Classical Implementation of Random Walks

June 5, 2025

[2]:

```python
import numpy as np
from midiutil import MIDIFile
import random
from io import BytesIO
```

0.0.1 Approach 1

[3]:

```python
# 1. Define musical vocabularies and mapping functions

notes = ['F4', 'G4', 'A4', 'B4', 'C5']
note_to_midi = {'F4': 65, 'G4': 67, 'A4': 69, 'B4': 71, 'C5': 72}

durations = [1/6, 1/4, 1/3, 1/2, 3/4]

intensities = ['pp', 'p', 'mp', 'mf', 'f']
intensity_to_velocity = {'pp': 40, 'p': 55, 'mp': 70, 'mf': 85, 'f': 100}

# 2. Example: Transition matrices for each musician
def random_transition_matrix(size):
    M = np.random.rand(size, size)
    return (M.T / M.sum(axis=1)).T

# Musician 1 prefers starting at F4, 1/4, pp
T_note1 = random_transition_matrix(5)
T_dur1  = random_transition_matrix(5)
T_int1  = random_transition_matrix(5)
init1 = (0, 1, 0)  # F4, 1/4, pp

# Musician 2 prefers starting at C5, 3/4, mf
T_note2 = random_transition_matrix(5)
T_dur2  = random_transition_matrix(5)
T_int2  = random_transition_matrix(5)
init2 = (4, 4, 3)  # C5, 3/4, mf

def random_walk(N, T_note, T_dur, T_int, init):
    idx_note, idx_dur, idx_int = init
    seq_notes = []
    seq_durs = []
    seq_ints = []
    for _ in range(N):
        seq_notes.append(idx_note)
        seq_durs.append(idx_dur)
        seq_ints.append(idx_int)
        idx_note = np.random.choice(5, p=T_note[idx_note])
        idx_dur  = np.random.choice(5, p=T_dur[idx_dur])
        idx_int  = np.random.choice(5, p=T_int[idx_int])
    return seq_notes, seq_durs, seq_ints

# 3. Generate random walks
N_steps = 50
part1 = random_walk(N_steps, T_note1, T_dur1, T_int1, init1)
part2 = random_walk(N_steps, T_note2, T_dur2, T_int2, init2)

# 4. Convert walks to MIDI-ready values
def to_midi_triplets(seq, notes, durations, intensities):
    midi_notes = [note_to_midi[notes[n]] for n in seq[0]]
    midi_durs  = [durations[d] * 4 for d in seq[1]]  # Convert to quarter note units (beats)
    midi_vels  = [intensity_to_velocity[intensities[i]] for i in seq[2]]
    return list(zip(midi_notes, midi_durs, midi_vels))

midi_part1 = to_midi_triplets(part1, notes, durations, intensities)
midi_part2 = to_midi_triplets(part2, notes, durations, intensities)

# 5. Create the MIDI duet file
def create_duet_midi(melody1, melody2, tempo=80, filename="Duet.mid"):
    midi = MIDIFile(2)  # Two tracks
    midi.addTempo(0, 0, tempo)
    midi.addTempo(1, 0, tempo)
    midi.addProgramChange(0, 0, 0, 1)  # Acoustic Grand Piano
    midi.addProgramChange(1, 1, 0, 41)  # Violin

    time1 = 0
    for note, dur, vel in melody1:
        midi.addNote(0, 0, note, time1, dur, vel)
        time1 += dur

    time2 = 0
    for note, dur, vel in melody2:
        midi.addNote(1, 1, note, time2, dur, vel)
        time2 += dur

    with open(filename, "wb") as output:
        midi.writeFile(output)
    print(f"MIDI file saved as {filename}")
```

```
# 6. Produce the music
create_duet_midi(midi_part1, midi_part2, tempo=80, filename="Approach 1.mid")
```

```
MIDI file saved as Approach 1.mid
```

[ ]:

```

```

0.0.2 Approach 2

[6]:

```
import numpy as np
from mido import Message, MidiFile, MidiTrack, MetaMessage

# --------- Parameters & Mappings ---------
notes = ['F4', 'G4', 'A4', 'B4', 'C5']
durations = [1/6, 1/4, 1/3, 1/2, 3/4]
intensities = ['pp', 'p', 'mp', 'mf', 'f']

# MIDI note numbers for mapping (C4 = 60)
note_to_midi = {'F4': 65, 'G4': 67, 'A4': 69, 'B4': 71, 'C5': 72}

# MIDI velocity mapping (dynamics)
intensity_to_vel = {'pp': 30, 'p': 45, 'mp': 60, 'mf': 80, 'f': 100}

# --------- Random Walk Functions ---------
def index_to_tuple(idx):
    note_idx = idx // 25
    dur_idx = (idx % 25) // 5
    int_idx = idx % 5
    return (notes[note_idx], durations[dur_idx], intensities[int_idx])

def tuple_to_midi(tup):
    n, d, i = tup
    return note_to_midi[n], d, intensity_to_vel[i]

def random_transition_matrix(size, min_out=2, seed=None):
    rng = np.random.default_rng(seed)
    T = np.zeros((size, size))
    for i in range(size):
        outgoing = rng.choice(size, min_out, replace=False)
        probs = rng.random(min_out)
        probs /= probs.sum()
        T[i, outgoing] = probs
    T /= T.sum(axis=1, keepdims=True)
    return T

def random_walk_tuple_graph(N, T, init_idx):
    seq = []
    idx = init_idx
    for _ in range(N):
        probs = T[idx]
        next_idx = np.random.choice(len(probs), p=probs)
        seq.append(index_to_tuple(next_idx))
        idx = next_idx
    return seq

# --------- Generate Sequences for Two Musicians ---------
N = 150  # Number of notes for each musician
size = 125

T1 = random_transition_matrix(size, min_out=2, seed=42)
T2 = random_transition_matrix(size, min_out=2, seed=2024)

init_vertex1 = 0     # ('F4', 1/6, 'pp')
init_vertex2 = 124   # ('C5', 3/4, 'f')

seq1 = random_walk_tuple_graph(N, T1, init_vertex1)
seq2 = random_walk_tuple_graph(N, T2, init_vertex2)

# --------- MIDI Generation ---------
def write_duet_midi(seq1, seq2, filename='RandomWalk_Duet.mid', tempo_bpm=80):
    mid = MidiFile()
    track1 = MidiTrack()
    track2 = MidiTrack()
    mid.tracks.append(track1)
    mid.tracks.append(track2)

    # Tempo (in microseconds per beat)
    tempo = int(60_000_000 / tempo_bpm)
    track1.append(MetaMessage('set_tempo', tempo=tempo))
    track2.append(MetaMessage('set_tempo', tempo=tempo))

    ticks_per_beat = mid.ticks_per_beat

    # Part 1
    time1 = 0
    for n, d, v in seq1:
        note, dur, vel = tuple_to_midi((n, d, v))
        duration_ticks = int(dur * ticks_per_beat)
        track1.append(Message('note_on', note=note, velocity=vel, time=time1))
        track1.append(Message('note_off', note=note, velocity=vel,␣
 ↪time=duration_ticks))
        time1 = 0  # Time accumulates only at note_on; set to 0 for subsequent␣
 ↪notes
```

```
    # Part 2 (put on channel 1 for a second instrument)
    time2 = 0
    for n, d, v in seq2:
        note, dur, vel = tuple_to_midi((n, d, v))
        duration_ticks = int(dur * ticks_per_beat)
        track2.append(Message('note_on', note=note, velocity=vel, time=time2,␣
 ↪channel=1))
        track2.append(Message('note_off', note=note, velocity=vel,␣
 ↪time=duration_ticks, channel=1))
        time2 = 0

    mid.save(filename)
    print(f"MIDI file saved as {filename}")

# --------- Generate the MIDI File ---------
write_duet_midi(seq1, seq2, filename='Approach 2.mid', tempo_bpm=80)
```

```
MIDI file saved as Approach 2.mid
```

[ ]:

[ ]: