Quantum Implementation of Random Walks

June 5, 2025

0.1 5x5x5 Case

[1]:
```
import numpy as np
import pretty_midi
import random

# 1. Musical elements
notes = ['F4', 'G4', 'A4', 'B4', 'C5']  # 5 notes
durations = [0.25, 0.5, 0.75, 1.0, 1.5] # quarter, half, dotted half, whole,␣
 ↪double (in beats)
intensities = [40, 60, 80, 100, 120]    # MIDI velocities (pp, p, mp, mf, f)

# 2. All possible tuples (vertices)
vertex_tuples = []
for n in notes:
    for d in durations:
        for i in intensities:
            vertex_tuples.append((n, d, i))

num_vertices = len(vertex_tuples)  # 125

# 3. Generate a transition matrix with at least 2 outgoing edges per vertex
def random_transition_matrix(n, min_edges=2, seed=42):
    np.random.seed(seed)
    T = np.zeros((n, n))
    for i in range(n):
        # Choose at least min_edges targets for outgoing edges
        targets = np.random.choice(n, min_edges, replace=False)
        probs = np.random.rand(min_edges)
        probs /= probs.sum()
        T[i, targets] = probs
        # To make the matrix fully stochastic, sprinkle small probability␣
 ↪everywhere
        extra = np.random.rand(n) * 1e-2
        T[i] += extra
        T[i] /= T[i].sum()
    return T
```

```python
# 4. Random walk function
def random_walk_tuple(T, init_vertex, N):
    seq = []
    current = init_vertex
    for _ in range(N):
        seq.append(current)
        probs = T[current]
        current = np.random.choice(np.arange(len(T)), p=probs)
    return seq

# 5. Utility to convert note name to MIDI number
def note_name_to_midi(note_name):
    return pretty_midi.note_name_to_number(note_name)

# 6. Build two different transition matrices and initial states (for two␣
 ↪musicians)
T1 = random_transition_matrix(num_vertices, min_edges=2, seed=2024)
T2 = random_transition_matrix(num_vertices, min_edges=2, seed=42)

init1 = 0       # e.g., start at ('F4', 0.25, 40)
init2 = 124     # e.g., start at ('C5', 1.5, 120)

N_steps = 50

# 7. Generate sequences
seq1_indices = random_walk_tuple(T1, init1, N_steps)
seq2_indices = random_walk_tuple(T2, init2, N_steps)

seq1 = [vertex_tuples[i] for i in seq1_indices]
seq2 = [vertex_tuples[i] for i in seq2_indices]

# 8. Generate MIDI file
def make_duet_midi(seq1, seq2, filename="5x5x5 case.mid", tempo=80):
    midi = pretty_midi.PrettyMIDI()
    inst1 = pretty_midi.Instrument(program=0, name='Musician1')
    inst2 = pretty_midi.Instrument(program=12, name='Musician2')  # Different␣
 ↪instrument

    # Musician 1
    t = 0
    for note_name, dur, vel in seq1:
        pitch = note_name_to_midi(note_name)
        note = pretty_midi.Note(velocity=int(vel), pitch=pitch, start=t,␣
 ↪end=t+dur)
        inst1.notes.append(note)
        t += dur
```

```
    # Musician 2
    t = 0
    for note_name, dur, vel in seq2:
        pitch = note_name_to_midi(note_name)
        note = pretty_midi.Note(velocity=int(vel), pitch=pitch, start=t,␣
 ↪end=t+dur)
        inst2.notes.append(note)
        t += dur

    midi.instruments.append(inst1)
    midi.instruments.append(inst2)
    midi.write(filename)
    print(f"Saved duet MIDI as: {filename}")

make_duet_midi(seq1, seq2, filename="5x5x5 case.mid")
```

```
Saved duet MIDI as: 5x5x5 case.mid
```

[ ]:

0.2 3x3x3 Case

[2]:
```
import numpy as np
import pretty_midi

# 1. Define states for 3 notes, 3 durations, 3 intensities
notes = ['F4', 'G4', 'A4']
note_pitches = [65, 67, 69]  # MIDI numbers for F4, G4, A4
durations = [0.25, 0.5, 1.0]  # in quarter note lengths
intensities = [40, 70, 100]   # MIDI velocities: soft, medium, loud

# Build the 27 tuples and lookup dictionaries
tuple_labels = []
for i in range(3):
    for j in range(3):
        for k in range(3):
            tuple_labels.append( (note_pitches[i], durations[j],␣
 ↪intensities[k]) )

n_states = 27

# 2. Build a random row-stochastic 27x27 transition matrix
def random_stochastic_matrix(n, min_out=2, seed=42):
    np.random.seed(seed)
    T = np.zeros((n, n))
    for i in range(n):
        # Each state has at least min_out outgoing transitions
```

```
        out = np.random.choice(n, min_out, replace=False)
        probs = np.random.rand(min_out)
        probs /= probs.sum()
        T[i, out] = probs
        # Optionally add some noise to other entries
        noise = np.random.rand(n) * 0.01
        T[i] += noise
        T[i] /= T[i].sum()
    return T

T_tuple3 = random_stochastic_matrix(n_states, min_out=2)

# 3. Simulate the random walk
def random_walk(T, tuple_labels, N=32, init_state=0):
    seq = []
    current = init_state
    for _ in range(N):
        note, dur, vel = tuple_labels[current]
        seq.append( (note, dur, vel) )
        p = T[current]
        current = np.random.choice(len(T), p=p)
    return seq

walk_seq = random_walk(T_tuple3, tuple_labels, N=32, init_state=0)

# 4. Convert sequence to MIDI file
def sequence_to_midi(seq, filename="3x3x3 case.mid", tempo=120):
    midi = pretty_midi.PrettyMIDI()
    instrument = pretty_midi.Instrument(program=0)
    time = 0.0
    qtr_sec = 60.0 / tempo
    for note, dur, vel in seq:
        note_obj = pretty_midi.Note(velocity=int(vel), pitch=int(note),␣
 ↪start=time, end=time+dur*qtr_sec)
        instrument.notes.append(note_obj)
        time += dur * qtr_sec
    midi.instruments.append(instrument)
    midi.write(filename)
    print(f"MIDI saved to {filename}")

sequence_to_midi(walk_seq, filename="3x3x3 case.mid")
```

```
MIDI saved to 3x3x3 case.mid
```

[ ]:

[ ]: