YouTube video explaining the analytical solution

POWERPOINT PRESENTATION DOWNLOAD LINK

Problem

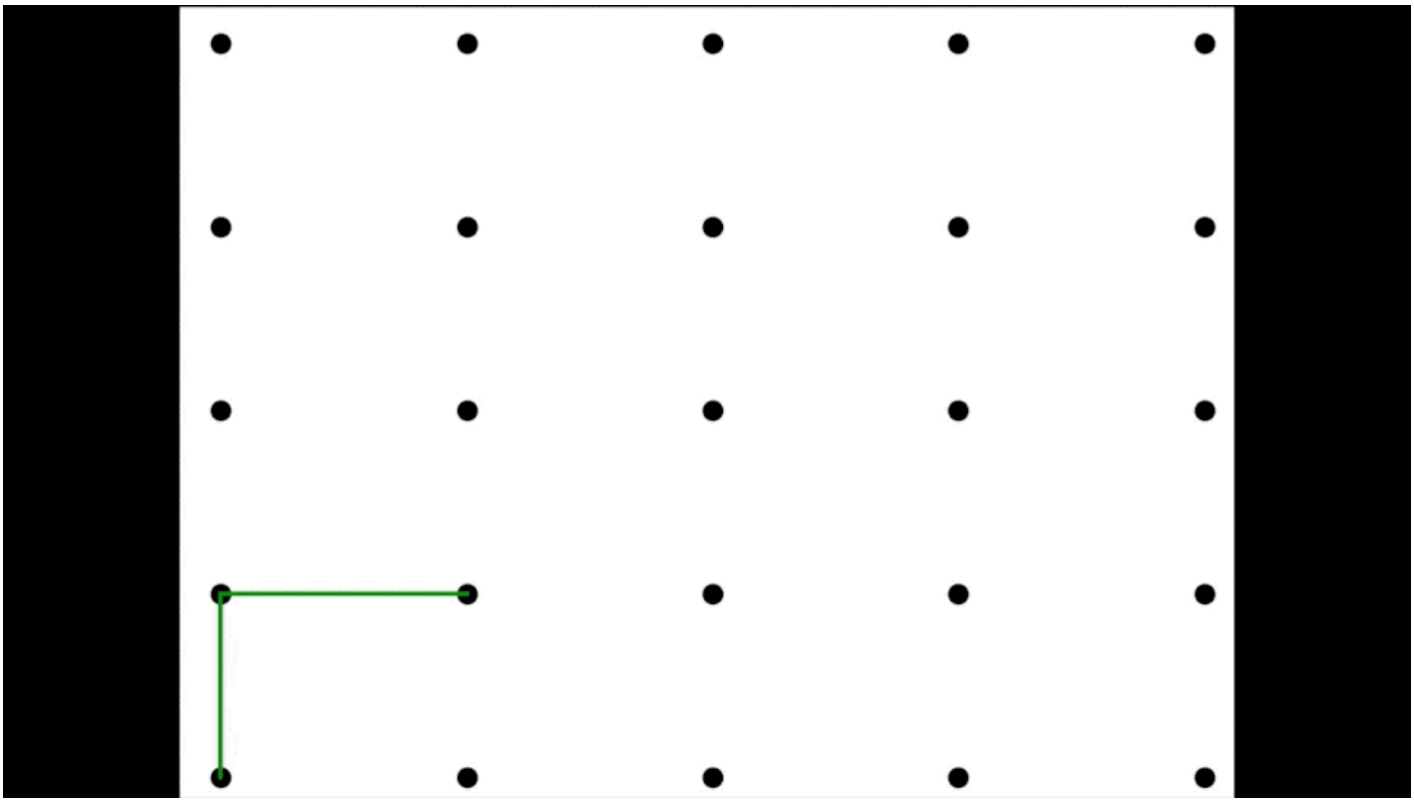
Alice (starts from lower left corner) will walk north(up) or east(right) with a probability of 50% each. At the same time and at the same speed, Bob (starts form upper right corner) will walk south (down) and west (left) with equal probabilities. What is the probability that they will meet?

Alice Random Walk

The following code simulates the random walk of Alice:

```
# Author: Manas Sharma
# License: MIT
# Random walker (Alice) walking on a 2d 4x4 grid.
import matplotlib.pyplot as plt
import numpy as np
# import matplotlib.animation as animation




# Points to draw the grid
x=[1,1,1,1,1,2,2,2,2,2,3,3,3,3,3,4,4,4,4,4,5,5,5,5,5]
y=[1,2,3,4,5,1,2,3,4,5,1,2,3,4,5,1,2,3,4,5,1,2,3,4,5]
# Draw the grid points
plt.scatter(x,y,c='black',s=50)
plt.title('Random walk of Alice')

# Starting position of Alice
pos = np.array([1,1])

# Arrays/Lists to store some positions to draw them later.
posX = []
posY = []
posX.append(pos[0])
posY.append(pos[1])

# Simulate the random walk upto nsteps steps
nsteps=8
for i in range(nsteps):
    randno=np.random.random_integers(1,2)
    if randno==1:# Right
        if pos[0]<5:
            pos = pos + [1,0]
        else:
            pos = pos + [0,1]
    else: # Up
        if pos[1]<5:
            pos = pos + [0,1]
        else:
            pos = pos + [1,0]
    posX.append(pos[0])
    posY.append(pos[1])
    # Draw the paths
    plt.plot(posX,posY,c='green',linewidth=3)
    # Pause for animation
    plt.pause(0.5)
plt.show()
```

Output

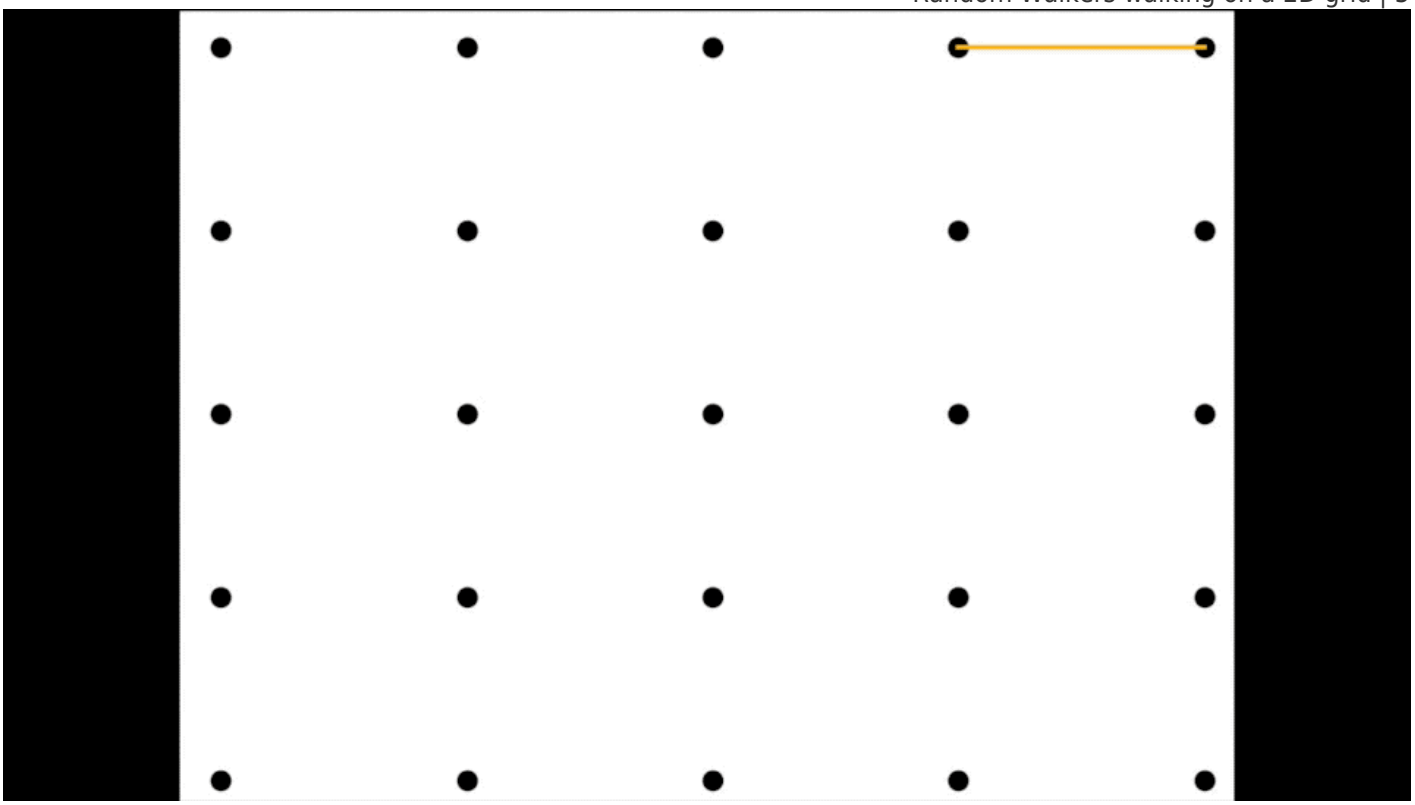


Bob Random Walk

The following code the random walk of Bob:

```
# Author: Manas Sharma
# License: MIT
# Random walker (Bob) walking on a 2d 4x4 grid.
import matplotlib.pyplot as plt
import numpy as np
# import matplotlib.animation as animation


# Points to draw the grid
x=[1,1,1,1,1,2,2,2,2,2,3,3,3,3,3,4,4,4,4,4,5,5,5,5,5]
y=[1,2,3,4,5,1,2,3,4,5,1,2,3,4,5,1,2,3,4,5,1,2,3,4,5]
# Draw the grid points
plt.scatter(x,y,c='black',s=50)
plt.title('Random walk of Bob')

# Starting position of Bob
pos = np.array([5,5])

# Arrays/Lists to store some positions to draw them later.
posX = []
posY = []
posX.append(pos[0])
posY.append(pos[1])

# Simulate the random walk upto nsteps steps
nsteps=8
for i in range(nsteps):
    randno=np.random.random_integers(1,2)
    if randno==1:# Left
        if pos[0]>1:
            pos = pos - [1,0]
        else:
            pos = pos - [0,1]
    else: # Down
        if pos[1]>1:
            pos = pos - [0,1]
        else:
            pos = pos - [1,0]
    posX.append(pos[0])
    posY.append(pos[1])
    # Draw the paths
    plt.plot(posX,posY,c='orange',linewidth=3)
    # Pause for animation
    plt.pause(0.5)

plt.show()
```

Output



OUTPUT: Random Walk simulation for Bob

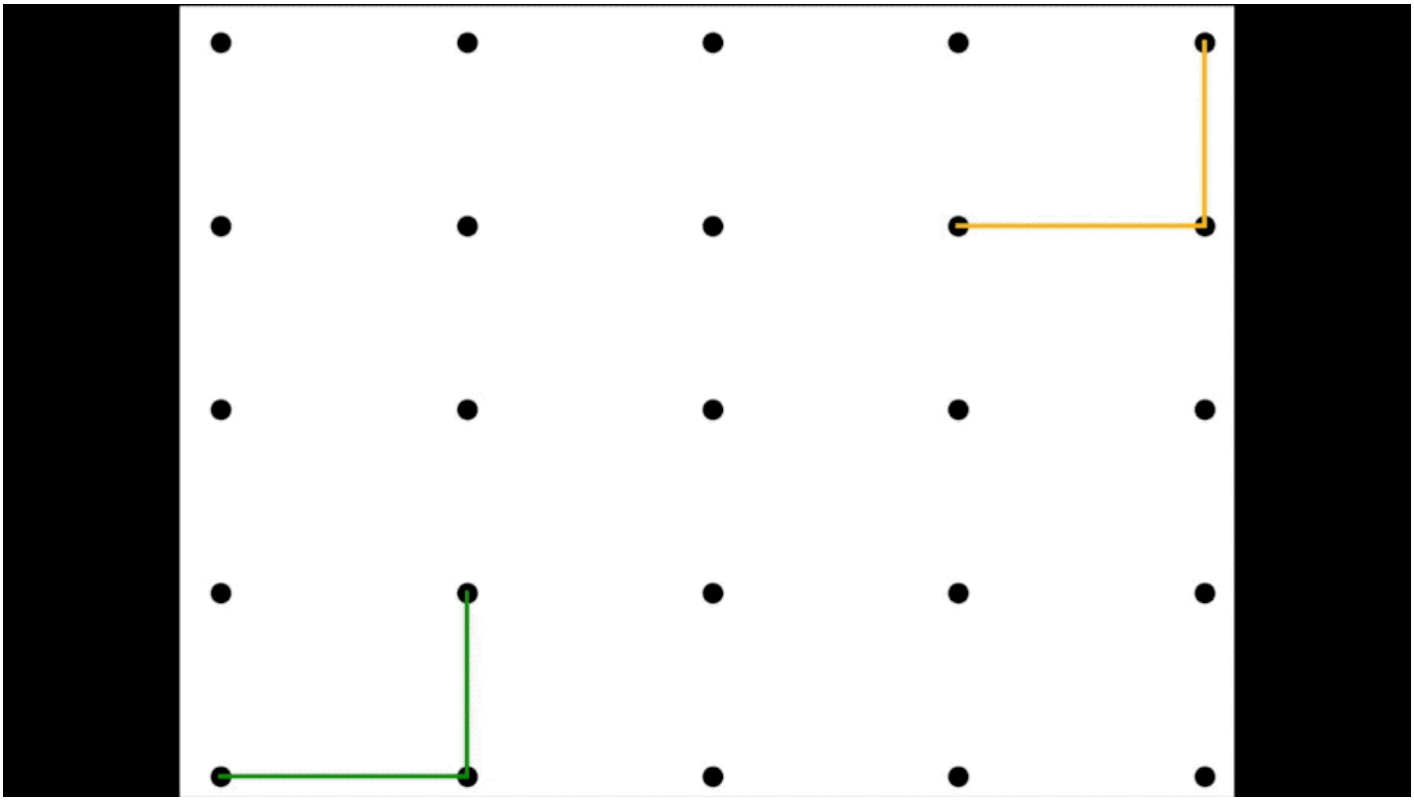
Alice and Bob Simultaneous Random Walks

The following code simulates the random walk of Alice and Bob starting simultaneously and going at the same speed.

```
# Author: Manas Sharma
# License: MIT
# Random walkers walking on a 2d 4x4 grid.

import matplotlib.pyplot as plt
import numpy as np
# import matplotlib.animation as animation




# Points to draw the grid
x=[1,1,1,1,1,2,2,2,2,2,3,3,3,3,3,4,4,4,4,4,5,5,5,5,5]
y=[1,2,3,4,5,1,2,3,4,5,1,2,3,4,5,1,2,3,4,5,1,2,3,4,5]
# Draw the grid points
plt.scatter(x,y,c='black',s=50)
plt.title('Random walk of Alice and Bob')

# Starting positions of Alice and Bob
posA = np.array([1,1])
posB = np.array([5,5])
```

```
# Arrays/Lists to store some positions to draw them later.
posX_A = []
posY_A = []
posX_B = []
posY_B = []
posX_A.append(posA[0])
posY_A.append(posA[1])
posX_B.append(posB[0])
posY_B.append(posB[1])

# Simulate the random walk upto insteps steps
nsteps=8
for i in range(nsteps):
    randno=np.random.random_integers(1,2)
    if randno==1:# Right
        if posA[0]<5:
            posA = posA + [1,0]
        else:
            posA = posA + [0,1]
    else: # Up
        if posA[1]<5:
            posA = posA + [0,1]
        else:
            posA = posA + [1,0]

    posX_A.append(posA[0])
    posY_A.append(posA[1])

    randno=np.random.random_integers(1,2)
    if randno==1:# Left
        if posB[0]>1:
            posB = posB - [1,0]
        else:
            posB = posB - [0,1]
    else: # Down
        if posB[1]>1:
            posB = posB - [0,1]
        else:
            posB = posB - [1,0]

    posX_B.append(posB[0])
    posY_B.append(posB[1])

    # Draw the paths
    plt.plot(posX_A,posY_A,c='green', linewidth=3)
    plt.plot(posX_B,posY_B,c='orange', linewidth=3)
    # Pause for animation
    plt.pause(1.0)
plt.show()
```

Output



OUTPUT: Random walks of Alice and Bob simultaneously

Probability calculation

The following code runs the complete simulation and calculates the probability of Alice and Bob meeting:

```
# Author: Manas Sharma
# License: MIT
# Probablitiy of Random walkers meeting on a 2d 4x4 grid.

import matplotlib.pyplot as plt
from matplotlib.pyplot import figure
import numpy as np
# import matplotlib.animation as animation


figure(figsize=(8, 8), dpi=120)


# Points to draw the grid
x=[1,1,1,1,1,2,2,2,2,2,3,3,3,3,3,4,4,4,4,4,5,5,5,5,5]
y=[1,2,3,4,5,1,2,3,4,5,1,2,3,4,5,1,2,3,4,5,1,2,3,4,5]

# Number of total events.
Ntrials = 500
```

```
# Counter to keep track of number of times they have met
nmeet = 0

# Run the different trials
for ntr in range(1,Ntrials+1):
    # Starting positions of Alice and Bob
    posA = np.array([1,1])
    posB = np.array([5,5])
    # Arrays/Lists to store some positions to draw them later.
    posX_A = []
    posY_A = []
    posX_B = []
    posY_B = []
    posX_A.append(posA[0])
    posY_A.append(posA[1])
    posX_B.append(posB[0])
    posY_B.append(posB[1])
    # Simulate the random walk upto nsteps steps
    nsteps=8
    for i in range(nsteps):
        #Plot the grid points
        plt.scatter(x,y,c='black',s=50)
        randno=np.random.random_integers(1,2)
        if randno==1:# Right
            if posA[0]<5:
                posA = posA + [1,0]
            else:
                posA = posA + [0,1]
        else: # Up
            if posA[1]<5:
                posA = posA + [0,1]
            else:
                posA = posA + [1,0]

        posX_A.append(posA[0])
        posY_A.append(posA[1])

        randno=np.random.random_integers(1,2)
        if randno==1:# Left
            if posB[0]>1:
                posB = posB - [1,0]
            else:
                posB = posB - [0,1]
        else: # Down
            if posB[1]>1:
                posB = posB - [0,1]
            else:
                posB = posB - [1,0]

        posX_B.append(posB[0])
        posY_B.append(posB[1])
```

```
        # Draw the paths
        plt.plot(posX_A,posY_A,c='green',linewidth=3,alpha=0.4)
        plt.plot(posX_B,posY_B,c='orange',linewidth=3,alpha=0.4)

        # Highlight the current path
        plt.plot([posX_A[-1],posX_A[-2]],[posY_A[-1],posY_A[-2]],c='green',linewidth=4)
plt.plot([posX_B[-1],posX_B[-2]],[posY_B[-1],posY_B[-2]],c='orange',linewidth=4)

        # Check if they met?
        if posB[0]==posA[0]:
            if posA[1]==posB[1]:
                # Increment meeting counter
                nmeet = nmeet + 1
                plt.scatter(posA[0],posA[1],c='red',s=250)
                plt.title('Nmeet = '+str(nmeet)+';      Ntrials = '+str(ntr)+';
Probability = '+str(round(probability,4)))
                plt.pause(1.0)
        # Calculate probability
        probability = nmeet/ntr
        plt.title('Nmeet = '+str(nmeet)+';      Ntrials = '+str(ntr)+';
Probability = '+str(round(probability,4)))
        #Pause for animation
        plt.pause(0.1)
        plt.clf()
    # Clear previous plot

print(probability)

plt.show()
```

Output

REFERENCES

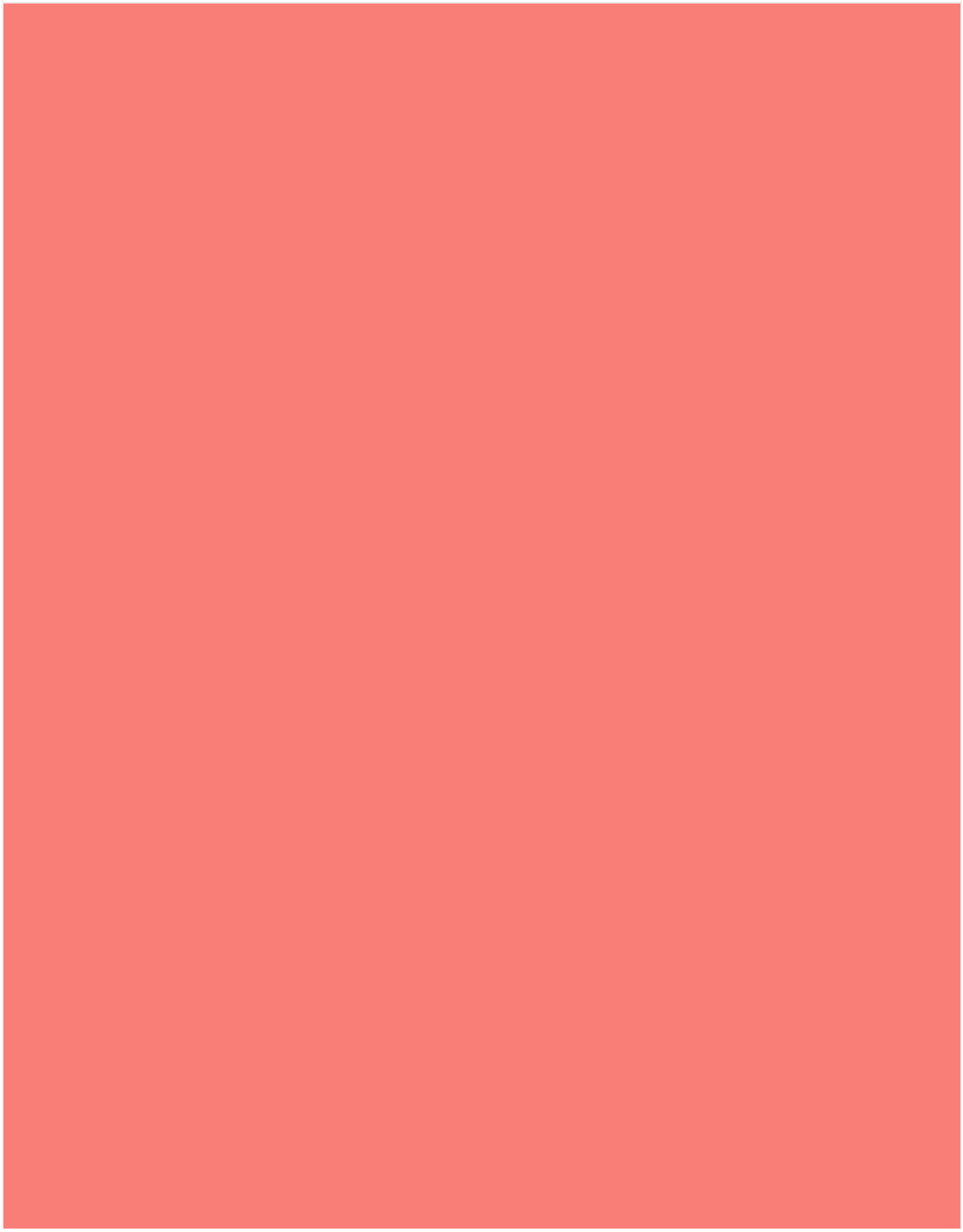
Inspiration:

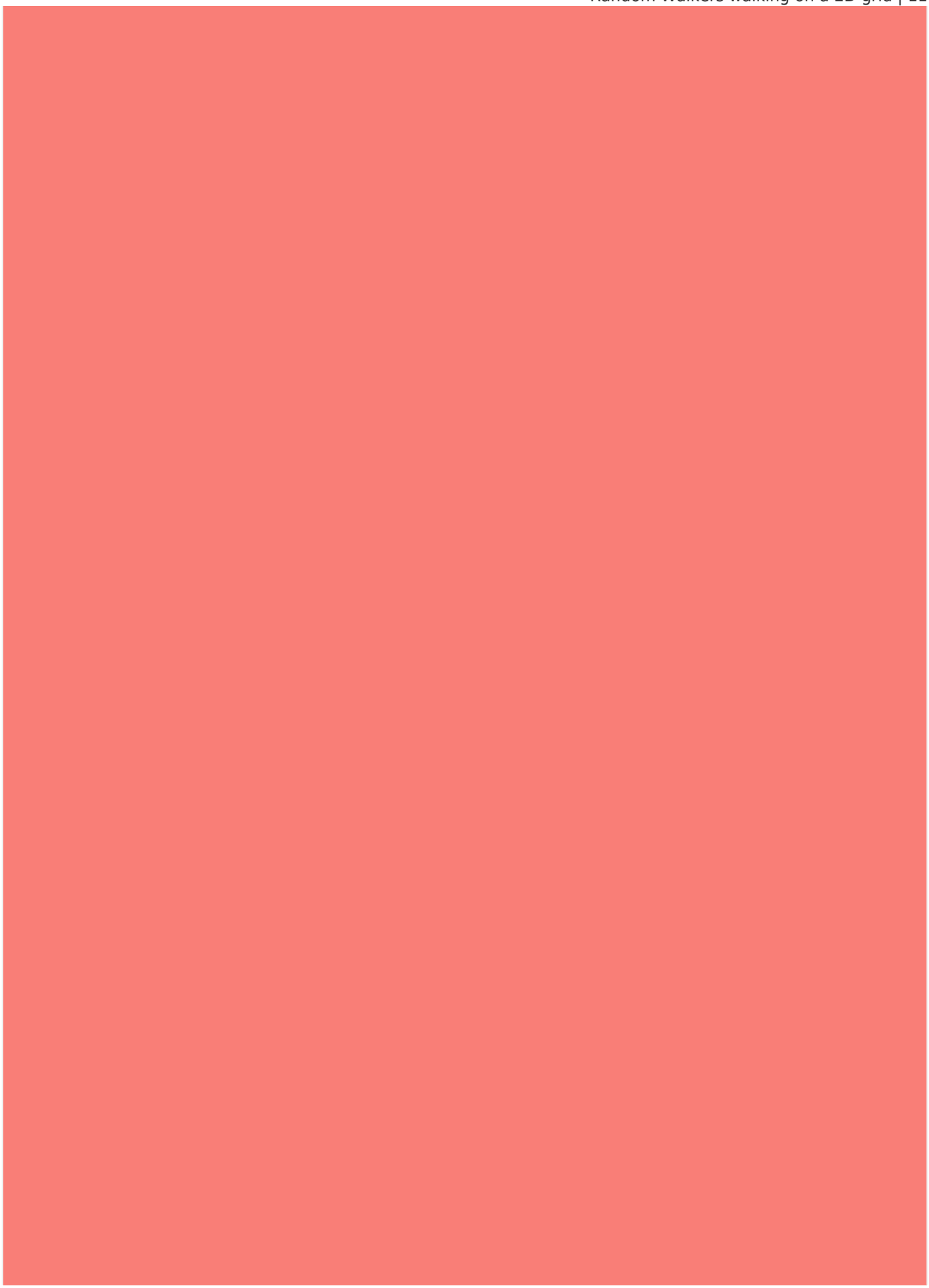
Manas Sharma

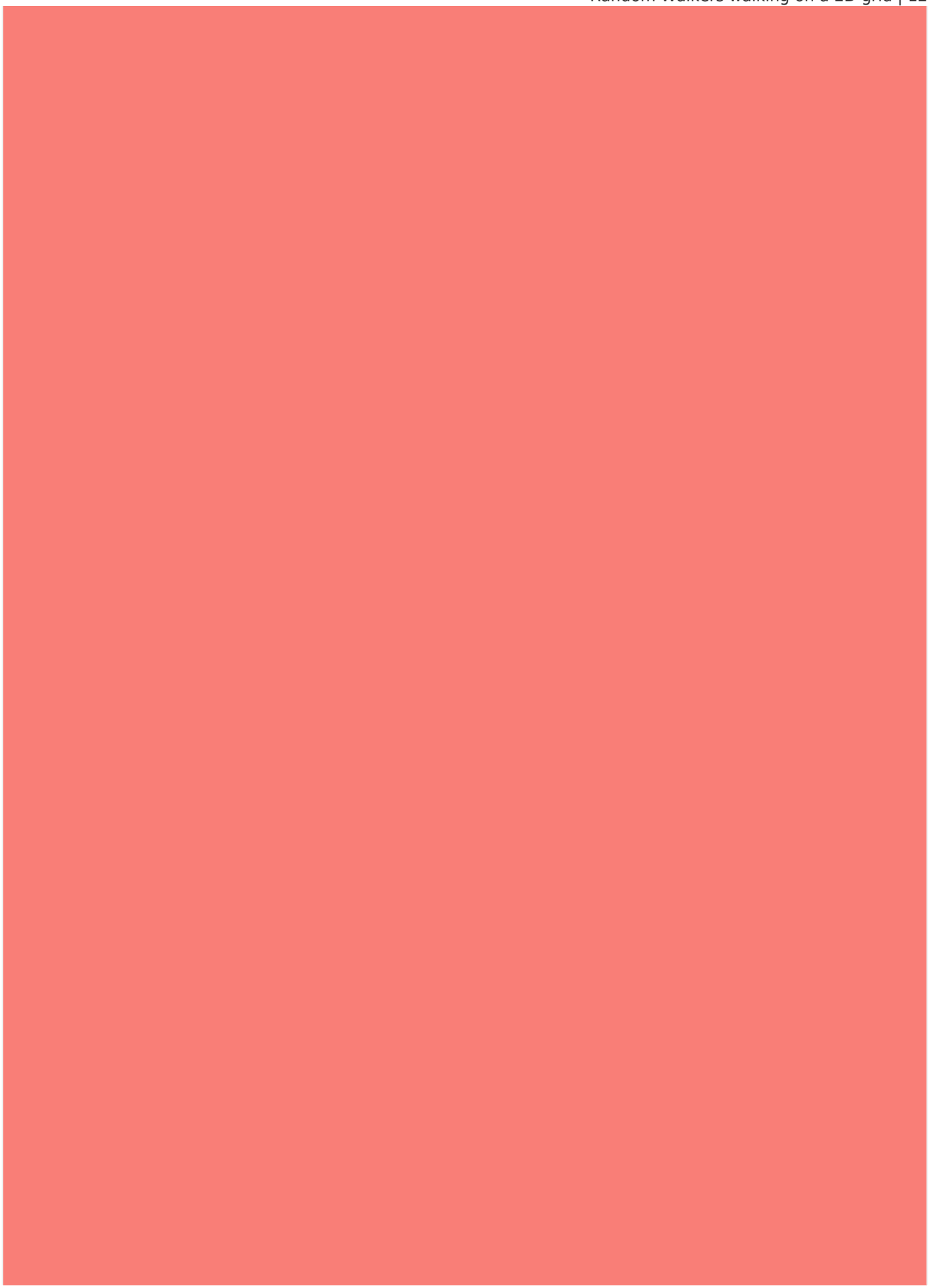
I’m a physicist specializing in computational material science with a PhD in Physics from Friedrich-Schiller University Jena, Germany. I write efficient codes for simulating light-matter interactions at atomic scales. I like to develop Physics, DFT, and Machine Learning related apps and software from time to time. Can code in most of the popular languages. I like to share my knowledge in Physics and

applications using this Blog and a YouTube channel.

manas.bragitoff.com/







Share this:

Click to share on Facebook (Opens in new window)

Click to share on Twitter (Opens in new window)

Click to share on WhatsApp (Opens in new window)

Click to share on Pinterest (Opens in new window)

Click to share on Reddit (Opens in new window)

Click to share on LinkedIn (Opens in new window)

Click to email a link to a friend (Opens in new window)

Click to print (Opens in new window)

Click to share on Tumblr (Opens in new window)

Click to share on Pocket (Opens in new window)

Click to share on Telegram (Opens in new window)

[wpedon id="7041" align="center"]