Assign_6_code_Hrishabh_Kulkarni

November 9, 2024

0.1 Question 2c

```python
[3]: import json
     import numpy as np

     with open("color_perception_data.json", "r") as file:
         color_data = json.load(file)

     wavelength = np.array(color_data["wavelength"]["data"])
     R = np.array(color_data["R_phosphor"]["data"])
     G = np.array(color_data["G_phosphor"]["data"])
     B = np.array(color_data["B_phosphor"]["data"])
     L = np.array(color_data["L_coefficients"]["data"])
     M = np.array(color_data["M_coefficients"]["data"])
     S = np.array(color_data["S_coefficients"]["data"])
     test_light = np.array(color_data["test_light"]["data"])

     LR = np.dot(L, R)
     MR = np.dot(M, R)
     SR = np.dot(S, R)

     MG = np.dot(M, G)
     LG = np.dot(L, G)
     SG = np.dot(S, G)

     LB = np.dot(L, B)
     MB = np.dot(M, B)
     SB = np.dot(S, B)

     mat = np.array([                              # Matrix of the above dot product
             [LR, LG, LB],
             [MR, MG, MB],
             [SR, SG, SB]
     ])

     L_test_light = np.dot(L, test_light)
     M_test_light = np.dot(M, test_light)
     S_test_light = np.dot(S, test_light)
```

```python
compared = np.array([L_test_light ,M_test_light, S_test_light])      #␣
  ↪Stored response to test light

weights = np.linalg.solve(mat, compared)
R_value = weights[0]
G_value = weights[1]
B_value = weights[2]

print("RGB weights are: \n")
print(f"R : {R_value}")
print(f"G : {G_value}")
print(f"B : {B_value}")
```

```
RGB weights are:

R : 0.42259299388355054
G : 0.0987425637870611
B : 0.5285525473174244
```

0.2 Question 3b

```python
[6]: import numpy as np

data = np.load("sensor_data.npy", allow_pickle=True).item()
A = data['A']
m, n = A.shape
y = data['y']
Intervals = 1

allones = np.ones((m, 1))
time = np.arange(m).reshape(-1, 1) * Intervals
B = np.hstack((A, allones, time))

# z = [x, alpha, beta]
z, residuals, rank, s = np.linalg.lstsq(B, y, rcond=None) # used -> least␣
  ↪square method wrt -> z

x = z[:n]
alpha = z[-2]
beta = z[-1]

print("Temperature Vector x:", x)
print("Bias Constant:", alpha)
print("Drift Constant:", beta)
```

```
Temperature Vector x: [24.90063639 21.61770814 33.8996902 ]
Bias Constant: 2.1937036086718855
```

Drift Constant: 0.18979612476283153