1928
K. N. Toosi University
of Technology

Faculty of Electrical Engineering

Course: Functional Brain Imaging Systems

Professor: Dr. Ali Khadem

Computer assinment 1

Forward Problem

Spring 2025

Prepared By:

Ramin Tavakoli, Mohammadreza Shahsavari, Mahdi Molaii

Introduction to the Forward Problem in functional Brain Imaging

This lab explores the forward problem in functional brain imaging, specifically focusing on Magnetoencephalography (MEG) and Electroencephalography (EEG). The forward problem involves calculating the signals (magnetic fields for MEG, electric potentials for EEG) that would be measured by sensors outside the head, given known electrical current sources within the brain. Understanding this process is fundamental for interpreting MEG/EEG recordings and accurately localizing brain activity.

For this exercise, we will simulate the head using a three-layer spherical model, a common simplification. This model incorporates the following standard electromagnetic parameters:

- Magnetic permeability: $\mu \approx \mu_o = 4\pi \times 10^{-7}$ (H/m)
- Electrical conductivity: $\sigma = 0.3$ $(\Omega \cdot m)^{-1}$
- Electrical permittivity: $\varepsilon \approx 10^5 \varepsilon_o = 8.85 \times 10^{-7}$ (F/m)

1. MEG forward problem.

a) Defining MEG Sensor Locations:

To simulate the MEG forward problem, we first establish the locations of 33 sensors on the upper scalp hemisphere (modeled as a sphere, radius $r_3 = 0.09$ m). The sensor layout is defined as follows:

- **Sensor 1:** Located at the apex (θ=0°, along the Z-axis).
- **Sensors 2-33:** Distributed over 8 longitudinal strips (i = 0 to 7), each at a constant azimuthal angle $\phi_i = 45° * i$.
- **Positions per Strip:** Four sensors are placed on each strip at elevation angles $\theta_j = 22.5° * j$ (j = 1 to 4).
- **Numbering:** Strip sensors are numbered 4i + j + 1. , with the θ=0° sensor (Z-axis) being number 1.

The goal is to determine the Cartesian coordinates (x, y, z) for all 33 sensors from these spherical definitions.

Task 1: Calculate and Visualize MEG Sensor Coordinates

Using the provided MEG_Sens.py script, implement the calculation of sensor Cartesian coordinates (using `numpy`) and create a 3D plot visualizing their positions (using `matplotlib`).

MEG_Sens.py

```python
import matplotlib.pyplot as plt
from mpl_toolkits.mplot3d import Axes3D
import numpy as np
from utility_functions import Conv_coordinates

# ----------------------  Calculate Coordinates of MEG sensors ------------------------

# TODO: Define the number of MEG sensors
num_points =

# TODO: Define theta angles for all sensors (zenith angle)
# First sensor at zenith (0°), then 4 sensors at each of the 8 longitudinal strips
# theta_degrees =

# TODO: Define phi angles for all sensors (azimuthal angle)
# phi_degrees =

# TODO: Convert degrees to radians for calculations
# theta =
# phi =

# TODO: Convert spherical coordinates to Cartesian coordinates
# radius =
# x, y, z =

# TODO: Save the calculated sensor coordinates to a file for later use
np.savez("sensor_coordinates.npz", x=x, y=y, z=z)
```

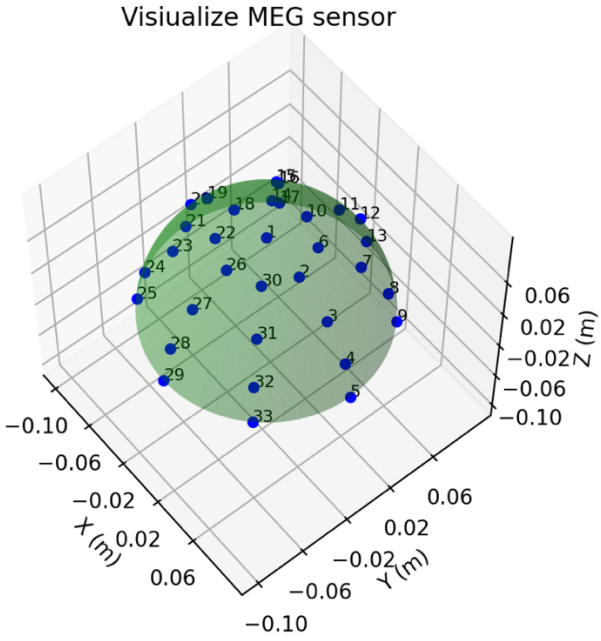

Figure 1) Visualization of all sensors placed on the scalp

- Explain the code.

b) Defining Dipole Sources

105 possible sources (dipoles) of brain electric current are randomly placed on a sphere of radius r_o = 7 cm. These dipoles are distributed using a uniform random distribution across the spherical surface. The dipoles are numbered based on their distance from the z-axis:

- Dipole 1: closest to z-axis
- Dipole 105: farthest from z-axis

This arrangement models potential neural activity sources within the brain, with the sphere representing a surface where active neural populations might be located.

Task 2

`Dipole_Loc.py`

```
# ---------------------------  Calculate Coordinates of Diapole  ----------------------

# TODO: Define the number of dipole sources to generate
# num_points =

# TODO: Initialize a random number generator with the default algorithm
# rng =

# TODO: Generate random theta values (polar angle) between 0 and π
# theta =

# TODO: Generate random phi values (azimuthal angle) between 0 and 2π
# phi =
```

After generating the random dipole positions, we need to arrange them according to their distance from the z-axis. This is accomplished using a utility function from `utility_functions.py`:

`utility_functions.py`

```
def sorted(x, y, z):
    # Calculate distance from z-axis for each point
    distances = np.sqrt(x**2 + y**2)
    sorted_indices = np.argsort(distances)
    x_sorted = x[sorted_indices]
    y_sorted = y[sorted_indices]
    z_sorted = z[sorted_indices]
    return x_sorted, y_sorted, z_sorted

def Conv_coordinates(phi, theta, radius):
    x = radius * np.sin(theta) * np.cos(phi)
    y = radius * np.sin(theta) * np.sin(phi)
    z = radius * np.cos(theta)
    return x, y, z
```

- Explain these two functions.

then we need to Convert spherical coordinates to Cartesian coordinates using `Conv_coordinates()` function in `utility_functions.py` and then Save sorted coordinates to a file:

```
# TODO
# radius =  # radius of the hemisphere in meters

# TODO: Convert spherical coordinates to Cartesian coordinates using utility_functions
# x, y, z =

# TODO: Sort the dipole coordinates based on distance from z-axis
# x_sorted, y_sorted, z_sorted =

np.savez("Dipole_coordinates.npz", x=x_sorted, y=y_sorted, z=z_sorted)
print(x_sorted.shape)  # Should output (105,)
```

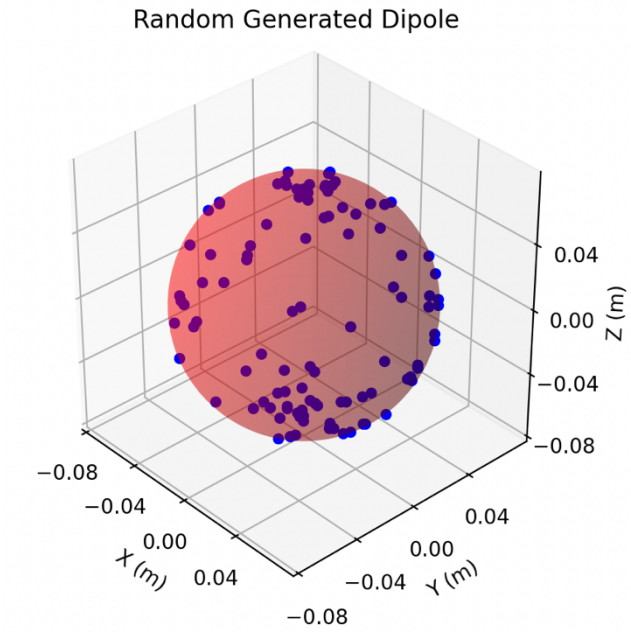

Figure 2) Visualization of all current sources located on the cerebral cortex

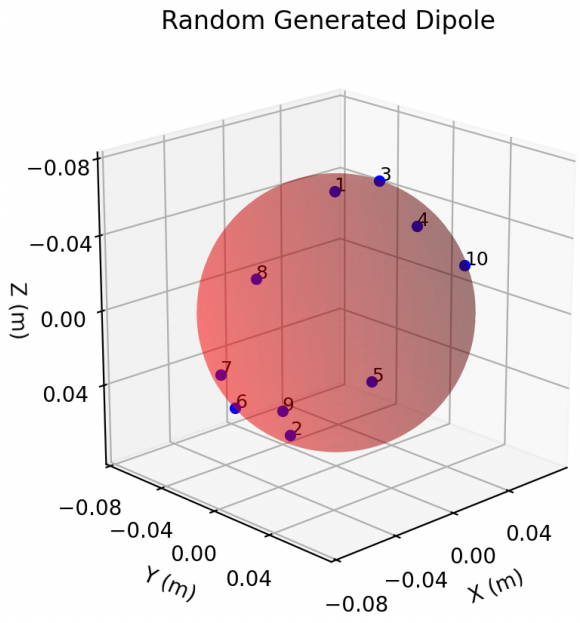

Figure 3) An example of a visualization of five sorted current sources located in the cerebral cortex for better understanding

- What would be the impact on our model if we clustered multiple dipoles in specific regions rather than distributing them randomly across the entire sphere.

c) Coordinates of Unit Vector

In the MEG forward problem, each sensor requires a unit vector oriented outward from the head model's center (perpendicular to the sensor surface). These unit vectors define the sensors' orientation and are essential for measuring magnetic field components.

Task 3

Calculate and visualize the unit vector for each sensor location.

Unit_Vect.py

```
# ------------------------  Calculate Coordinates of Unit Vector ---------------------
# TODO: Load the sensor coordinates from the previously saved file
data = np.load("sensor_coordinates.npz")
# x =
# y =
# z =
# radius = # radius of the hemisphere

# TODO: Initialize arrays for unit vector components
# ex =
# ey =
# ez =
print(ex.shape)  # Should output (33,)

# TODO: Calculate unit vectors for each sensor
for i in range(33):
    # TODO: Calculate the magnitude of the position vector
    # TODO: Normalize each component by dividing by the magnitude

# TODO: Save the unit vector components to a file for later use
np.savez("Unit_Vect_coordinates.npz", ex=ex, ey=ey, ez=ez)
```

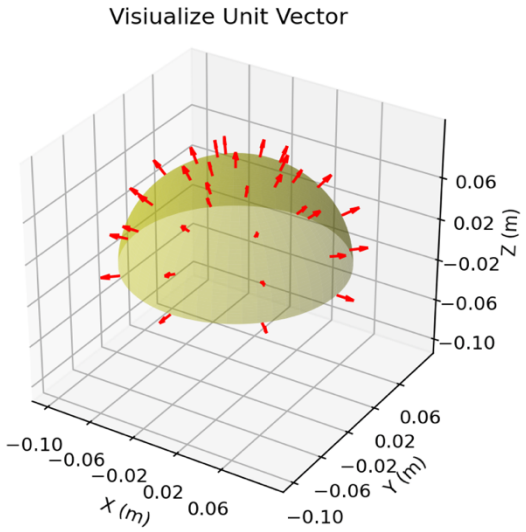

Figure 4) Visualization of radial vectors of sensors placed on the scalp

- Explain the code.

d) Head Model and Lead-Field Matrix

Head Model Parameters

The head is modeled using three concentric homogeneous spheres representing:

- Brain (radius $r_1 = 8.0$ cm), Skull (radius $r_2 = 8.5$ cm) and Scalp (radius $r_3 = 9.0$ cm).

With conductivities:

- Brain and Scalp: $\sigma_1 = \sigma_3 = \sigma$
- Skull: $\sigma_2 = \sigma/80$

```
# ---------------------------------  Constants ---------------------------------------

m = 33  # Number of MEG Sensor
n = 105
R0 = 0.07
R1 = 0.08
R2 = 0.085
R3 = 0.09

sigma = 0.3
sg1 = sigma
sg2 = sigma/80
sg3 = sigma

ep = 8.85 * pow(10, -7)
mu = 4 * math.pi * pow(10, -7)
```

Lead-Field Matrix (G)

The MEG lead-field matrix G represents the relationship between dipole sources and sensor measurements. For our configuration:

- What are the dimensions of the lead-field matrix G in this problem, and what do they represent?
- Explain the physical meaning of an element `G[i,j]` in the lead-field matrix.
- How does the lead-field matrix G relate dipole sources to sensor measurements?
- What will column 75 of G show?

Implement the lead-field matrix calculation using the formula then Plot column 75 (showing how one specific dipole affects all 33 sensors)

Task 4

task4.py

```
# ------------------------------  Loading Coordinates Data ------------------------------

# TODO: Load the coordinate data files for dipoles, sensors and unit vectors
# data1 =
# data2 =
# data3 =

# TODO: Extract and organize dipole coordinates into array
# rq_x =
# rq_y =
# rq_z =
# rq =

# TODO: Extract and organize sensor coordinates into array
# r_x =
# r_y =
# r_z =
# r =

# TODO: Extract and organize unit vectors into array
# er_x =
# er_y =
# er_z =
# er =

# ---------------------------  Calculate Lead Field Matrix ----------------------------

# TODO: Initialize the lead field matrix G
# G =

# TODO: Calculate lead field matrix components
# for i in range(m):
#     k = 0
#     for j in range(n):
#         ...
```

- Explain the code.
- How might the lead-field matrix change if we altered the conductivity ratio between skull and brain tissue?
- What would be the answer to the above question if we used a realistic model instead of a spherical head model?

Task 5: Radial Magnetic Fields from a Single Dipole Source

Calculate the radial component of the magnetic flux density (Br) at each MEG sensor due to a single dipole source, then create a visualization showing the field distribution on the *MEG sensor* and Plot the hemisphere surface.

- Single dipole source located at (r_o, θ_o, ϕ_o) = (7cm, 45°, 45°)
- Dipole moment vector $\vec{q}_o$ = [qx, qy, qz] = [0, 0, 1] (oriented along z-axis)
- 33 MEG sensors that saved in task 4 : `sensor_coordinates.npz file`

Task5.py

```
# ----------------------------  Loading Coordinates Data ----------------------------

# TODO: Load the sensor and unit vector coordinate data
# data2 =
# data3 =

# TODO: Extract and organize sensor coordinates into array
# r_x =
# r_y =
# r_z =
# r =

# TODO: Extract and organize unit vectors into array
# er_x =
# er_y =
# er_z =
# er =

# TODO: Define the dipole position parameters (45°, 45°, 7cm)
# theta =
# phi =
# radius =

# TODO: Convert from spherical to Cartesian coordinates
# x, y, z =

# TODO: Create the dipole position vector and moment vector
# rq =
# q =

# --------------------------  Calculate Lead Field Matrix --------------------------

# TODO: Initialize the lead field matrix G
# G =

# TODO: Calculate lead field matrix components
# for i in range(m):
```

```
#     k = 0
#     for j in range(n):
#         ...

# TODO: Calculate the radial magnetic flux density (Br) at each sensor
# B_r =

# TODO: Save the lead field matrix
# np.savez(...)
```

- Explain the code.
- What is the radial component of magnetic flux density (Br) and why is it important in MEG measurements?
- How does the distance between a sensor and the dipole source affect the measured magnetic field strength?
- Based on the visualization, where do you observe the strongest magnetic field measurements? Why do they occur at these locations?

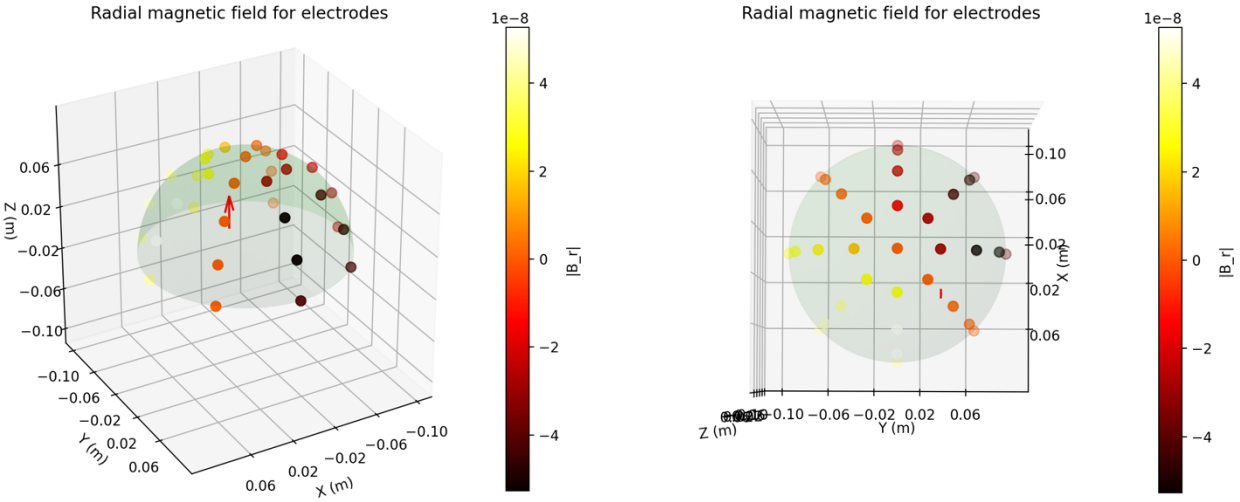

Figure 6) Radial magnetic fields recorded at the electrodes, indicated by color. The red vector is the source color.

Task 6: Simulating a Time-Varying MEG Signal from a Dipole Source

In this task, you'll extend the previous static dipole model to incorporate time variation. You'll simulate how a dipole with fixed position (7cm, 45°, 45°) and orientation [0,0,1], but time-varying magnitude, produces a changing MEG signal at sensor #30.

1. Use the same dipole position and orientation as before
2. Create a time series using the provided periodic function: `w(t) = 120 Sin(8πt) + 45 Sin(14πt) + 30 Sin(20πt) + 15 Sin(40πt) + 5 Sin(80πt)`
3. Generate 1 second of data at 1000 Hz sampling frequency

Task6.py

```
# ---------------------------- Calculate W(t) ----------------------------
# TODO: Define the periodic function w(t) as specified
# def w(t):
#     return ...
# TODO: Create a time vector with 1000 samples over 2 seconds
# t =
# TODO: Calculate w(t) values for each time point
# w_values =
```

4. Calculate the resulting MEG signal at sensor #30 over time
5. Create a plot showing the time-varying MEG signal

```

# -------------- Calculate the time-varying magnetic field at sensor 30  -------------

# TODO: Load the lead field matrix
# data =
# G =
# TODO: Define the dipole orientation vector
# q =
# q1 =
# TODO: Calculate the constant magnetic field at sensor 30
# B_r =
# TODO: Calculate the time-varying magnetic field at sensor 30
# B_r_new =
```

- What is the relationship between the dipole magnitude w(t) and the resulting magnetic field B_r_new?
- If we wanted to simulate two dipoles with different time-varying patterns, how would we modify this code?

- How would the MEG signal change if the dipole orientation varied over time instead of just its magnitude? Experiment it.

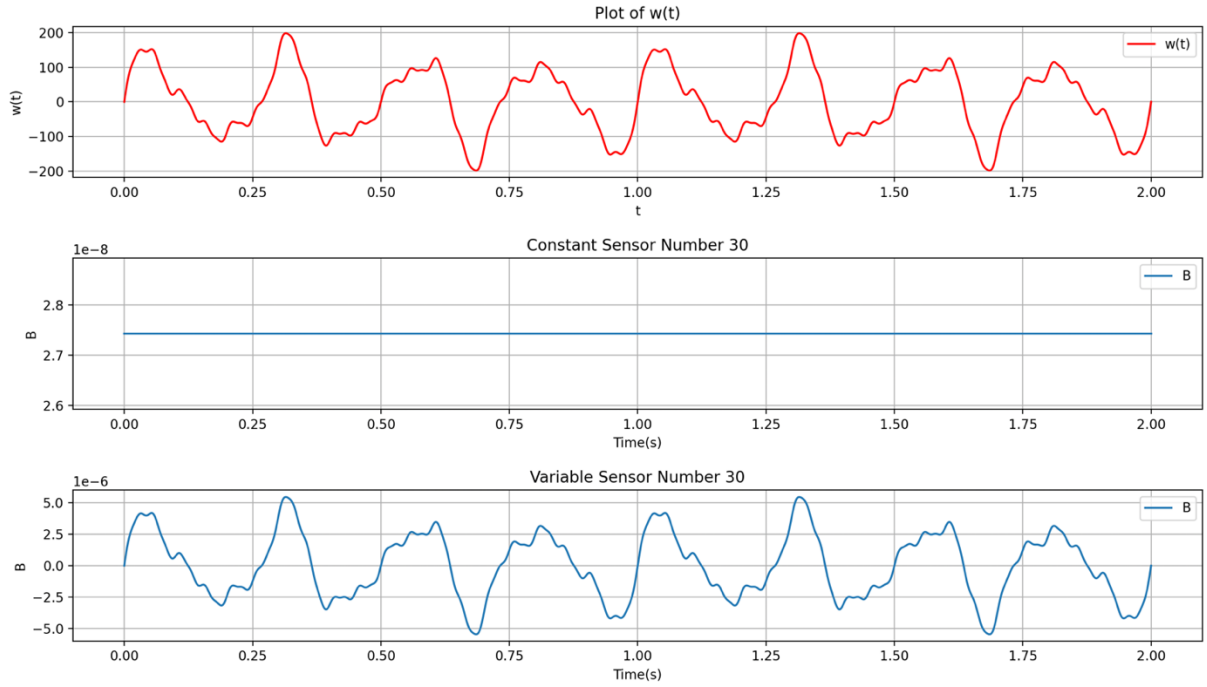

Figure 7) Radial magnetic fields recorded at the 30th sensor location

2. EEG forward problem

Task 7

Consider the formula for calculating the surface potential of the head in the three-layer spherical model that was explained in class. For an electric current source $\vec{m}$ = $[mx, my, mz]$ located at the point [0,0,z], write the mathematical relationship between the electric potential at the location of an EEG sensor in terms of $\vec{m}$ in matrix-scalar form.

Task 8

For the previous task, if the location of the electric current source is at an arbitrary point ($r0$, $\theta 0$, $\varphi 0$), modify the relation. Argue or prove that it is sufficient to replace φ with ($\varphi - \varphi 0$) and θ with ($\theta - \theta 0$) in the above relation.

Head Model Parameters : Same as Previous Tasks

EEG Lead-Field Matrix

In this task, we'll calculate the EEG lead-field matrix (L). Unlike the MEG lead-field matrix which relates dipole sources to magnetic field measurements, the EEG lead-field matrix relates the same dipole sources to electric potential measurements at EEG electrodes on the scalp.

Task 9

Implement the lead-field matrix calculation using the formula then Plot column 75 (showing how one specific dipole affects all 33 sensors)

Task9.py

```
# ------------------------  Loading Dipole, EEG sensor Cordinates ----------------------

# TODO: Load the coordinate data files for dipoles and sensors
# data1 =
# data2 =

# TODO: Extract dipole coordinates
# rq_x =
# rq_y =
# rq_z =

# TODO: Convert dipole coordinates from Cartesian to spherical
# r_0, theta_0, phi_0 =

# TODO: Extract sensor coordinates
# r_x =
# r_y =
# r_z =

# TODO: Convert sensor coordinates from Cartesian to spherical
# r, theta, phi =

# ----------------------------  Create EEG Lead_Field  --------------------------------
# TODO: Define function to calculate lead field components for a sensor-dipole pair
def Calc_L():

#     # Calculate the lead field components using the formula
    return a_ij

# TODO: Define function to calculate d_n coefficient for the three-sphere model
def d_n():
    # Apply the formula for d_n coefficient
    return dn
```

```
# TODO: Initialize the EEG lead field matrix L
# L =

# TODO: Define conductivity ratio
# xi =
# b =

# TODO: Calculate EEG lead field matrix components
for i in range():
    # k =
    for j in range():
        . . .
```

- Explain the code and the output.
- What are the dimensions of the lead-field matrix L in this problem, and what do they represent?
- Explain the physical meaning of an element `L[i,j]` in the lead-field matrix.
- How does the lead-field matrix L relate dipole sources to sensor measurements?
- What will column 75 of L show?

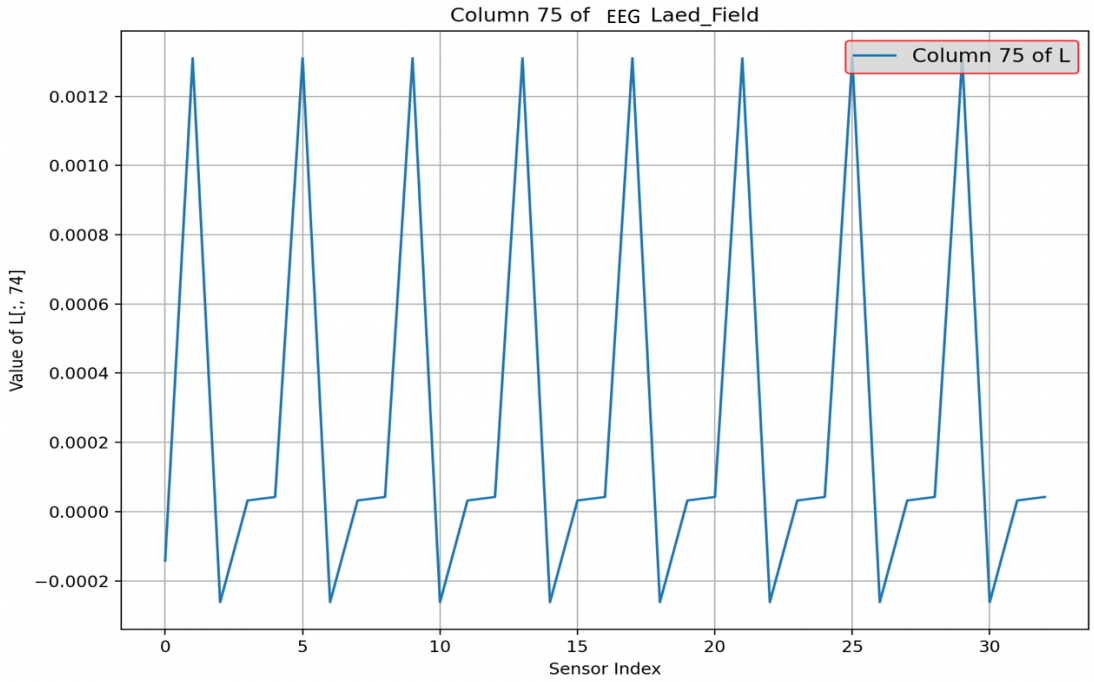

Figure 8) Visiualizing the column 75 of L

Task 10: Electric Potential from a Single Dipole Source in EEG

In this task, we'll calculate the electric potential at each EEG electrode due to a single dipole source, then create a visualization showing the potential distribution on the EEG sensors and plot the head surface like task 5 for MEG.

Task10.py

```
# ---------------------  Loading Dipole, EEG sensor Cordinates ----------------------
# TODO: Load sensor coordinates data
# data2 = ...

# TODO: Define dipole parameters
# theta_0 = ...
# phi_0 = ...
# radius = ...

# TODO: Convert dipole position from spherical to Cartesian coordinates
# x = ...
# y = ...
# z = ...

# TODO: Define dipole moment vector and position
# q = ...
# rq = ...

# TODO: Extract sensor coordinates
# r_x = ...
# r_y = ...
# r_z = ...

# TODO: Convert sensor coordinates from Cartesian to spherical
# r, theta, phi = ...

# ---------------------------  Create EEG Lead_Field  -------------------------------

# TODO: Define function to calculate lead field components for a sensor-dipole pair
def Calc_L():

    # Calculate the lead field components using the formula
    return a_ij

# TODO: Define function to calculate d_n coefficient for the three-sphere model
def d_n():
    # Apply the formula for d_n coefficient
    return dn

# TODO: Initialize the EEG lead field matrix L
# L =

# TODO: Define conductivity ratio
# xi =
```

```
# b =

# TODO: Calculate EEG lead field matrix components
for i in range():
    # k =
    for j in range():
    . . .


# ------------------------  Calculate EEG Voltage of sensors  ------------------------

# TODO: Calculate the electric potential at each sensor
# V = ...

# TODO: Save the EEG lead field matrix
# np.savez("EEG_Lead_Field.npz", L=L)
```

- Explain the code and output.

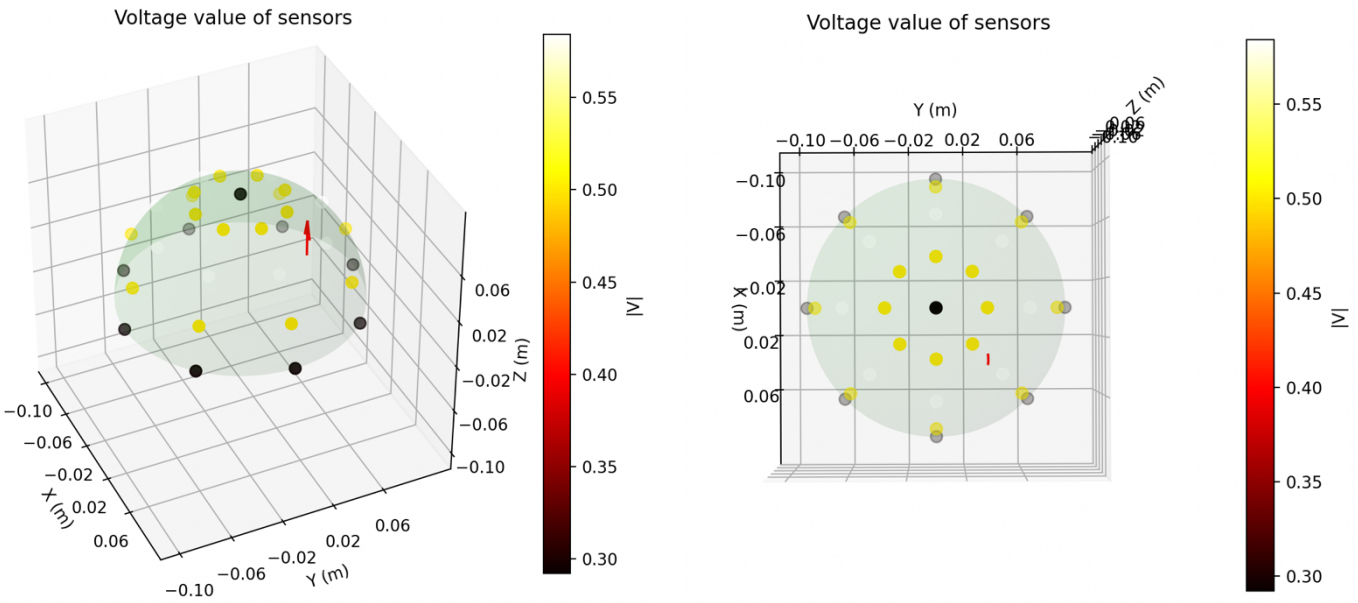

Figure 9) Electrical potential recorded at the electrodes, indicated by color. The red vector is the source color.

Task 11: Simulating a Time-Varying EEG Signal from a Dipole Source

Task 11 is essentially the EEG equivalent of Task 6, where instead of simulating time-varying MEG signals, we're now simulating time-varying EEG signals from a dipole source.

Task11.py

```
# ---------------------------- Calculate W(t) -----------------------------
# TODO: Define the periodic function w(t) as specified
# def w(t):
#     return ...
# TODO: Create a time vector with 1000 samples over 2 seconds
# t = ...
# TODO: Calculate w(t) values for each time point
# w_values = ...
# -------------- Calculate the time-varying magnetic field at sensor 30  ------------
# TODO: Load the lead field matrix
# data = ...
# L = ...
# TODO: Define the dipole orientation vector
# q = ...
# q1 = ...
# TODO: Calculate the constant magnetic field at sensor 30
# V = ...
# TODO: Calculate the time-varying magnetic field at sensor 30
# V_new = ...
```

- Explain the code and output.

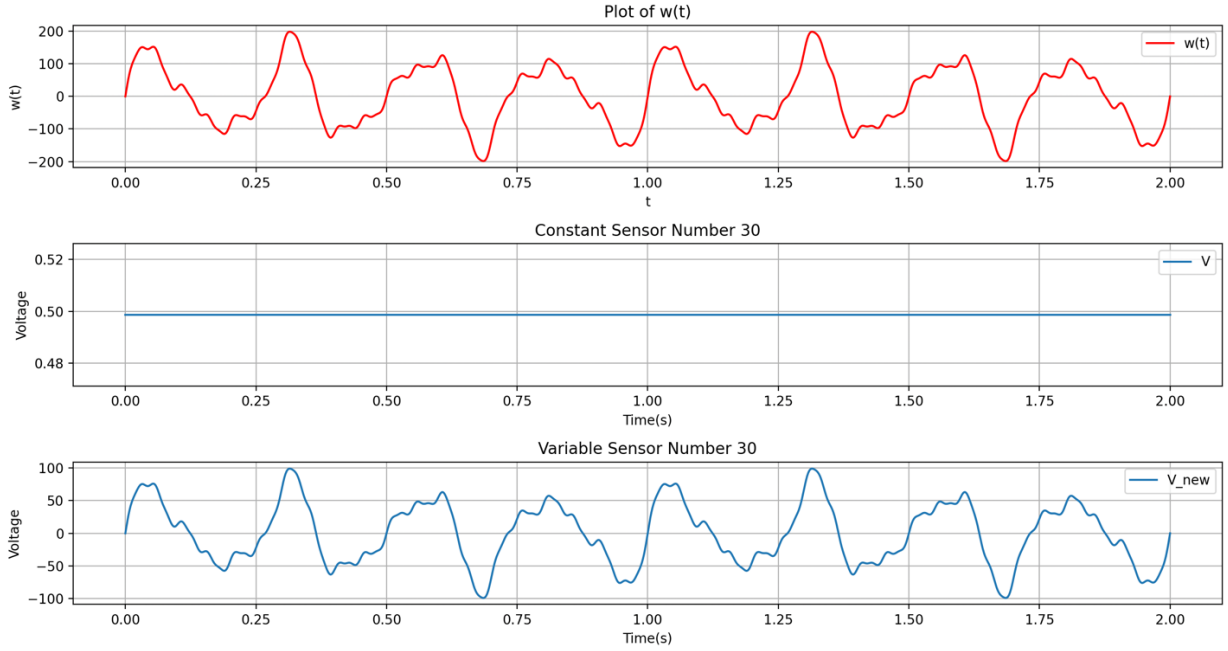

Figure 10) Voltage recorded at the 30th sensor location

توجه:

1. پاسخ ارسالی شما باید شامل یک گزارش برای پاسخ به سوالات داخل متن کارگاه در یک فایل ورد، به همراه کدهای مربوط به تسک‌ها باشد و هر دو در یک فایل rar ارسال گردند.
2. حتماً در موعد مقرر به بارگذاری پاسخ تمرین در سامانه VC اقدام کنید (تاخیر قابل قبول نیست)
3. در صورت بروز هرگونه ابهام یا سوال در مورد تمرین حتما با دستیاران آموزشی در ارتباط باشید.
4. لطفاً نام خانوادگی خود را به صورت لاتین به همراه شماره تمرین به عنوان نام فایل بارگذاری شده قرار دهید.

   به عنوان مثال :

   FIS_Ramintavakoli_CA1
5. زمان کافی جهت تحقیق جهت یافتن پاسخ تمارین به دانشجویان گرامی داده شده. لازم به ذکر است که هر شخص باید برداشت و نتایج تحقیقات خود را ارائه دهد پس به همین دلیل از کپی برداری پاسخنامه یکدیگر شدیدا خودداری فرمایید و در صورت مشاهده مسئولیت عواقب منفی متوجه تمامی افراد خاطی میگردد. همچنین ممکن است جلسه های مجازی تحت عنوان پرسش و پاسخ در خصوص گزارش دوستان گرامی برگزار شود.

موفق باشید.