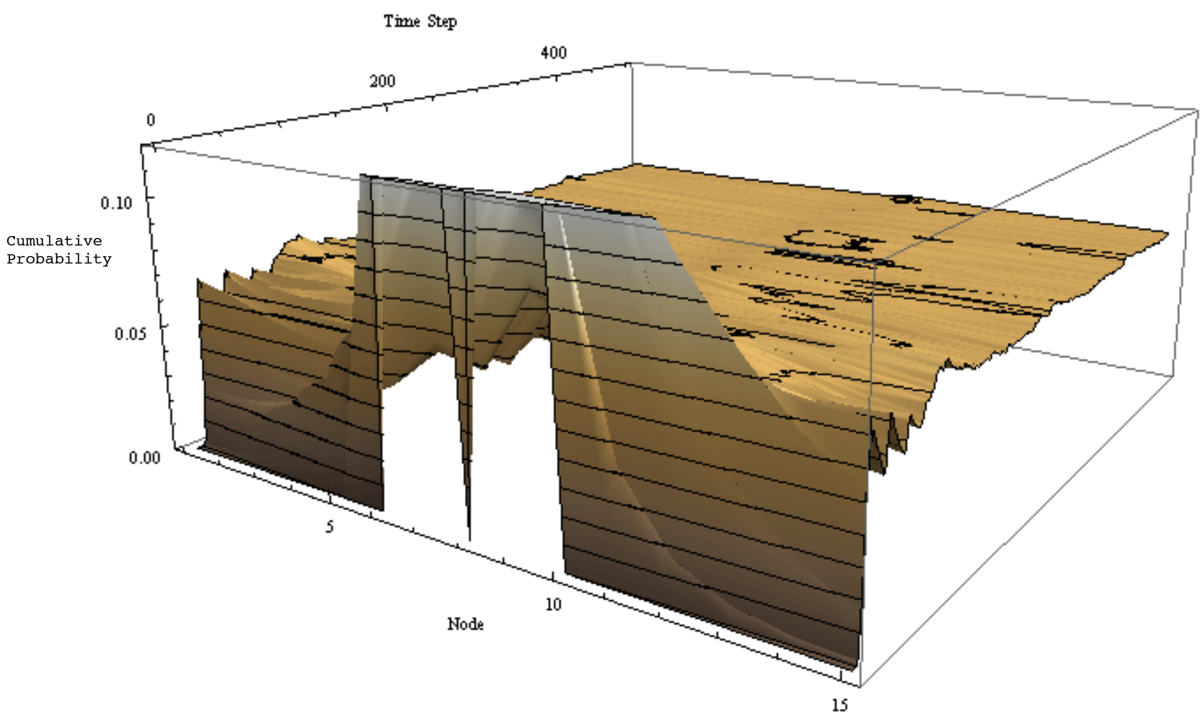


Figure 3-23: Qubit walk on a 15 node cycle, starting state B1

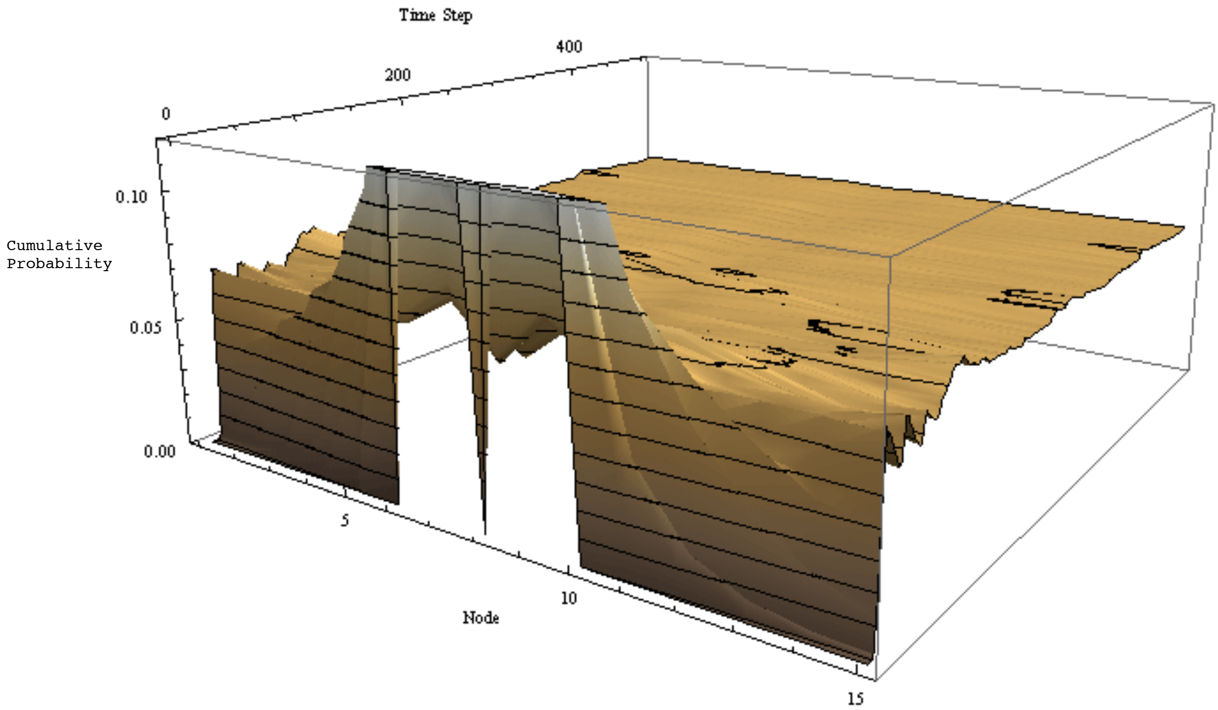


Figure 3-24: Qubit walk on a 15 node cycle, starting state B5

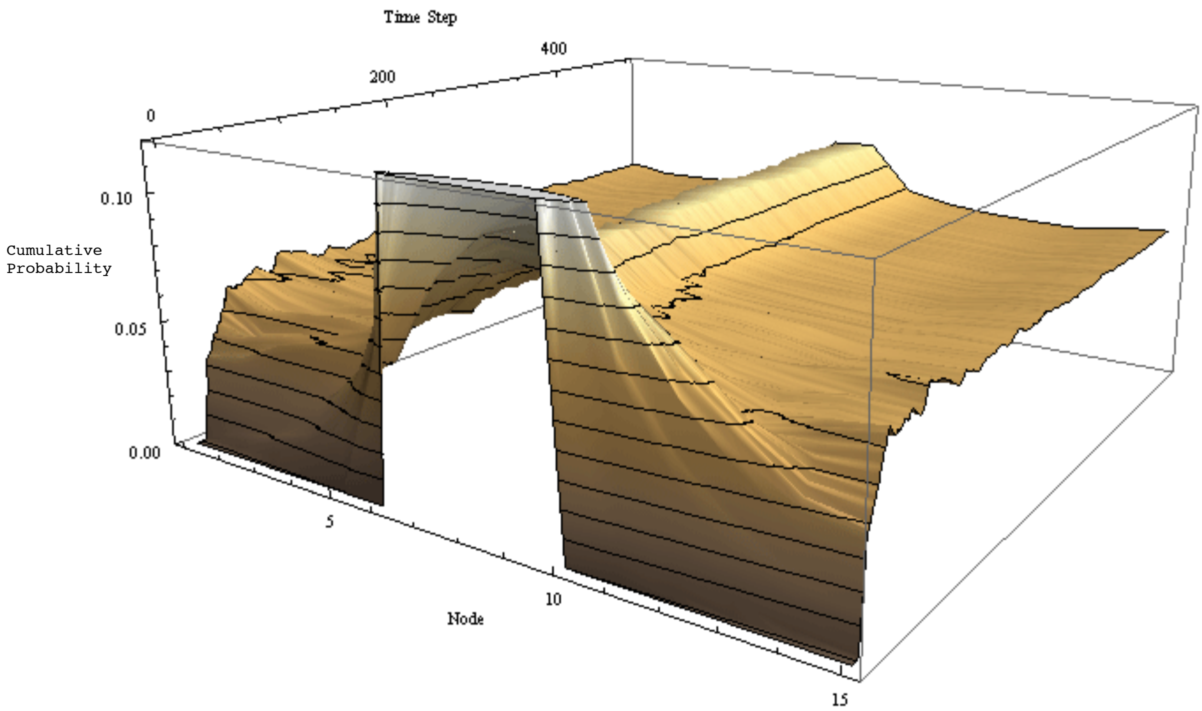


Figure 3-25: Qutrit walk on a 15 node cycle, starting state T1

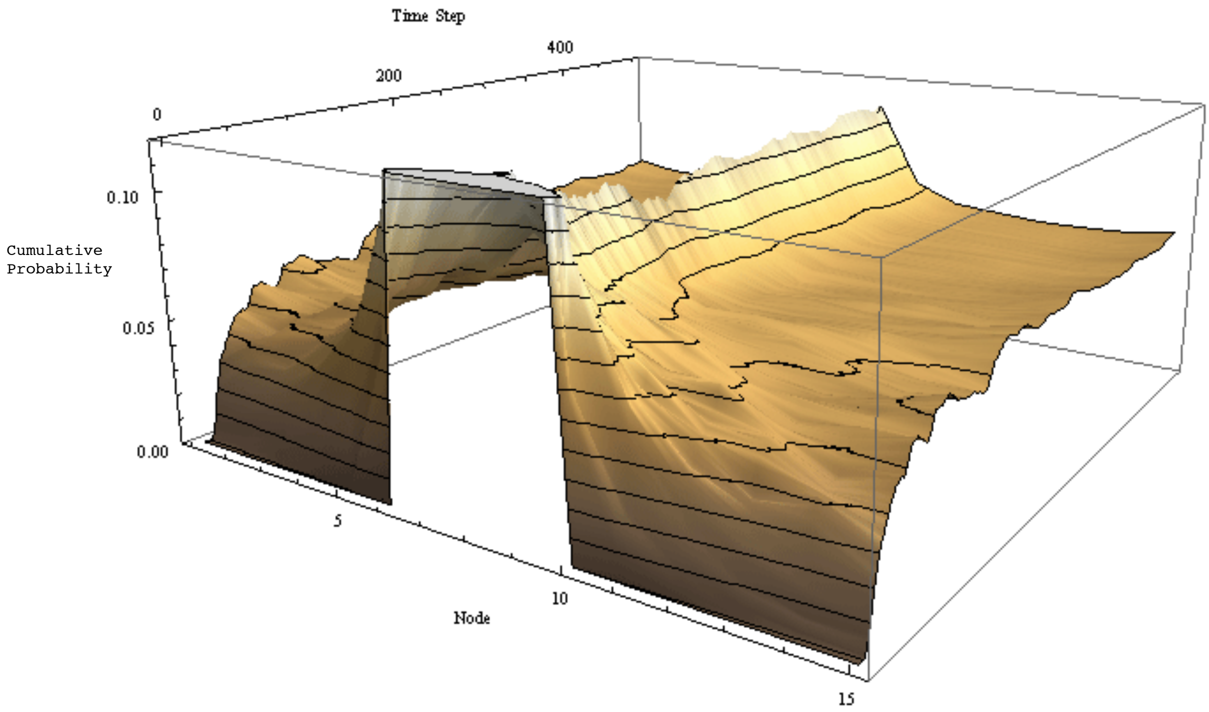


Figure 3-26: Qutrit walk on a 15 node cycle, starting state T2

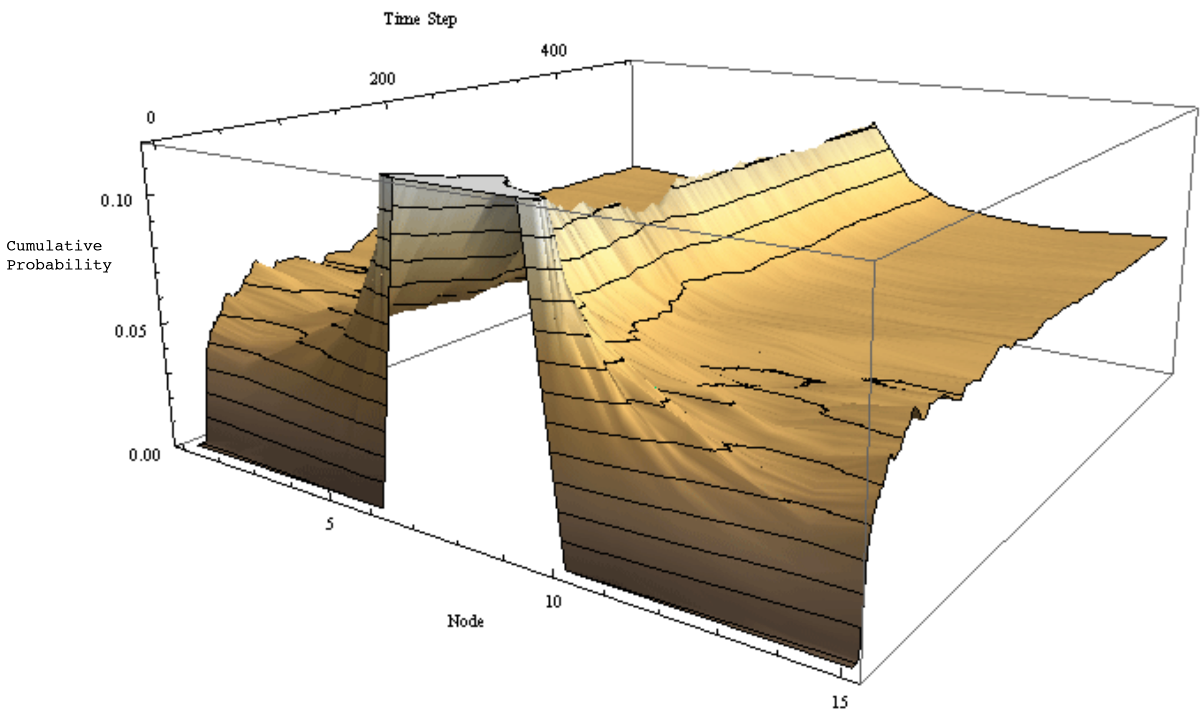


Figure 3-27: Qutrit walk on a 15 node cycle, starting state T4

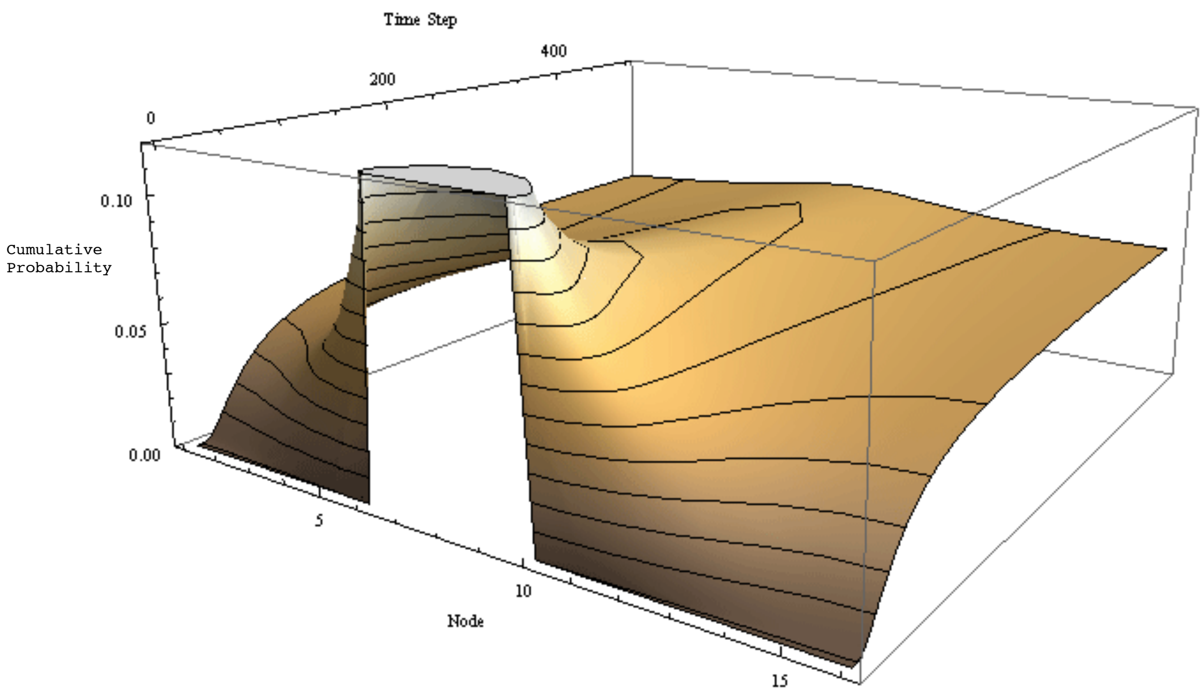


Figure 3-28: Non-lazy classical walk on a 16 node cycle

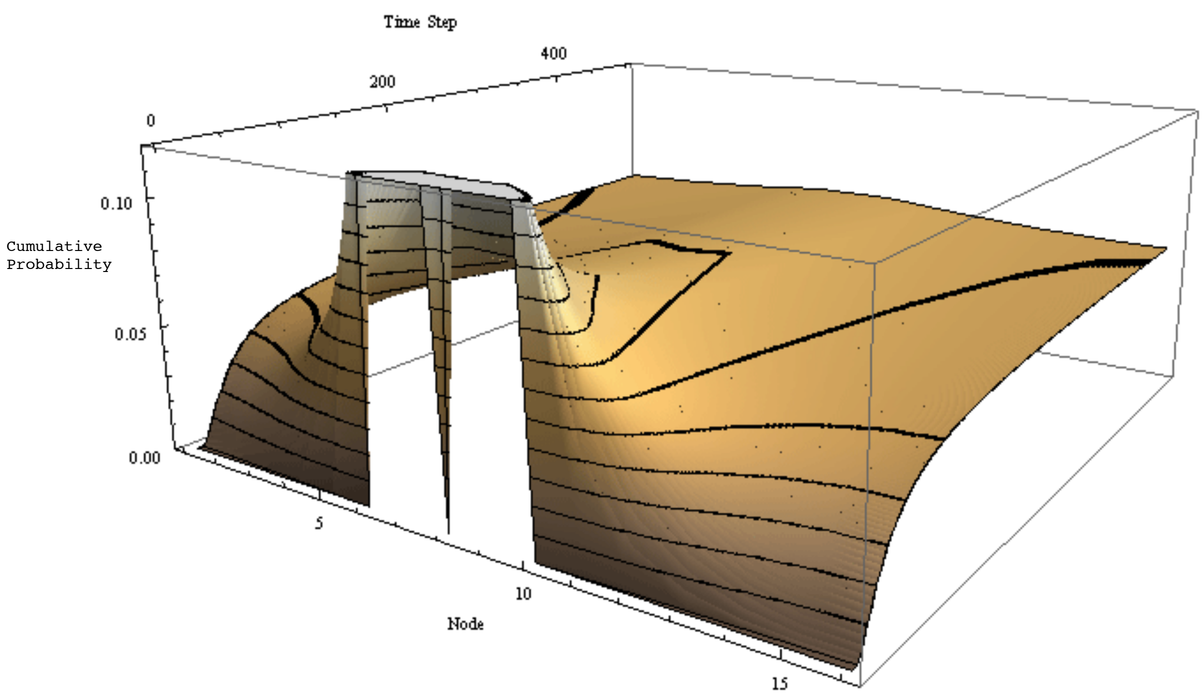


Figure 3-29: Lazy classical walk on a 16 node cycle

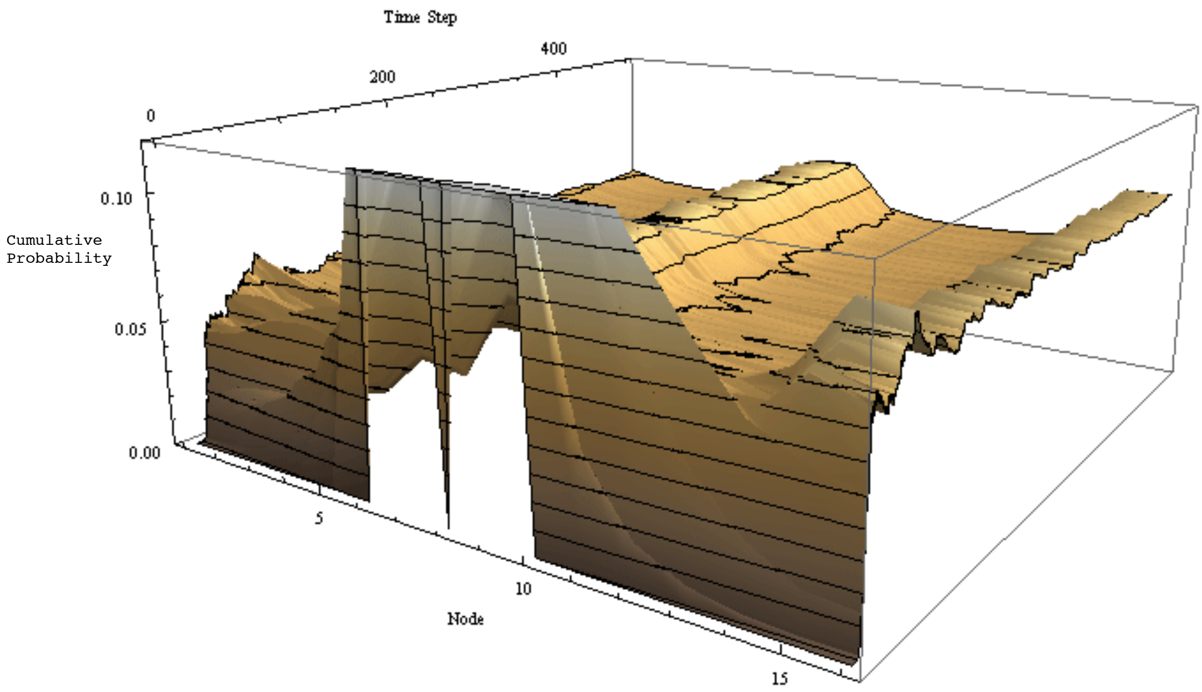


Figure 3-30: Qubit walk on a 16 node cycle, starting state B1

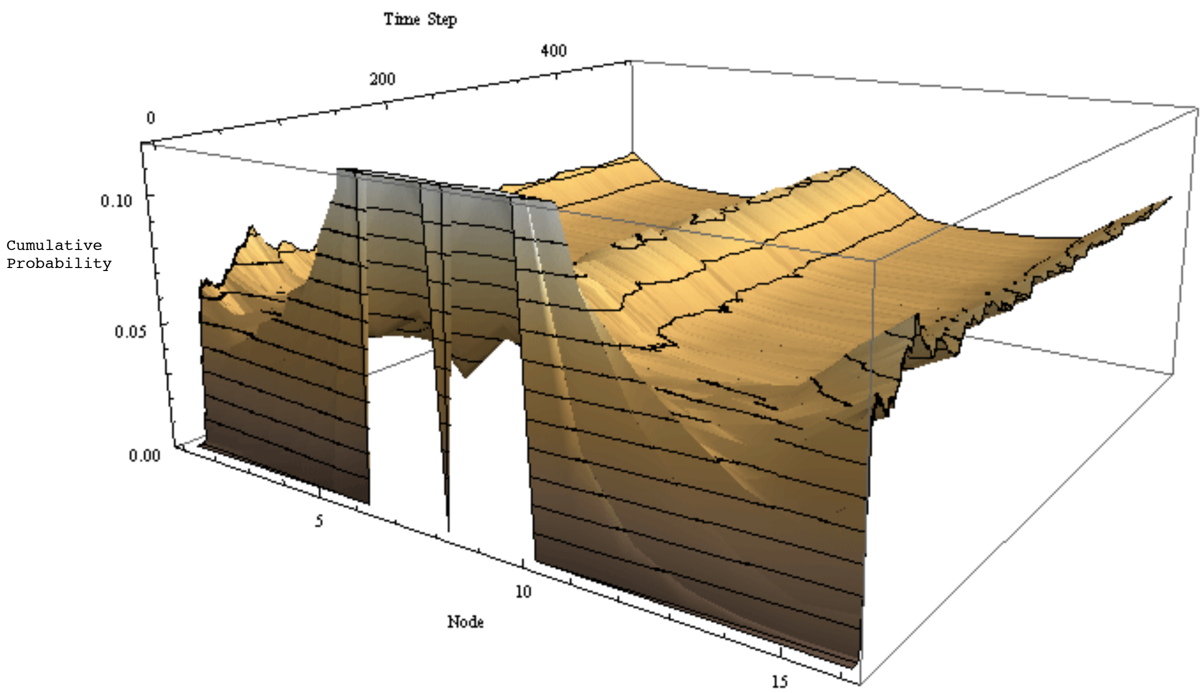


Figure 3-31: Qubit walk on a 16 node cycle, starting state B5

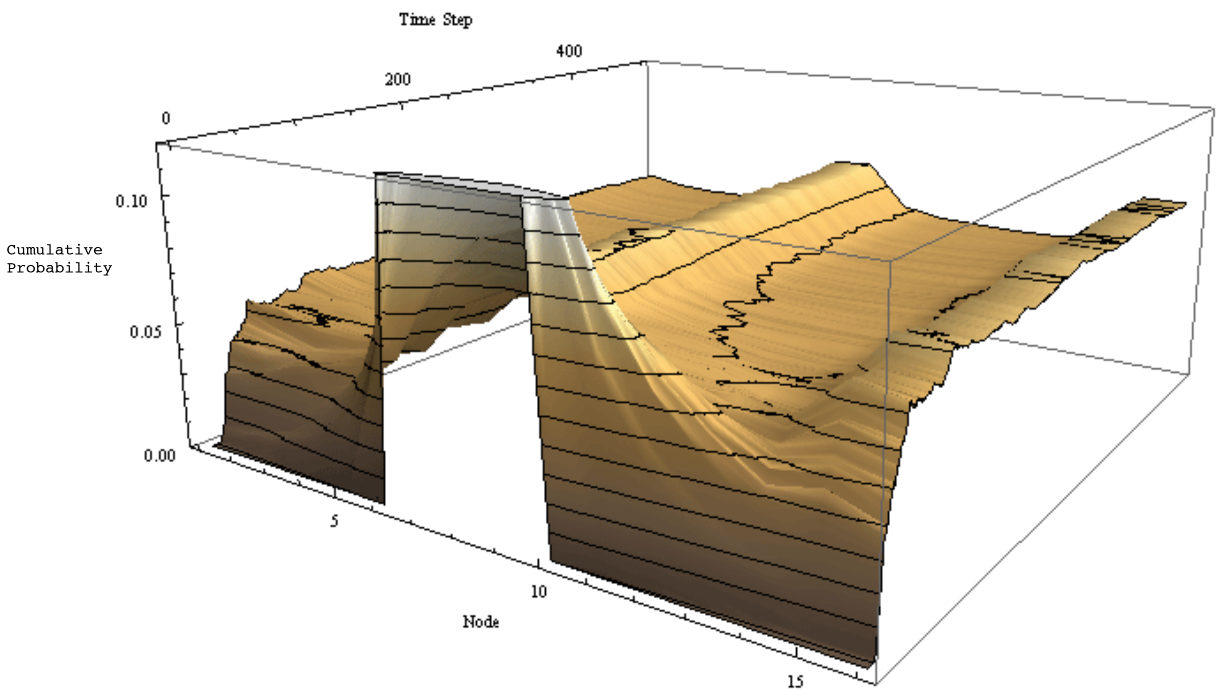


Figure 3-32: Qutrit walk on a 16 node cycle, starting state T1

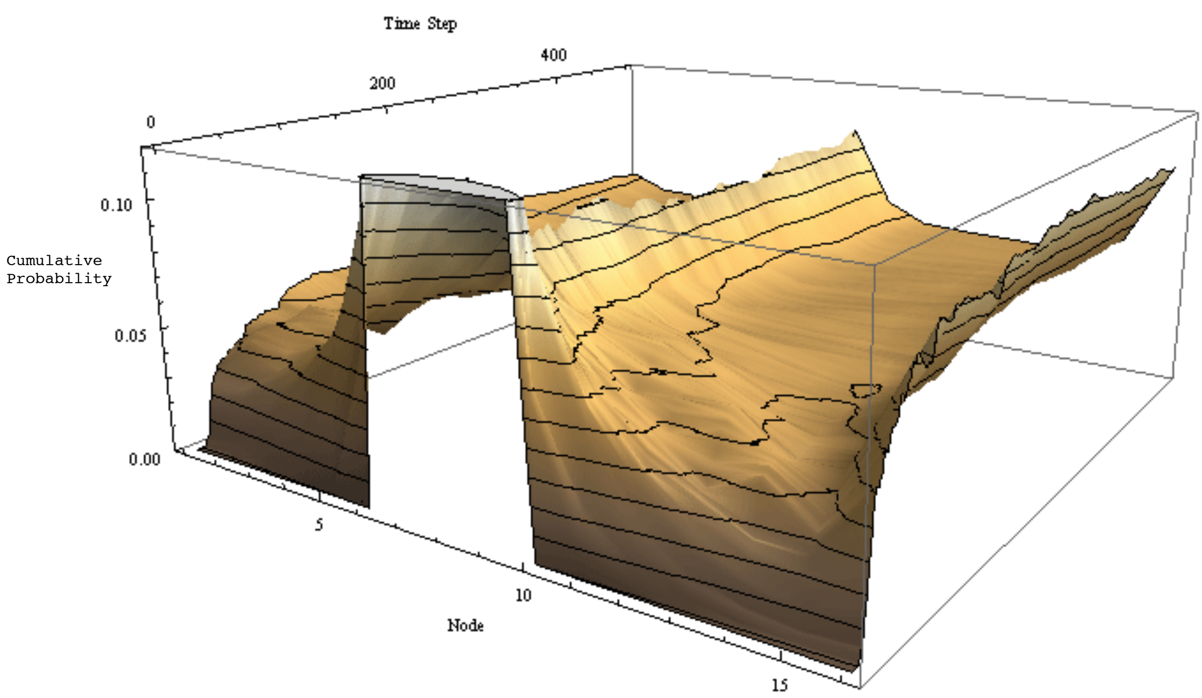


Figure 3-33: Qutrit walk on a 16 node cycle, starting state T2

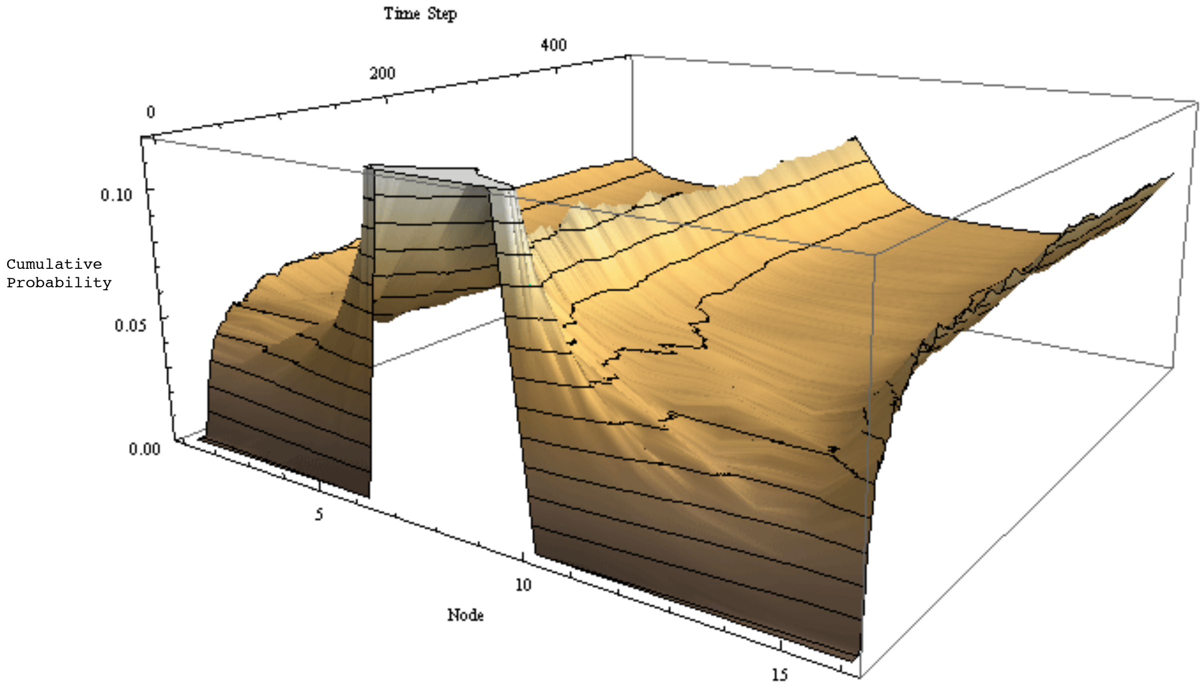


Figure 3-34: Qutrit walk on a 16 node cycle, starting state T4

3.2.2 Cumulative probability distribution

The following graphs compare the cumulative probability distribution for symmetric qubit and qutrit walks on a cycle. They each show the cumulative probability after 1000 time steps.

The cumulative probability distributions of both a lazy and non---lazy classical walk tend toward a flat distribution for cycles of odd and even node length. In this regard, the qubit walk on a cycle of odd length appears to exhibit similar properties.

All other symmetrical cumulative probability distributions differ from their classical counterparts. Initial states are listed on page 11.

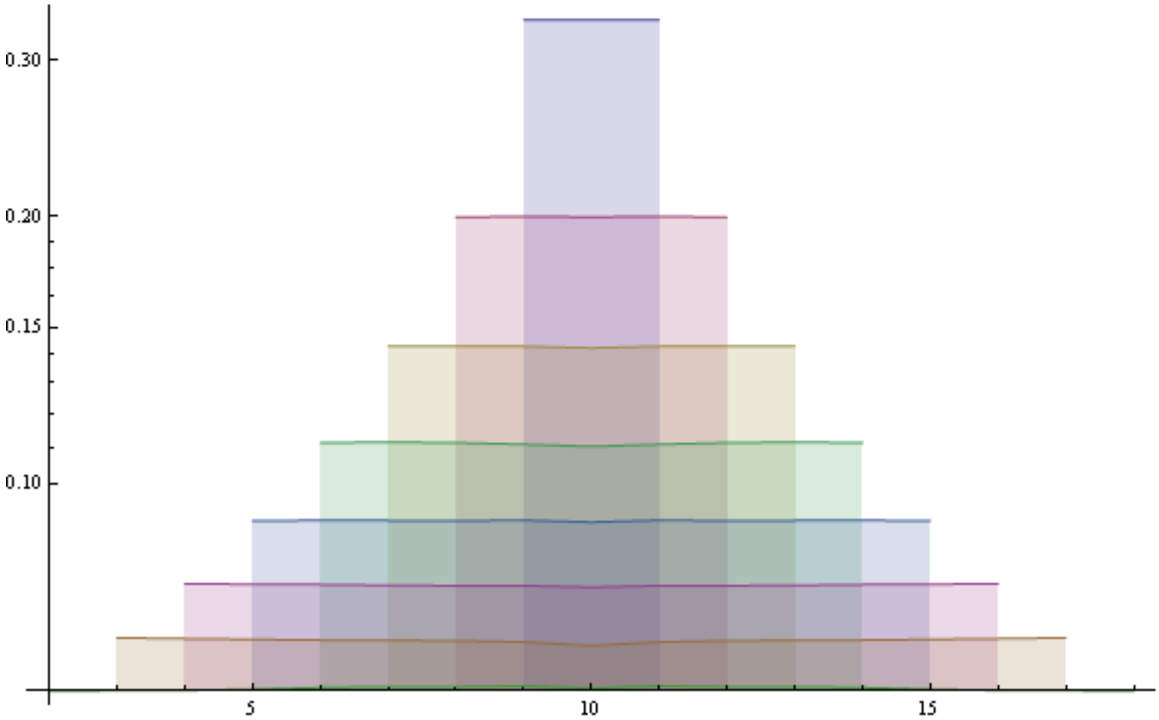


Figure 3-35: Cumulative probability for a qubit walk on a cycle after 1000 time steps. B5 starting state, {3, 5, 7, 9, 11, 13, 15, 17} nodes

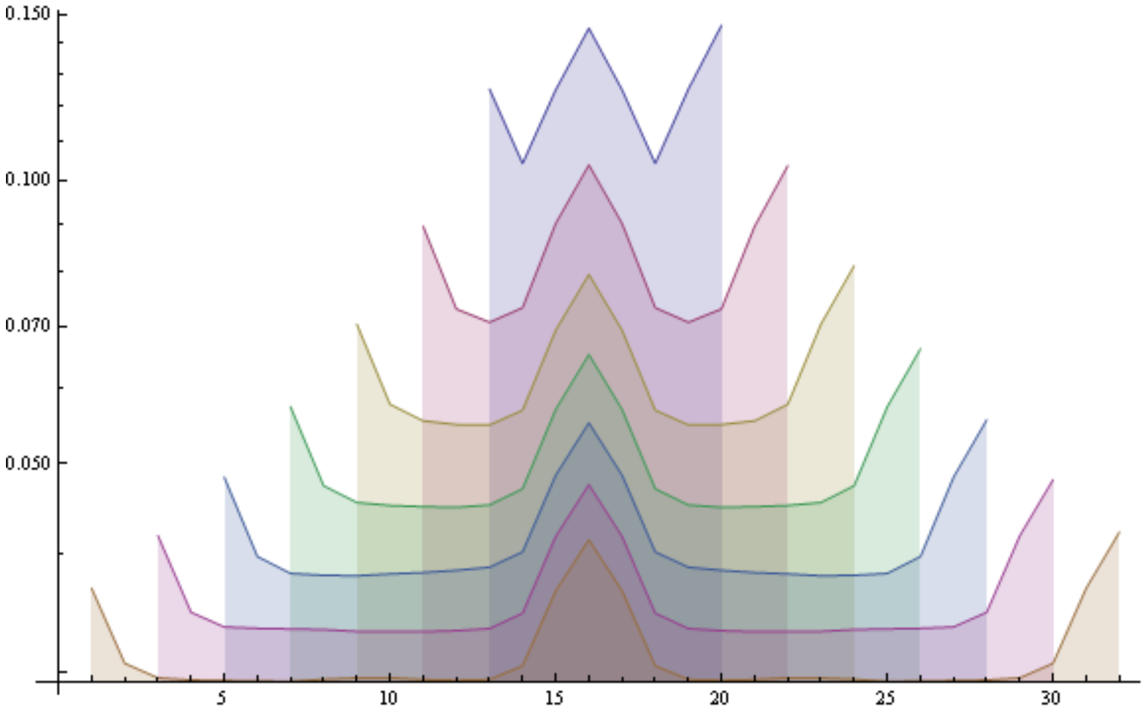


Figure 3-36: Cumulative probability for a qubit walk on a cycle after 1000 time steps. B5 starting state, {8, 12, 16, 20, 24, 28, 32} nodes

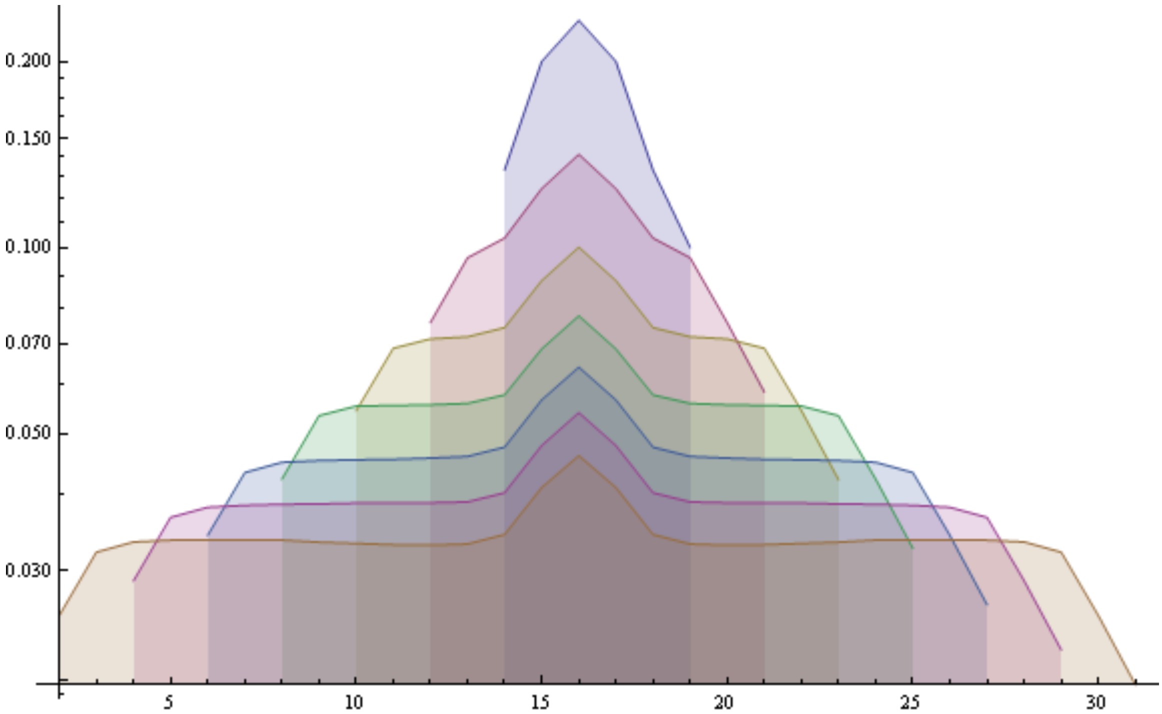


Figure 3-37: Cumulative probability for a qubit walk on a cycle after 1000 time steps. B5 starting state, {10, 14, 18, 22, 26, 30} nodes

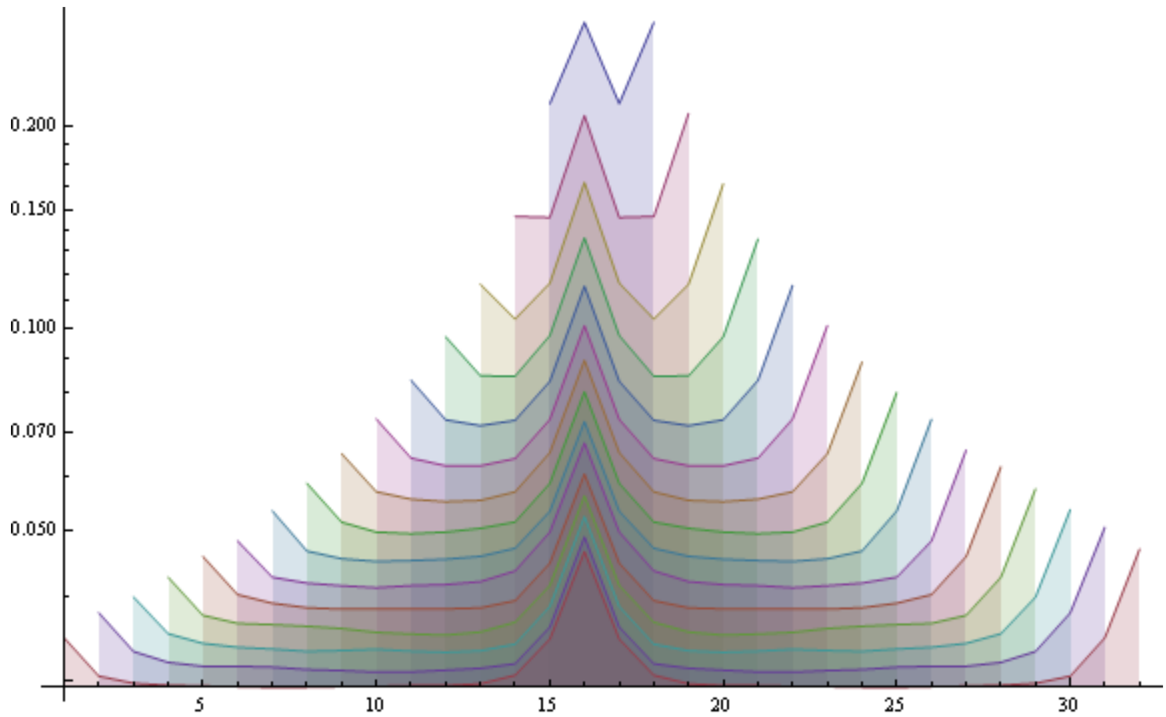


Figure 3-38: Cumulative probability for a qutrit walk on a cycle after 1000 time steps. T4 starting state, {4, 6, 8, 10, 12, 14, 16, 18, 20, 22, 24, 26, 28, 30, 32} nodes.

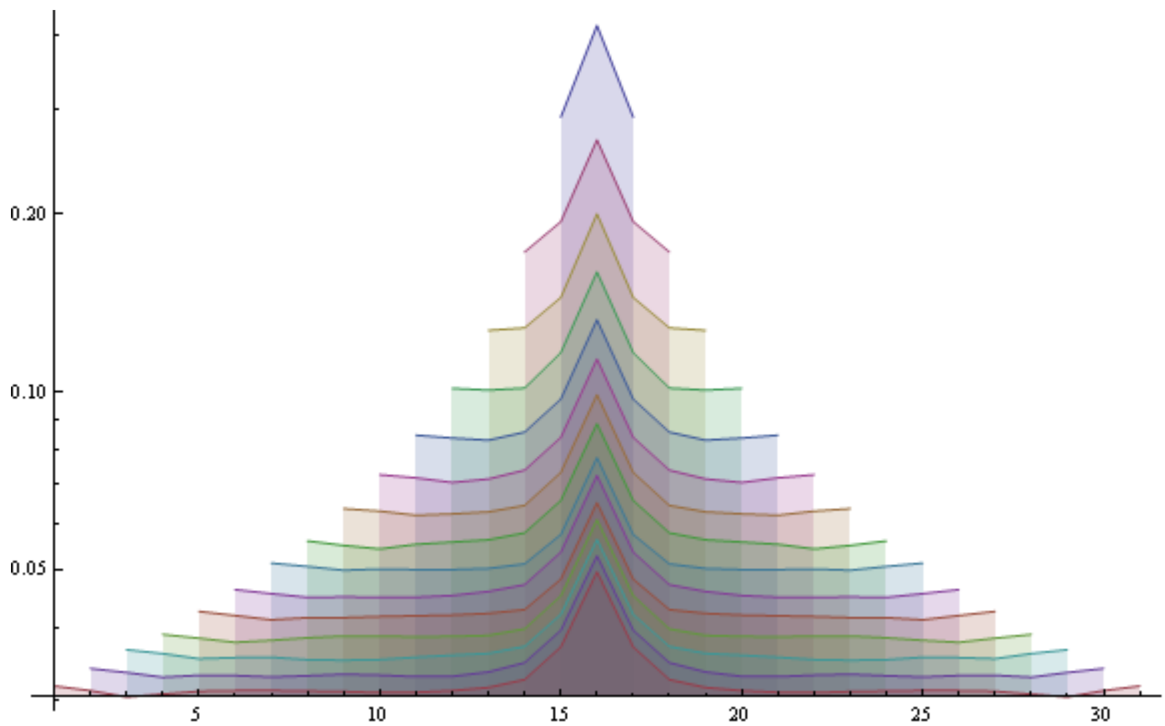


Figure 3-39: Cumulative probability for a qutrit walk on a cycle after 1000 time steps. T4 starting state, {3, 5, 7, 9, 11, 13, 15, 17, 19, 21, 23, 25, 27, 29, 31} nodes.

4 Conclusions

Qutrit walks using a Hadamard coin are unique. While both symmetric and asymmetric walks have been observed, the initial states differ from those of qubit walks.

On the line a starting state of two coins, with initial spins in opposite directions, produces an asymmetric qubit walk. This same superposition produces a symmetric qutrit walk. It is necessary to use two opposing coins, one of which has an imaginary coefficient, to produce a symmetric qubit walk. However, this state produces an asymmetric qutrit walk (Figure 3-4 and Figure 3-8).

Both qubit and qutrit walks (on the line or cycle) which start with a directionally biased state (S_1) produce an asymmetric walk (Figure 3-3 and Figure 3-5).

For walks on a cycle we take the cumulative distribution function.

On a cycle of odd numbered nodes qubit walks produce a flat cumulative probability distribution (Figure 3-35). This is the same behaviour has classical walks on cycles of any length. However, qutrit walks on cycles of odd length produce a cumulative probability distribution which has a significantly higher probably at the initial position (Figure 3-39).

Qubit cycles of even length fall into two categories. For cycles where $(n/4) \in \mathbb{Z}$ (where n is the number of nodes on the cycle) the starting and directly opposing node have approximately equal values after 1000 time steps[2] (see Figure 3-36). For cycles where $((n-2)/4) \in \mathbb{Z}$ the starting node has a probably $\sim$2.4 greater than the node opposite the starting node, after 1000 time steps[3] (see Figure 3-37).

All qutrit cycles of even length are similar to qubit cycles where$(n/4) \in \mathbb{Z}$, they have approximately equal probably at the starting node and opposite node (see Figure 3-38). Qutrit cycles of odd length have their highest probably at the starting node, after 1000 time steps.

In terms of the maximum values for the symmetric walks on the line, T_4 produces the lowest maximum value and T_7 the highest[4].

[2] $f(n,s) = {}^{n_{origin}}/_{n_{opposite}}$
$f(\{8,12,16,20,24,28,32\},1000) = \{0.9932, 1.002, 0.9790, 0.9862, 0.9926, 0.9880, 0.9815\}$

[3] $f(\{6,10,14,18,22,26,30\},1000) = \{2.328, 2.422, 2.377, 2.377, 2.421, 2.418, 2.347\}$
[4] g(b, s) is the maximum probability using state b after s steps
$g(\{S_2, S_4, S_6, S_7\}) = \{0.0191, 0.0163, 0.02, 0.0225\}$

5 Appendix

5.1 Mathematica module

```
BeginPackage["lqrw`"];
SingleLazyItteration::usage = "SingleLazyItteration[State, Coin] returns the\
state after it has been randomised by one iteration of a given three state coin."

CumulativeMean::usage = "CumulativeMean[List] returns a list where each \
value is the cumulative mean of all previous list values.";

QuantumStatesToPositionalProbabilites::useage =
"QuantumStatesToPositionalProbabilites[State]\
returns the position probabilities from a matrix of quantum states.";

ClassicalStatesToPositionalProbabilites::useage =
"ClassicalStatesToPositionalProbabilites[State]\
returns the position probabilities from a matrix of classical states.";

MultipleLazySteps::useage = "MultipleLazySteps[State, Coin, Steps] returns the state\
after a number of iterations using the Coin has been performed."

MultipleLazyStepsHistory::useage = "MultipleLazyStepsHistory is identical \
to MultipleLazySteps but returns all steps instead of just the final result"

MultipleLazyStepsRecursive::useage = "MultipleLazyStepsRecursive is functionally\
identical to MultipleLazySteps, but uses a recursive function."
Begin["`Private`"];

SingleLazyItteration[State0_,Coin0_] := Module[{State = State0, Coin = Coin0},
   oState = Normal[SparseArray[{Dimensions[State]->0}]];
   Steps  = Dimensions[State][[1]];

   For[i=1,i<=Steps,i++,
     oState[[i]]={
       State[[i]].Coin[[1]],
       State[[Mod[i-1,Steps,1]]].Coin[[2]],
       State[[Mod[i+1,Steps,1]]].Coin[[3]]
     };
   ];
   Return[oState];
]

QuantumStatesToPositionalProbabilites[State0_] := Module[{State=State0},
   Return[Simplify[Total[State*Conjugate[State],{2}]]]];

ClassicalStatesToPositionalProbabilites[State0_] := Module[{State=State0},
   Return[Simplify[Total[State,{2}]]]];

CumulativeMean[List0_] := Module[{List = List0},
   Return[Accumulate[List]/Range[Length[List]]
]];

MultipleLazyStepsRecursive[State0_,Coin0_,Steps0_] := Module[
{State = State0, Coin = Coin0, Steps = Steps0},
  Return[Simplify[
    Nest[
      (SingleLazyItteration[#,Coin])&,
      State,Steps]]]]

MultipleLazySteps[State0_,Coin0_,Steps0_] := Module[
{State = State0, Coin = Coin0, Steps = Steps0},
  inState = State;
  For[Step=1,Step<=Steps,Step++,
    outState = Simplify[
      SingleLazyItteration[inState, Coin]];
    inState = outState;];
  Return[outState];
]

MultipleLazyStepsHistory[State0_,Coin0_,Steps0_] := Module[
{State = State0, Coin = Coin0, Steps = Steps0},
  returnHistory = List[];
  inState = State;
  For[Step=1,Step<=Steps,Step++,
```

```
    outState = Simplify[
      SingleLazyItteration[inState, Coin]];
    AppendTo[returnHistory, outState];
    inState = outState;];
  Return[returnHistory];
]

End[];
EndPackage[];
```

5.2 Using the module

We used Wolfram Mathematica 7.0 and can only confirm that the library works for that version.

First we need to import the module. Directory slashes need to be doubled.

```
Get["lqrw.m", Path->{"F:\\MSc"}]
```

Next we set up our coins. In this example we are going to use a three state Hadamard coin.

```
BitOrder = 3;
σ = E^((2*Pi*I) / BitOrder);
HadamardThree = {
  {1, 1, 1},
  {1, σ^(BitOrder - 1), σ},
  {1, σ, σ^(BitOrder - 1)}
}/Sqrt[BitOrder];
```

This gives us our coin, H_3.

$$\mathrm{H}_3 = \frac{1}{\sqrt{3}}\begin{pmatrix} 1 & 1 & 1 \\ 1 & -\sqrt[3]{-1} & (-1)^{2/3} \\ 1 & (-1)^{2/3} & -\sqrt[3]{-1} \end{pmatrix}$$

We now create our initial state. In this example we are going to use equal amplitudes of each state (left, right and non-moving). However, the non-moving state will have a complex amplitude.

```
StartState = Normal[SparseArray[{
  {200, 3} -> 0,
  {100, 1} -> I/Sqrt[3],
  {100, 2} -> 1/Sqrt[3],
  {100, 3} -> 1/Sqrt[3]
}]];
```

This is our state matrix, for all spins and all positions. As we want to obtain the results for a 100 step line we create a 200 row matrix (100 steps left and 100 steps right) and place the starting state in the middle of this line. This is a 200 node cycle; however, if we perform fewer than 101 operations then no crossover will occur at the ends of the cycle.

A pared version of this starting state is show below.

$$\frac{1}{\sqrt{3}}\begin{pmatrix} 0 & 0 & 0 \\ \vdots & \vdots & \vdots \\ 0 & 0 & 0 \\ i & 1 & 1 \\ 0 & 0 & 0 \\ \vdots & \vdots & \vdots \\ 0 & 0 & 0 \end{pmatrix}$$

Next we perform the 100 transformations on our state with our chosen coin.

```
QuantumLazyLine100Steps =
  QuantumStatesToPositionalProbabilites /@
   MultipleLazyStepsHistory[StartState, HadamardThree, 100];
```

The `MultipleLazySteps` command takes three arguments, the start state, the coin and the number of transformation to perform. It returns a Matrix with the same dimensions as the start state matrix (in our example 200 × 3). The returned matrix is the state after *n* transformations.

The `MultipleLazyStepsHistory` command is an extension of the `MultipleLazySteps` command. It takes the same arguments, but returns a three dimensional matrix, with every transformation recorded. In our example it returns a 100 × 200 × 3 matrix.

The `QuantumStatesToPositionalProbabilites` command takes an $n \times m$ state matrix and returns the positional probability for each position in the form of an $n \times 1$ matrix.

In this example we want to show the positional probabilities at each time step, and thus we must map the `QuantumStatesToPositionalProbabilites` to the three dimensional matrix returned by `MultipleLazyStepsHistory`. This will return a 100 × 200 matrix of probabilities which we can plot.

This is the command we used to plot the results.

```
ListPlot3D[QuantumLazyLine100Steps,
 PlotRange -> {0, 0.2},
 ImageSize -> 666,
 ViewPoint -> {-.6, 2, .8},
 ColorFunction -> "CoffeeTones",
 Ticks -> {
  {{200, 100}, {150, 50}, {100, 0}, {50, -50}, {0, -100}},
   Automatic,
  Automatic},
 Axes -> True,
 AxesLabel -> {"Position", "Time Step", "Probability"},
 Mesh -> 9,
 MeshFunctions -> {Function[{x, y, z}, z]}]
```

The resulting graph is Figure 3-10: Qutrit walk, T7 starting state.

5.3 Graph showing all steps on a two state walk

On a two state walk odd numbered positions have zero valued probabilities on even numbered time steps. Likewise, even numbered positions have zero valued probabilities on odd numbered time steps. This is illustrated in Figure 2-2.

Mathemathica's graphing library produces graphs with many artefacts when using this dataset. An example can be seen below in Figure 5-1.

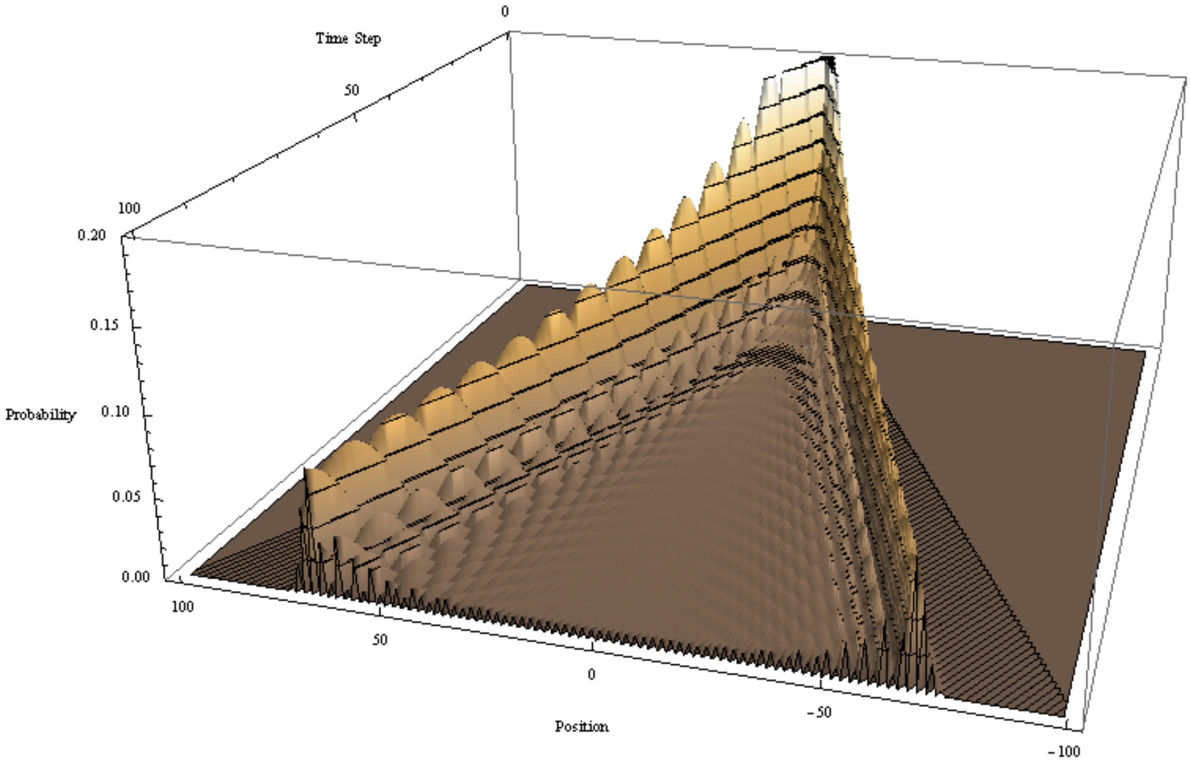


Figure 5-1: Qubit walk, B5 starting state. All time steps and positions shown

6 Bibliography

Aharonov, D., Ambainis, A., Kempe, J., & Vazirani, U. (2001). Quantum Walks on Graphs. *Proceedings on 33rd Annual ACM Symposium on Theory of Computing* (pp. 50-59). Heraklion, Crete, Greece: Association for Computing Machinery.

Brun, T. A., Carteret, H. A., & Ambainis, A. (2003). Quantum walks driven by many coins. *Phys. Rev. A , 67* (5), 052317-17.

Dolphin, D. F., & McGettrick, M. (2011). *Fairness in lazy quantum random walks*. (14th Workshop on Quantum Information Processing) Retrieved from http://www.skynet.ie/~tyrion/msc/david-dolphin_qip2011_poster.pdf

Grover, L. K. (1996). A Fast Quantum Mechanical Algorithm for Database Search. *Proceedings on 28th Annual ACM Symposium on Theory of Computing* (pp. 212-219). Philadephia, Pennsylvania, USA: Association for Computing Machinery.

Kempe, J. (2003). Quantum random walks - an introductory overview. *Contemporary Physics , 44* (4), 307-327.

Marttala, P. (2007). *An extension of the Deutsch-Jozsa algorithm to arbitrary qudits.* Saskatchewan: University of Saskatchewan.

McGettrick, M. (2010). One-Dimensional Quantum Walks with Memory. *Quantum Information & Computation , 10* (6), 509-524.

Shenvi, N., Kempe, J., & Whaley, K. B. (2003). Quantum random-walk search algorithm. *Physics Review A , 67* (5), 052307.

Yanofsky, N. S., & Mannucci, M. A. (2008). *Quantum Computing for Computer Scientists.* New York: Cambridge University Press.