Journal of Informatics and Mathematical Sciences
Vol. 10, Nos. 1 & 2, pp. 23–32, 2018
ISSN 0975-5748 (online); 0974-875X (print)
Published by RGN Publications
http://dx.doi.org/10.26713/jims.v10i1-2.685

http://www.rgnpublications.com

Research Article

MATLAB Programming to Implement Quantum Walk Algorithm for Presenting Probability Distributions of Quantum Walks

Lila Yuwana*, Agus Purwanto and Endarko

Department of Physics, Institut Teknologi Sepuluh Nopember, Surabaya, Indonesia
***Corresponding author:** lila@physics.its.ac.id

Abstract. There are numerous ideas that have been provided by quantum walks for new quantum algorithms. In this article, we surveyed the discrete quantum walk algorithm to present probability distributions. Moreover, we transformed the algorithm into MATLAB programming. Finally, the programming can be utilised to compare coin flip transformations that generate probability distributions as proposed in several previous articles and also to identify a unitarity of a coin flip transformation.

Keywords. Quantum Walks; Probability distributions; Coin flip transformation

MSC. 81P16; 81P45; 81P68

Received: September 19, 2017 **Accepted:** November 23, 2017

1. Introduction

An efficiency is the key to sustain quantum information research. The consequence of that is from a few decades, researchers have been pursuing cutting-edge methods in order to solve hard problems.

One of interesting quantum algorithms is quantum walks algorithm. Quantum walks were coined by Aharonov in 1993 as quantum random walk that was utilized in a quantum-optics application [1]. In 2003, Kempe [6] revealed that quantum walks have essential role in quantum computation, hence will be deployed to provision breakthrough and fast algorithms. Therefore, those algorithms could be run on a quantum computer. This article also showed valuable information of quantum walks introduction, the contrast differences to classical walks, and developments of quantum walks in quantum information science. Inspired by tremendous achievements of random walks and Markov chain methods (quantum walk's counterpart in classical walks) in the development of classical algorithms [8], Ambainis et al. presented quantum walks on interesting graphs by plotting approximately probability distributions after deployed Fourier analysis of Hadamard walk numerically [2]. Excitingly, a main role of quantum computation, that are able to carry out problems faster than classical computer, was proved by Childs et al. [5]. They have successfully solved a hard problem exponentially faster on a quantum computer than on a classical computer. Furthermore, research in quantum walksare still continue, e.g. the article delivered by Dheeraj et al. that presented alternative approach by defining two different ways: discrete-time and continuous-time quantum walks (DTQWs and CTQWs) [9]. The other important work was delivered by Ambainis [3] who explained about algorithmic application to plot probability distributions of quantum walk. More detail about the algorithmic application will be shown in the next section. Recently, Montero offered versatile utility to construct both quantum and random walks in 2017 using mechanical quantum approximation [7]. However, methods of Ambainis to obtain probability distributions [3] is fully expected to be converted into progammable algorithm to achieve probability distributions as well as analytical results. This article aimed to generate MATLAB programming based on the quantum walks algorithmic application. Furthermore, the coin flip operator is changeable, as a result, one can compare probability distributions of previous articles that using various coin flip operators. In addition, the programming comprises both numerical and visual results.

The following sections of this article are arranged as follows. In Section 2, we deliver rudimentary calculations on quantum walks that embrace iterations in each step. Section 3 exhibits the algorithm of calculations of quantum walks, then the algorithm is converted into MATLAB programming to solve quantum walks problems rapidly. Finally, the conclusion of capability of MATLAB programming proposed in this article and the comparison with the previous method is explained obviously in Section 4.

2. Discrete Quantum Walks Calculation

Suppose a quantum process which has initial basis state $|n\rangle$, $n \in Z$. A unitary transformation leads to [3]

$$|n\rangle \to a\,|n-1\rangle + b\,|n\rangle + c|n+1\rangle\,. \tag{1}$$

Equation (1) indicates the position moves left, moves right, or holds from the current state with probability $|a|^2$, $|c|^2$ or $|b|^2$, respectively.

By utilizing a "coin" state, the position can be evaluated the position after m steps. There are two operations applied at each step:

(1) A coin flip transformation or C

$$C\,|n,0\rangle = a\,|n,0\rangle + b\,|n,1\rangle\,, \quad C\,|n,1\rangle = c\,|n,0\rangle + d\,|n,1\rangle\,. \tag{2}$$

(2) Shift or S

$$S|n,0\rangle = |n-1,0\rangle\,, \quad S|n,1\rangle = |n+1,1\rangle\,. \tag{3}$$

Operator C is a unitary matrix. Consider if C is Hadamard operator

$$\begin{pmatrix} a & b \\ c & d \end{pmatrix} = \begin{pmatrix} \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} & -\frac{1}{\sqrt{2}} \end{pmatrix}. \tag{4}$$

Substituting eq. (4) to (2) yields

$$C|n,0\rangle = \frac{1}{\sqrt{2}}|n,0\rangle + \frac{1}{\sqrt{2}}|n,1\rangle\,, \quad C|n,1\rangle = \frac{1}{\sqrt{2}}|n,0\rangle - \frac{1}{\sqrt{2}}|n,1\rangle\,. \tag{5}$$

Eq. (5) shows probabilities of each state after Hadamard transformation is $\frac{1}{2}$. It is easy to understand classically that a state after applying C operation has the same probability to move left or right. Surprisingly, the results are not similar as our prediction. The derivations below are first four steps of quantum walks after performing two operation, S and C, at initial state $|0,0\rangle$ (eq. (3) and (5) are involved in this calculation).

# 1st step: $SC|0,0\rangle = S\left[\frac{1}{\sqrt{2}}|0,0\rangle + \frac{1}{\sqrt{2}}|0,1\rangle\right] = \frac{1}{\sqrt{2}}|-1,0\rangle + \frac{1}{\sqrt{2}}\,|1,1\rangle\,.$

# 2nd step: $SC\left[\frac{1}{\sqrt{2}}|-1,0\rangle + \frac{1}{\sqrt{2}}|1,1\rangle\right] = \frac{1}{2}|-2,0\rangle + \frac{1}{2}|0,1\rangle + \frac{1}{2}|0,0\rangle - \frac{1}{2}|2,1\rangle\,.$

# 3rd step: $SC\left[\frac{1}{2}|-2,0\rangle + \frac{1}{2}|0,1\rangle + \frac{1}{2}|0,0\rangle - \frac{1}{2}|2,1\rangle\right]$

$$= \underbrace{\begin{pmatrix} \frac{1}{2\sqrt{2}}|-3,0\rangle + \frac{1}{2\sqrt{2}}|-1,1\rangle + \frac{1}{2\sqrt{2}}|-1,0\rangle - \frac{1}{2\sqrt{2}}|1,1\rangle \\ +\frac{1}{2\sqrt{2}}|-1,0\rangle + \frac{1}{2\sqrt{2}}|1,1\rangle - \frac{1}{2\sqrt{2}}|1,0\rangle + \frac{1}{2\sqrt{2}}|3,1\rangle \end{pmatrix}}_{2^3=8}$$

# 4th step: $SC\Big[\frac{1}{2\sqrt{2}}|-3,0\rangle + \frac{1}{2\sqrt{2}}|-1,1\rangle + \frac{1}{2\sqrt{2}}|-1,0\rangle - \frac{1}{2\sqrt{2}}|1,1\rangle$

$$+\frac{1}{2\sqrt{2}}|-1,0\rangle + \frac{1}{2\sqrt{2}}|1,1\rangle - \frac{1}{2\sqrt{2}}|1,0\rangle + \frac{1}{2\sqrt{2}}|3,1\rangle\Big]$$

$$= \underbrace{\frac{1}{4}|-4,0\rangle + \cdots + -\frac{1}{4}|4,1\rangle}_{2^4=16}$$

$$= \underbrace{\frac{1}{4}|-4,0\rangle + \frac{3}{4}|-2,0\rangle + \frac{1}{4}|-2,1\rangle - \frac{1}{4}|0,0\rangle + \frac{1}{4}|0,1\rangle + \frac{1}{4}|2,0\rangle - \frac{1}{4}|2,1\rangle - \frac{1}{4}|4,1\rangle}_{8}\,. \tag{6}$$

By performing similar calculation, we can obtain next steps. The first two steps show that the amplitudes of states are the same. Meanwhile, at third step and after, the amplitudes are not uniform. This is because there is quantum interference among the same states. In other words, the amplitudes of the same states can be doubled or vanish. That is why, e.g. at fourth step, the number of terms can be reduced from 16 terms to 8 terms. The probabilities of each state can be easily examined by calculating $|a_i|^2$, where a_i is the amplitude of each state. Finally, the terms can be attenuated again by grouping the states for the same position (see part (c) in the next section).

3. Developing MATLAB Programming

Considering involving immense states when we examine quantum walk after next steps, it will be helpful to employ numerical programming to support calculations. Firstly, we show the main stages of the entire programming (Figure 1).

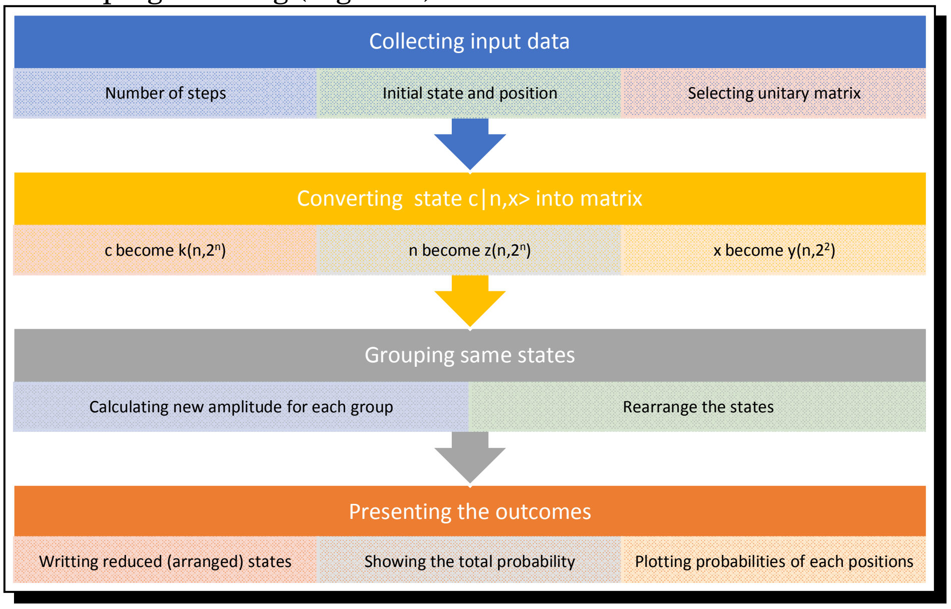

Collecting input data
Number of steps
Initial state and position
Selecting unitary matrix
Converting state c|n,x> into matrix
c become k(n,2ⁿ)
n become z(n,2ⁿ)
x become y(n,2²)
Grouping same states
Calculating new amplitude for each group
Rearrange the states
Presenting the outcomes
Writting reduced (arranged) states
Showing the total probability
Plotting probabilities of each positions

Figure 1. Diagram block of the entire programming

Secondly, we reveal the MATLAB programming based on each stage in Figure 1 and also screen shots of compiled programming.

(a) Collecting input data

```
disp('==========================================');
disp('          SC - Transformation          ');
disp('==========================================');
n=input('Number of Steps: ');
s1=input('Initial State (0 or 1) :');
p1=input('Initial Position (x) :');
a=1/sqrt(2); b=1/sqrt(2);
c=1/sqrt(2); d=-1/sqrt(2);
fprintf('\nInitial State and Position: |%1.0f,%1.0f>\n',p1,s1);
fprintf('\n');
```

Output on the screen:

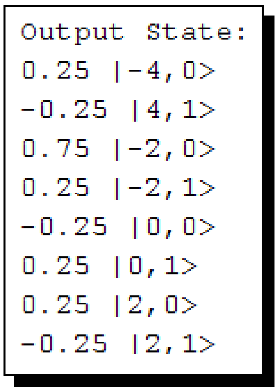
```
==========================================
          SC - Transformation
==========================================
Number of Steps: 4
Initial State (0 or 1) :0
Initial Position (x) :0

Initial State and Position: |0,0>
```

(b) Converting state $c|n,x\rangle$ into matrix and grouping for the same states

```
for r=1:n
    for s=1:2^r
        if(mod(s,2)==0)
            y(r,s)=1;
        elsey(r,s)=0;
        end
        if((r==1))
            if(s1==0)
                if(s==1)
                    k(r,s)=a;
                elsek(r,s)=b;end
            elseif(s==1)
                    k(r,s)=c;
elsek(r,s)=d;endend
        elset=t+1;
if(t==1)
                k(r,s)=a*k(r-1,round(s/2));
                else if(t==2)
                    k(r,s)=b*k(r-1,round(s/2));
                        else if(t==3)
                            k(r,s)=c*k(r-1,round(s/2));
                                else if(t==4)
                                    k(r,s)=d*k(r-1,round(s/2));
                                    endendendend
            if(t==4)
                t=0;endend
    if(r==1)
        if(y(r,s)==0)
            z(r,s)=p1-1;
        elsez(r,s)=p1+1;end
    elseif(y(r,s)==0)
            z(r,s)=z(r-1,round(s/2))-1;
        elsez(r,s)=z(r-1,round(s/2))+1;
        endendendend
[m, bin] = histc(z(n,:), unique(z(n,:)));
multiplez=find(m>1);
indexz=find(ismember(bin,multiplez));
[bz,nz]=size(indexz);
[bz,mz]=size(multiplez);
rz=2;nz;zz(n,1)=z(n,1);zz(n,2)=z(n,2^n);
yy(n,1)=y(n,1);yy(n,2)=y(n,2^n);
kk(n,1)=k(n,1);kk(n,2)=k(n,2^n);rz1=rz; kz=0;
for r=2:2^n-1
 rz1=rz1+1;zz(n,rz1)=z(n,r);
yy(n,rz1)=y(n,r);kk(n,rz1)=k(n,r);
end
rzz=rz; pk=0; qk=0;
for i=1:mz
    indexz1=find(ismember(bin,multiplez(i)));
    kkz(i)=z(n,indexz1(i));
end
for i=1:mz
    rzz=rzz+1;
    for j=(rz+1):2^n
        if(yy(n,j))==0
            if(zz(n,j))==kkz(i)
                pk=pk+kk(n,j);end
        else if(yy(n,j))==1
                if(zz(n,j))==kkz(i)
                    qk=qk+kk(n,j);
                endendendend
    zz(n,rzz)=kkz(i);yy(n,rzz)=0;
    kk(n,rzz)=pk;zz(n,rzz+1)=kkz(i);
    yy(n,rzz+1)=1;kk(n,rzz+1)=qk;
    pk=0; qk=0;rzz=rzz+1;
end
fprintf('Output State: \n');
for s=1:rzz
    fprintf('%s |%1.0f,%1.0f>  \n',num2str(kk(n,s)),zz(n,s),yy(n,s));
    k2(n,s)=(abs(kk(n,s)))^2;
    p=p+k2(n,s);
end
```

Output on the screen:



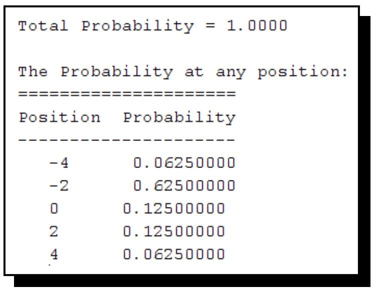
```
Output State:
0.25 |-4,0>
-0.25 |4,1>
0.75 |-2,0>
0.25 |-2,1>
-0.25 |0,0>
0.25 |0,1>
0.25 |2,0>
-0.25 |2,1>
```

These parts have a main role in the programming, because in this stage enormous number of states can be reduced in much simple form. In this case, the desired number of step is 4, hence the total states involved in calculations are 2^4 terms or 16 terms. After attempting to group for the same states and recalculating each amplitude, 16 terms become 8 terms.

(c) Presenting the outcomes

```
fprintf('\nTotal Probability = %1.4f\n',p);
kkz=rz;nrzz=(rzz-2)/2+2;pzz(n,1)=zz(n,1);
pzz(n,nrzz)=zz(n,2);pk2(n,1)=abs(kk(n,1))^2;
pk2(n,nrzz)=abs(kk(n,2))^2;
for s=2:nrzz-1
    kkz=kkz+1;pzz(n,s)=zz(n,kkz);
    pk2(n,s)=abs(kk(n,kkz))^2+abs(kk(n,kkz+1))^2;
    kkz=kkz+1;end
fprintf('\nThe Probability at any position: \n');
fprintf('===================== \n');
fprintf('Position  Probability\n');
fprintf('--------------------- \n');
for s=1:nrzz
    fprintf('   %1.0f      %1.8f\n',pzz(n,s),pk2(n,s));end
plot(pzz(n,1:nrzz),pk2(n,1:nrzz))
```

Output on the screen:



```
Total Probability = 1.0000

The Probability at any position:
=====================
Position  Probability
---------------------
   -4       0.06250000
   -2       0.62500000
   0       0.12500000
   2       0.12500000
   4       0.06250000
```

Finally, the states can be reduced again to five (m steps $+1$) terms, because each position involve both state 0 and 1. The next figure is obtained from the five even positions and the probabilities is zero for odd position.

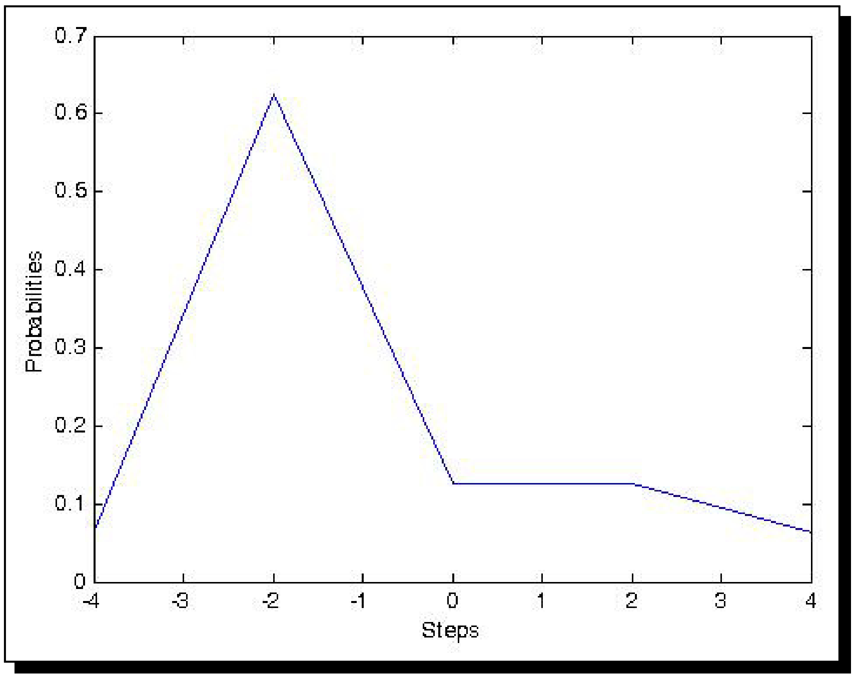


Figure 2. The probabilities of each states in any positions after 4 steps

In addition, if the total probability is not 1, it indicates that the selected matrix C is not unitary, then choose other matrix which is really unitary. The probability of each states for four steps generated by the programming has the same result as the previous manual calculation (eq. (6)). Figure 2 represents the probabilities in each states and positions. The outcomes for 24 steps ($n = 24$) is depicted in Figure 3.

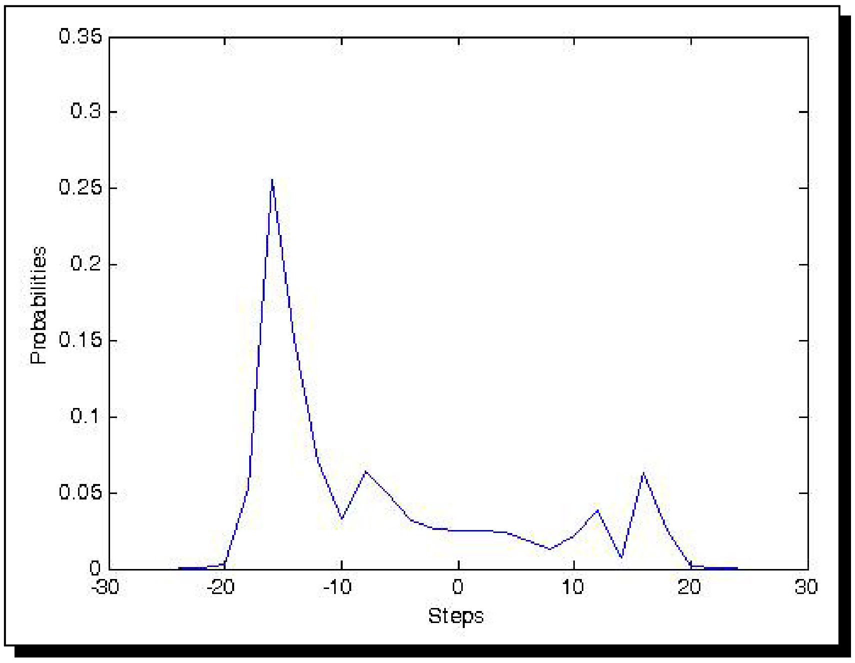


Figure 3. The probabilities of each states in any positions after 24 steps

This result is very exciting because the number of state reduction is very effective, from 2^{20} terms or 16,777,216 terms become 25 terms. The result is also similar as Ambainis's article in 2008 [3] that has showed analytical calculation to result probability distributions using Hadamard operator as a coin flip transformation. Moreover, there are several articles using numerical approximation to produce similar graphics of probability distributions [6, 2, 4]. However, we prefer to utilise a programming based on analytical calculation for constructing probability distributions of quantum walks. This is because the results both manual calculation and the programming are the same.

On the other side, there are different results when other unitary operator used for coin flip transformation. Nayak and Vishwanath [4], Kempe [6], and Ambainis [3] proposed unitary operator

$$C' = \begin{pmatrix} \frac{1}{\sqrt{2}} & \frac{i}{\sqrt{2}} \\ \frac{i}{\sqrt{2}} & \frac{1}{\sqrt{2}} \end{pmatrix} \tag{7}$$

to produce symmetrical probability distributions. Surprisingly, after we had modified our programme to apply that operator, the probability distribution is still asymmetric as depicted in Figure 4. Next, we calculated manually to reveal behaviour of implementation of operator C' on probability distribution in four steps.

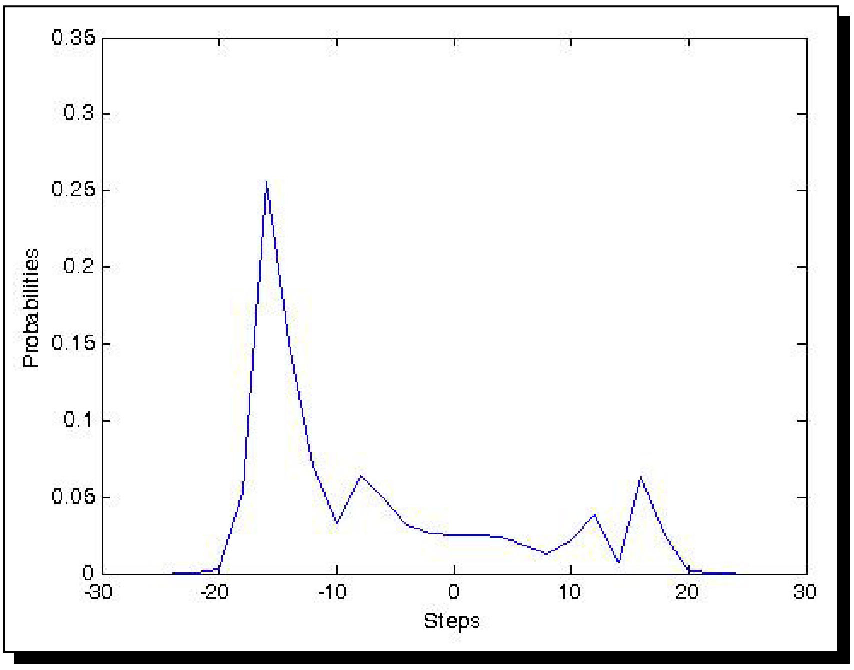


Figure 4. The probabilities of each states in any positions after 24 steps using C' as a coin flip operator

According to Figure (4), we can verify the result by calculating analytically by applying method in eq. (4), and the result after four steps is

$$\frac{1}{4}|-4,0\rangle - \frac{3}{4}|-2,0\rangle + \frac{1}{4}i|-2,1\rangle - \frac{1}{4}|0,0\rangle - \frac{1}{4}i|0,1\rangle - \frac{1}{4}|2,0\rangle - \frac{1}{4}i|2,1\rangle + \frac{1}{4}i|4,1\rangle \,.$$

By calculating all of amplitude and combining state 0 and 1 for the same position, we have probabilities for each positions (Table 1).

Table 1. Probabilities for each positions based on manual calculation for 4 steps

Position	Probability
-4	$\frac{1}{16}$
-2	$\frac{10}{16}$
0	$\frac{2}{16}$
2	$\frac{2}{16}$
4	$\frac{1}{16}$

It is obvious that probabilities in Table 1 presents asymmetric distribution that is different as previous articles that have been proposed [6, 3, 4]. Finally, we firmly believe that the programming proposed in this article is reliable to be utilised because of the equivalence of manual calculations and programming results.

4. Conclusion

In this article we have examine quantum walks obtained by applying Hadamard transformation and shift of position and convert the algorithmic application into MATLAB programming. The results provide numerical values and a graph represents probabilities in each states and positions. In addition, the programming also can identify whether a selected matrix is unitary or not. However, there are different results of probability distributions from previous articles when we use eq. (7) as a coin flip operator instead of Hadamard operator. The strong good point is that the programming is able to simplify significantly enormous number of terms of calculation.

Competing Interests

The authors declare that they have no competing interests.

Authors' Contributions

All the authors contributed significantly in writing this article. The authors read and approved the final manuscript.

References

[1] Y. Aharonov, L. Davidovich and N. Zagury, *Quantum random walks, Physical Review A* **48** (1993), 1687 – 1690.

[2] A. Ambainis, E. Bach, A. Nayak, A. Vishwanath and J. Watrous, One-dimensional quantum walks, in *The Thirty-Third Annual ACM Symposium on Theory of Computing* (STOC'01), New York (2001).

[3] A. Ambainis, Quantum walks and their algorithmic applications, *International Journal of Quantum Information* **10** (4) (2003), 507.

[4] N. Ashwin and V. Ashvin, *Quantum Walk on the Line*, Center for Discrete Mathematics & Theoretical Computer Science, New York (2000).

[5] A.M. Childs, R. Cleve, E. Deotto, E. Farhi, S. Guttman and D.A. Spielman, Exponential algorithmic speedup by quantum walk, in *Proceedings 35th ACM Symposium on Theory of Computing* (STOC 2003), San Diego (2003).

[6] J. Kempe, Quantum random walks: An introductory overview, *Contemporary Physics* **44** (2003), 307 – 327.

[7] M. Montero, Quantum and random walks as universal generators of probability distributions, *Physical Review A* **95** (2017), 062326; 062326-1.

[8] R. Motwani and P. Raghavan, *Randomized Algorithms*, 1st edition, Cambridge University Press (1995), New York.

[9] M.N. Dheeraj and A.T. Brun, Continuous limit of discrete quantum walks, *Physical Review A* **91** (6) (2015), 062304-1.