



הפקולטה להנדסה

The Neural Processing and Brain Networks Lab

ויזואלייזציה של מיפוי מוחי עם גיריה חשמלית והקלות של פעילות מוחית

יובל צור (בן שלמה)

פרויקט שנה ד' לקריאת תואר ראשון בהנדסה

מנחה: שיר הרטמן

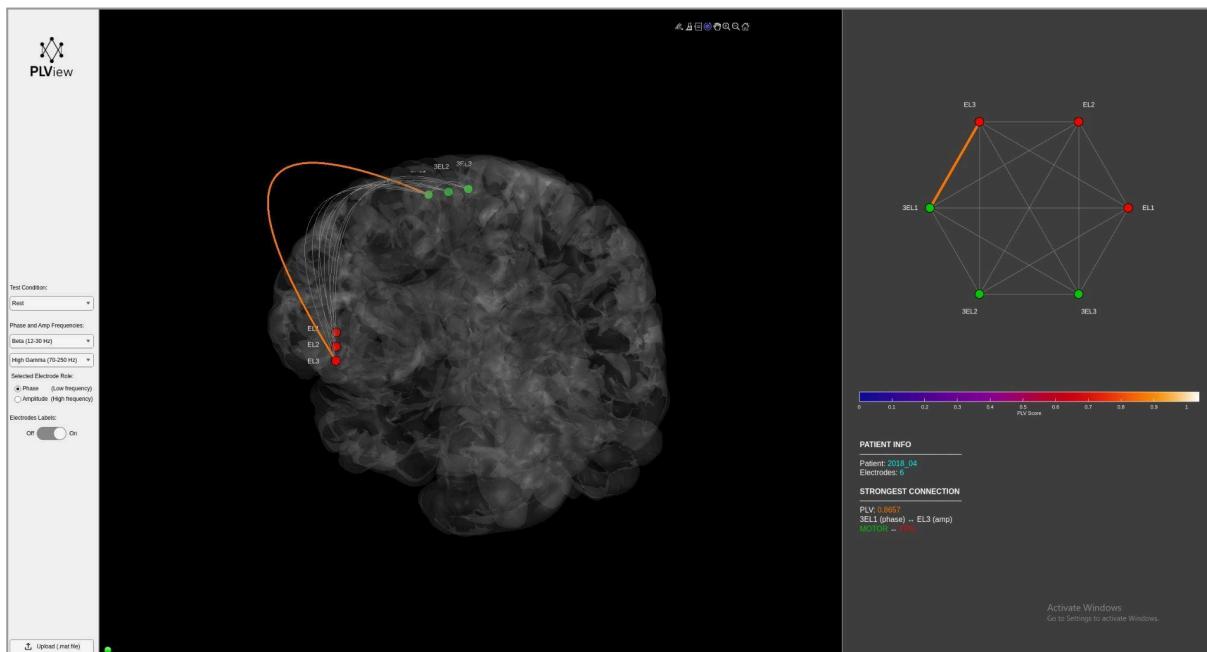
מנחה אקדמי: ד"ר יערה ארץ

אוקטובר 2025

תוכן עניינים

1. תקציר.....	3
2. תעודות.....	4
3. הצגת הבעיה.....	5
4. רקע עיוני	7
5. השיטה הנבחרת	12
6. בדיקת השיטה, תוצאות וניתוח.....	21
7. תרשימים מלבנים.....	31
8. מסקנות וסיכום.....	54
9. ריעונות להמשך.....	57
10.ביבליוגרפיה.....	59
11. נספחים.....	61

1. תקציר



* צילום מסך מתוך הכלי *PLView*, לאחר טעינת מידע שהוקלט מ-6 אלקטטרודות הממוקמות ב-2 רשתות שונות. (Fig. 1.)

ניתור של פעילות מוחית בזמן ניתוח מוח הינו פרוצדורה המתבצעת בניתוחים רבים על מנת למנוע פגיעה באיזורים הקשורים לתפקידים קריטיים בעיקר באיזורי הדיבור, הבנת שפה ותנועה. פולסים קצרים של גיריה חשמלית מופעלים באופן מקומי על המוח בזמן הניתוח על מנת להזזה איזורים קריטיים אלו. במידה והגיריה הובילה לפגיעה (הpicah) בתפקוד כלשהו, אישור הגיריה מזזה קרייטי לתפקוד והדבר משמש לקבלת החלטות קליניות באופן מיידי.

למרות היכולת הזאת לאיזורים קריטיים אלו, זיהוי איזורים הקשורים לתפקידים קוגנטיביים כמו קשב, היכולת להחליף בין משימות, ועוד, נשאר מתガר. כלי משלים לכך, כפי שהוצע ב & Assem (2023), הוא הקלטה של פעילות מוחית חשמלית באמצעות אלקטטרודות ייעודיות המונחות על קליפת המוח בטכנולוגיה הנקראת אלקטרוקורטיקוגרפיה (ECoG).

במטרה להבין את ארגון הרשתות המוחיות במוח האדם בעזרת הקלטות ECoG, חישבו קשריות תפקודית בין זוגות של אלקטטרודות בהתחשב במיקומים השונים. כדי לפענה את התוצאות בצורה טוביה, בנוינו כלי שמטרתו להציג בצורה ויזואלית את הקישוריות בין האלקטרודות.

הכלי שפותח כולל הצגה ומין של הנתונים לפי פרמטרים שונים, כמו: עוצמת הסync'רוניון בין האלקטרודות, מיקום האלקטרודות, סיווג האלקטרודות לרשתות מוחיות, ועוד.

2. תודות

הפרויקט יצא לדרך באוקטובר 2024 - שנה מأتגרת במירוץ, שנה של מלחמה והרבה ימי מילואים. זה דרש מני, מהמשפחה ומהאוניברסיטה התגויות מיוחדת כדי לאפשר את השלמת הפרויקט בצורה המיטבית, ועל כך אני מוקיר תודהعمוקה.

בראש ובראשונה, אני רוצה להודות למנהל שלי, ד"ר יערה ארץ, על ההתחשבות בכל הנסיבות, על הగמישות המחשבתית להתאים את הפרויקט לתחומי העניין והחוזקות המקצועיות שלי, ועל הדרך שבה אפשרה לי לעבוד בקצב ובצורה שנכונה לי, מבלי לוותר על איזוט ומקצועיות. הפרויקט הזה פותח בפניי דלת לעולם מרתק, עולם שאין מקווה להמשיך לפתח בו את דרכיו, להעמיק בו, ולהשתלב בו בעתיד - בתחום שבו אוכל לתרום, ליצור ולהמשיך לצמות.

תודה גדולה גם למנהל והמלואה האקדמית שיר הרטמן, על נכונות לעזר ולתמוך בכל חלק קטן וגadol בפרויקט, על הדחיפה למציאות, הלוי והיעוז המקצועי, ובעיקר על העין הבוחנת שהובילה את העבודה לרמה גבוהה יותר. אציין במיוחד את נוכנותך להמיר כל פגישה באוניברסיטה לפגישת זום, גם בהתראות קצרות במירוץ.

תודה למשפחה שלי, שבכל ארוחת שיש התעניינה, שאללה, התלהבה והקשיבה. אתם נתתם לי את המוטיבציה להמשיך ולהתמיד בפרויקט.

התודה הכי גדולה שמורה לאשתי, מORG. בתוך התקופה המורכבת זו יולדת את בתנו הראשונה, תמר, ובמקביל אפשרה לי להמשיך, דחפת אותה והשארת לי מקום לעבוד על הפרויקט עד שהושלים. מעבר לזה, את הייתה זאת שדחתת אותה מלכתחילה להירשם לתואר ולעזוב עבודה בהיינט - ובcludייר לא רק שהפרויקט הזה לא היה מסתiem, אלא גם המשע האקדמי שלו לא היה מתחילה.

ולבסוף - תודה לתמר, שעלה אף שלא הייתה שותפה רשמית בפרויקט, תרומתה בשעות שינה רצופות הייתה קריטית להצלחתו.

3. הצגת הבעה

קושי מרכזי במהלך ניתוח מוח פתוח (awake craniotomy) להסרת גידולים הוא להסיר את הגידול תוך פגיעה מינימלית ביכולות התפקודיות והקוגניטיביות של המטופל.

פגעה ברקמה בריאה עלולה לגרום לפגיעה בתפקודים הקרייטיים לח"י היום יום של החולים, כגון דיבור, הבנת שפה, תנועה, ותהליכי חשיבה שונים.

בכדי להבין בזמן האם הרקמה אותה מתכוונים המנתחים להסיר לocket חלק בתהליכי קרייטיים מתבצע שימוש בගיריה חשמלית מקומית ובcheinה האם מתרחש שינוי בהתנהגות (כמו הפרעה בדיבור, בהבנה של שפה, בתנועה). בוצרה זו המנתחים יודעים להימנע מפגיעה באותו איזור ובכך שומרים על איכות החיים של המטופל.

עדויות רבות מראות על כך שאוזן ההישרדות לאחר ניתוחים אלה גבוהים, אך רוב המטופלים מתארים שינויים ביכולות הקוגניטיביות שלהם שפוגעים באיכות החיים שלהם. במאבדה של ד"ר יערה ארץ שואפים למפות את האיזורים האלה הקשורים לתפקידים קוגניטיביים ברמה גבוהה, הכוללים בתוכם קשב, החלפה בין משימות, תכנון לטוויה רחוק, ועוד. תפקידים אלה נקראים **תפקידים ניהוליים (Executive function)** והם מושרים לפעולות של הרשת הפרונטופריאלית (Frontoparietal network, FPN) שבה אנחנו נתמקד.

הנתונים נאספו ממטופלים עם גידול מוח הנכנסו לניתוח מוח בערות מלאה במטרה להסרת הגידול. בזמן הניתוח המטופלים התבklassו לבצע משימות קוגניטיביות ברמות קושי עולות, בזמן שהפעולות המוחית שלהם נמדדה עם אלקטרודות ECoG. אלקטրודות מסווג זה מונחות על קליפת המוח החשוף, והן בעלות רזולוציה זמנית-מרחבית גבוהה.

עבוד וניתוח המידע במאבדה הוא ממשימה מורכבת הדורשת הבנה טכנית ותיאורית, ובנוסך יש לו מספר מגבלות המונעות מהם להתבצע בחדר הניתוח בזמן-אמת. אחת מהן היא יכולת להציג את המידע בצורה אינטואיטיבית וגרפית כדי להקל על מציאת האיזורים הקרייטיים למרחב. מגבלה נוספת היא יכולת לבצע חישובים מורכבים על המידע בזמן-אמת, וגם השוואה עם מטופלים רבים. מגבלות נוספות באנליה של נתונים אלו לרונטיות גם בזמן העבודה במאבדה, לדוגמה יכולת להראות ויזואלית איזה רשת יותר פעילה במטופל בודד.

בפרויקט זה בחרנו להמחייב את הפעולות המוחית בעזרת חישוב קשריות תפקודית. במדידות ECoG, ניתוח קשריות מאפשר לחושף דפוסים של תקשורת עצבית ולקור כיצד רשתות מוחיות משתנות בתגובה לגירויים, משימות או מצבים שונים. נעשה שימוש במדד Phase Locking Value (PLV), שהוא אחת השיטות הנפוצות למדידת קשריות המבוססת על סyncron פאזה בין שני

אותות. חישוב PLV מהוּה כלי עוצמתי להבנת דינמייקת התקשרות בין אזרורים שונים במוח בזמן משימה או במנוחה, ומאפשר לזהות האם שני אזרורים במוח "נעולים" על אותה פaza בתדר מסוים - כלומר, פעילים במצב מתואמת בזמן.

ויזואлизציה הינה כלי הכרחי להבנת תופעות רחבות יותר וביניהן שינויי משמעותיים בפעולות המוחית וקשרים בין ממדים ונתונים שונים.

באמצעות ייצוג גרפי של מידע מורכב, ניתן לזהות דפוסים, מגמות וחריגות שקשה או בלתי אפשרי לגלוּת בעת בחינת מידע גלומי. ויזואлизציה מאפשרת לחקרם ולטפלים לצפות בשינויים לאורך זמן, להשוות בין מצבים שונים, ולהזהות קורלציות בין משתנים רבים.

בקשר של פעילות מוחית, כלים ויזואליים כגון מפות חום, גרפים תלת-ממדיים וייצוגים טופוגרפיים מסייעים בהערכת אזורים המוח הפעילים, עצמת הפעולות ואופן התקשרות בין אזרורים שונים.

יתרה מכך, ויזואлизציה עילה הופכת למצאים מדעיים מורכבים לנגישים ומובנים גם לקהלים שאינם מומחים, ובכך מקלת על קבלת החלטות מושכלות בסביבות קליניות ומחקריות.

בשרה התchapונה, המוטיבציה לייצר כלי המציג את הפעולות המוחית במצב גרפית בזמן-אמת היא להגעה למיפוי טוב יותר של המוח, ספציפית באזרורים שאחראים לתפקידים הקרייטיים. (Executive functions

4. רקע עיוני

הרקע העיוני יעסוק במספר נושאים:

1. טכנולוגיית ECoG

- a. איסוף הדadata
- 2. קישוריות תפקודית
 - a. רשותות מוחיות
 - b. תפקודים ניהוליים (Executive funtions)
- c. רשת NPN
- d. חישוב קישוריות תפקודית
- e. חישוב PLV

3. מה אנחנו מצפים לראות

1. טכנולוגיית ^טECoG

אלקטרוכורטיקוגרפיה (ECoG) היא שיטת מדידה נוירופיזיולוגית המתבצעת ישירות על פני קליפת המוח (cerebral cortex) באמצעות רשת של אלקטרודות המונחות על פני החיצוניים של המוח במהלך ניתוחים נירוכירורגיים. בניגוד ל-EEG, שבו האות נמדד מחוץ לאגולגולת, ECoG מספקת אותות בעלי רוחלואה מרחבית וזמן גבואה במילוי, עם יחס אות-לרעש משופר ויכולת למדוד פעילות מוחית מקומית. טכנולוגיה זו משמשת בעיקר לצורך המוקד האפילפטי אצל חוליאפילפסיה עםידה לתרופה או במטרה מחקרית, להבנת מנגנון עיבוד עצבי, ולחקור רשותות מוחיות בבני אדם בזמן אמת.

a. איסוף הדadata¹

הדadata במעבדה נאסף במהלך מוח בערות (awake craniotomy) בעזרת סטריפים (strips) של 4 אלקטרודות ECoG. (בכל סטריף) במהלך הניתוח ניתן למטופלים 2 משימות ברמות קושי משתנות, מטלה קלה (Easy) ומטלה קשה (Hard), ובנוסף הוקלטו מדדים בזמןמנון. (Rest) המטלה הקלה היא ספירה פשוטה בסדר עולה - "1...2...3...". המטלה הקשה היא ספירה משתנה של אותיות ומספרים ליטיגין - "1...a...b...c...3...2...". היא דורשת שליטה קוגניטיבית גבוהה יותר כיוון שהיא מעורבת תכנון, גמישות מחשבתיות, שמירה ועיבוד של כלים בו-זמןית, ויכולת להחליף בין מצבים פוליה שונים - תפקודים המזוהים עם רשות

ה-*FPN* (Frontoparietal Network). (ירוחב בהמשך) במצב מנוחה התבקו המטופלים לשbat בשקט ולא דיבור במשך מספר דקות. CIDOU מחקרים קודמים, הרשות הפרונוטו-פריאטאלית פעילה יותר בזמן משימות קוגנטיביות מהסוג זהה. מטרת הפרויקט היא לחשוף קישוריות תפקודית בהתחשב ברשומות השונות תוך התמקדות ב-*FPN*, בעזרת שיטת *Phase-Amplitude coupling* ולבנות כלי סייעור לאפיון קישוריות של קליפת המוח הקדםית עם סנסורים רבים שמנוחים על המוח (ECoG), ספציפית באיזורים שאחראים לתפקידים הניהוליים.

2. קישוריות תפקודית

קישוריות תפקודית (Functional connectivity) מתייחסת לאופן שבו אזורים שונים במוח מחוברים ביניהם וכיידם מידע זורם ביניהם. קישוריות זו חיונית לתפקידים קוגניטיביים, רגשיים וגופניים, שכן אזורים שונים במוח מתקשרים זה עם זה כדי לבצע משימות. מכך קישוריות מבוסס EEG או ECoG מספקים תובנות חשובות על ארגון הרשות במוח במצב מנוחה, ביצוע משימות קוגניטיביות, או על הפרעות נוירולוגיות שונות.

a. רשותות מוחיות¹⁰

המוח האנושי מאורגן במבנה של רשתות עצביות (Brain Networks), שכל אחת מהן אחראית על קבוצת תפקידים ייחודית. רשותות אלו מתקשרות זו עם זו באופן DINAMI, ומרמת הפעולות והקישוריות ביניהן ניתן ללמוד על מגנוני עיבוד מידע, קשב, קבלת החלטות ותוכנן. שבע רשותות המוח העיקריות המצוות לעיתים קרובות הן: רשת ברירת המחדל (DMN), רשת הקשב הדורסליית (DAN), רשת הקשב הווונטראלית (VAN), רשת ניהול המרכזית (CEN, נקראת גם NP), הרשות הלימבית (LIMN), הרשות הסנסומוטורית (SMN), ורשת הראייה (VIS). רשותות אלו מייצגות אזורים מחוברים במוח שעובדים יחד כדי לבצע פונקציות קוגניטיביות ופיזיות שונות, כאשר חלק מהפונקציות המורכבות יותר נובעות מאיינטראקציה בין רשותות רבות.

b. תפקידים ניהול¹² (Executive functions)

תפקידים ניהול הם קבוצה של יכולות קוגניטיביות גבוהות המאפשרות שליטה והתאמנה של ההתנהגות למטרות משתנות. יכולות אלו אחראיות על היכולת ליזום, להתמיד, לעכב, לווסת, להבחן ולשנות, ומעורבים ביכולת הקשב וזיכרון העבודה. הן מתווכות בעיקר על ידי אזורים פרונטליים במוח, ובפרט הרשות הקדםית-פריאטאלית. (*FPN*)

c. רשת¹¹ FPN

רשת השליטה הקדםית-פריאטאלית (Fronto-Parietal Network, *FPN*) היא אחת מהרשותות המרכזיות במוח האחראיות על תפקידים ניהול (Executive functions) ועל ייסודות קוגניטיבי. היא

כלולת אזורים באונה הפרונטלית ובאונה הפריאטלית, ופעולת כמרכזי בקירה המאפשר מעבר בין משימות, קבלת החלטות, ושמירה על מיקוד. הרשות מתאפיינת בקשריות גבוהה עם רשותות אחרות, דבר המאפשר לה למלא תפקיד חשוב בתיאום פעילות מוחית רחבה.

ד. חישוב קשריות תפקודית

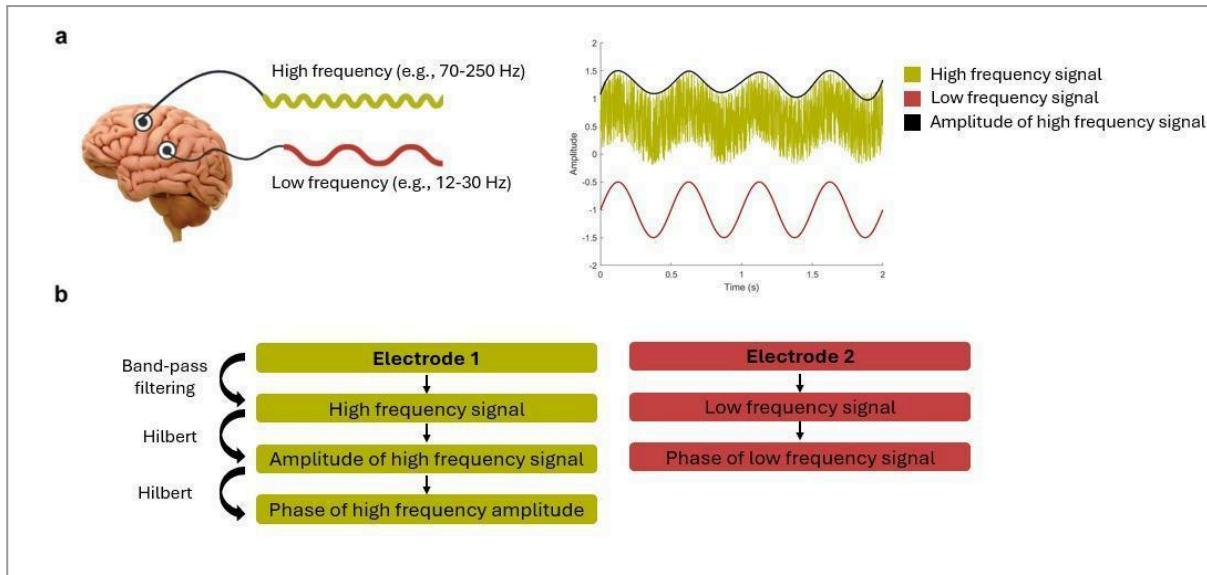
חישוב קשריות תפקודית מבוצע על ידי ניתוח היחסים בין אוטות נירוגניים מאזורים שונים במוח. קיימים מדרדים מגוונים, כגון: מתאם (correlation), קוורנטיות (coherence), סיבתיות (causality), וינכרון פאזה (phase synchrony). מטרת חישובים אלו היא לאזחות עד כמה האזורים פועלים בסyncrown או משפיעים זה על זה. במדידות ECoG, ניתוח קשריות מאפשר לחוש דפוסים של תקשורת עצבית ולחקר כיצד רשותות מוחיות משתנות בתגובה לגירויים, משימות או מצבים שונים. בפרויקט זה נעשה שימוש במדד Phase Locking Value (PLV), מסווג סינכרון פאזה (phase synchrony), לצורך מדידת קשריות תפקודית.

ה. חישוב PLV¹³

מדד PLV (Phase Locking Value) הוא אחת השיטות הנפוצות למדידת קשריות המבוססת על סינכרון פאזה בין שני אוטות. הממדד בוחן עד כמה ההפרש בין הפאזה של שני אוטות נשמר קבוע לאורך זמן. ערך PLV גבוה מעיד על סינכרון יציב - כלומר, שהאזורים המוחיים מתואימים בפעולותם - בעוד שערך נמוך מצביע על קשריות חלה או אקראית. חישוב PLV מהווה כלי עוצמתי להבנת דינמיות התקשרות בין אזורים שונים במוח בזמן משימה או במנוחה, ומאפשר לאזות האם שני אזורים במוח "נעולים" על אותה פאזה בתדר מסוים - כלומר, פועלים בצורה מתואמת בזמן.

תקופדים ניהוליים מרובים תיאום בין אזורים פרונטליים ופריאטליים - בדיקות איזורים המרכיבים את רשת ה-FPN. תיאום זה מתבטא בסינכרון זמני בין האותות החשימיים של הרשותות, בעיקר בתדרים מסוימים (כמו Beta או High Gamma).

נפרט כיצד מתבצע החישוב לפי המאמר¹³:



*תרשים חישוב PLV באמצעות מסן Hilbert. קרדיט: שיר הרטמן. (Fig. 2)

הчисוב מתבסס על ניתוח שתי אותות בתדרים שונים המוקלטיים מאזורים שונים במוח. (Fig.2.a). תהליך החישוב מתחילה בסינון האותות לשני תחומי תדר עיקריים.

מחקרים קודמים הראו עליה בטוחה ה **High Gamma** בין Hz 70-250 ברשת ה-**FPN** כشكוצי המשימה גדול, ובהתאם הראו גם ירידה בפעולות בטוחה ה **Beta** בין Hz 12-30, ולכן בחרנו להתמקד בטוחה תדריות אלה. (בתפריט הכליל ניתן לנגן גם לפ' טוויחים נוספים לבחירת המשתמש האות בתדר הגבוח (HG) מייצג פעילות עצבית מהירה ומקומית הקשורה לעיבוד מידע ולפעולות קוגניטיבית, בעוד שהאות בתדר הנמוך (Beta) משקף תנודות איטיות ורחבות יותר, המօסנתות את הקשר בין אזורים מרוחקים במוח. השילוב ביניהם מאפשר לבחון את יחסי הסync'רין בין דינמיות מהירות ואיטיות, שהם הבסיס לתקשרות בין רשתות מוחיות. (בקצהה: תנודות איטיות פועלות כ"אות נשא" המסנכרן את פעילות האזורים השונים, בעוד שתתנדות המהירות "מקודדות" את המידע המוקומי המועבר דרך. מנגנון זה, הידוע כ-**CFC** (Cross-Frequency Coupling) או **PAC** (Phase-Amplitude Coupling), מאפשר לאזורים מרוחקים לתאם ביניהם את תזמון העיבוד (הזמן).

בשלב הבא (Fig.2.b), נעשה שימוש בטרנספורם הילברט (Hilbert) - מסנן מתמטי המאפשר להמיר את האות הממשי לאוט מרוכב שטמנו ניתן לחץ את שני המרכיבים המרכזיים של כל תדר: מעטפת האמפליטודה של האות בתדר הגבוח (amplitude envelope) והפazaה של האות בתדר הנמוך (instantaneous phase). בעזרת טרנספורם הילברט מתקבלת הפazaה הרגעית של כל אות בזמן, מה שמאפשר לחשב את הפרש הפazaות בין שני אותות בזמן. לאחר הפקת הפazaה מכל אלקטרודה, מחושב ה-**PLV** באמצעות ממוצע אקספוננציאלי של הפרשי הפazaות בין שני האותות $\phi_{A_{HG}}(t, n) - \phi_{P_{Beta}}(t, n)$, על פי הנוסחה:

$$PLV = \left| \frac{1}{N} \sum_{i=1}^t e^{i(\phi_A(i) - \phi_p(i))} \right|$$

ערך PLV הקרוב ל-1 מצביע על סyncron פאזה גבוהה (כלומר, שני האזוריים פעילים בקצב ובתיאום קבוע), בעוד שערך קרוב ל-0 מעיד על פעילות בלתי מתואמת. באמצעות גישה זו ניתן להעריך את עצמת הקשריות בין אזוריים שונים במוח ולזהות את רמות הסyncron בין רשתות, במיוחד במצבים קוגניטיביים הדורשים שליטה ובקירה, כמו תפקודים ניהוליים.

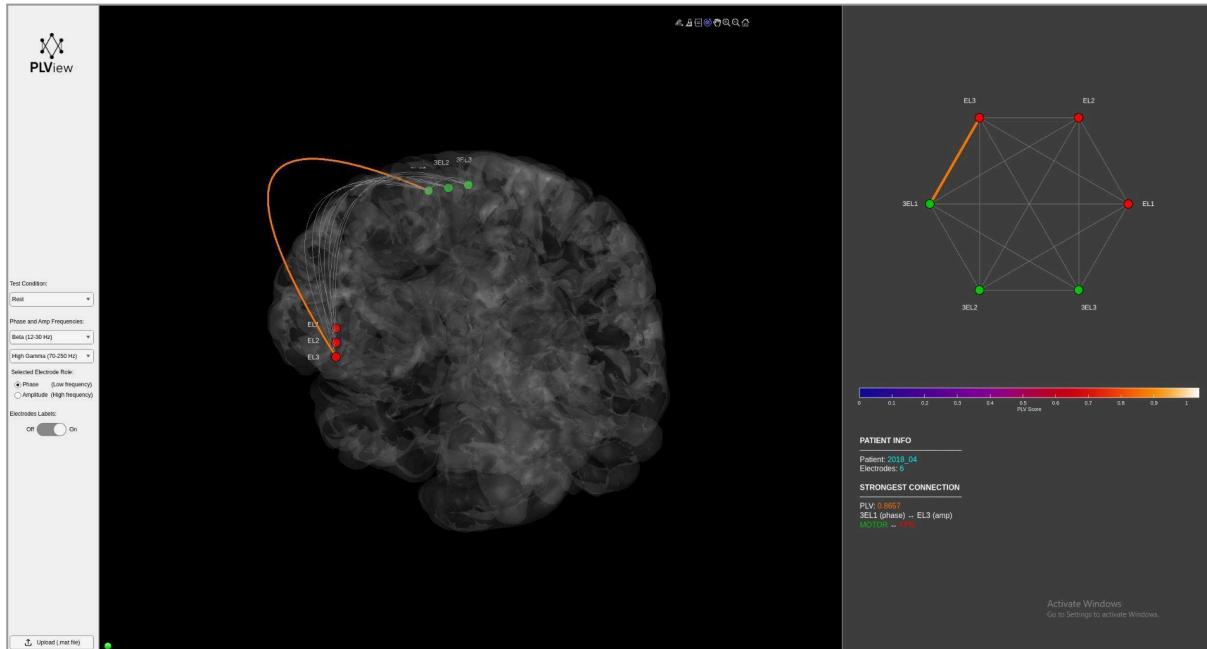
3. מה אנחנו מצלפים לראות

בהתאם לממצאי המאמר של¹ Assem & Erez (2023), פעילות מוחית בתדר גבוהה (High Gamma) משמשת להבנה בין אזורי שליטה קוגניטיבית באונה הפרונטלית לבין רשתות סמוכות במוח. פעילות זו משקפת את המעורבות של הרשת הפרונטו-פריאטלית (FPN) והציפייה היא לראות ערכי PLV גבוהים יותר בתוך רשת ה-FPN בזמן שימוש הדורשות שליטה קוגניטיבית (משימה קשה - Hard) - ספירה של אותיות ומספרים לעומת מצב מנוחה (Rest), וכן ירידת הקשריות בין רשתות שונות (כגון FPN ו-DMN) כאשר מתבצעת משימה הדורשת מיקוד גבוה. (משימה קלה - Easy) - ספירה פשוטה של מספרים) דפוא זה צפוי לשקוף את הארגון הפנימי של רשתות השליטה במוח ואת יכולתן לתאמם פעילות לצורכי ביצוע יעיל של תפקודים ניהוליים.

5. השיטה הנבחרת

במהלך לחישוב הקישוריות בין זוגות האלקטרודות, ובמטרה להצליח להבין טוב יותר את ארגון הרשתות, יצרתי כלי להציג גרפית של נתוני המעבדה (הקלוטות ECoG), וניתוח הקישוריות בין רשתות המוח השונות. (**חישוב הPLV בין אלקטרודות**)

הכלי נבנה בעזרת סביבת העבודה **MATLAB** **App Designer**⁷ הייעודית של MATLAB, ונitin להתקינה בכל תוכנת MATLAB דרך **File Exchange**⁶ של App Store של MATLAB's File Exchange מטלב)



*צילום מסך מתוך הכלי PLView לאחר טעינת דאטא שהוקלט מ-6 אלקטרודות הממוקמות ב-2 רשתות שונות. (1).

הכלי קיבל את השם "PLView", שילוב של PLV ו-View:



*לוגו הכלי, עוצב על ידי. (Fig. 3).

הכלי מאפשר עיבוד וניתוח מידע גולמי של המעבדה בפורמט mat., הציגו בצורה גרפית (דו ותלת מימדית), ושינוי פרמטרי הניתוח ע"י תפריט בחירה פשוט ומונגןש.

החלון הראשי מחולק ל-3 עמודות:

עמודה (פאנל) שמאלית - תפריט שליטה על פרמטרי הניתוח.

מאפשר בחירה בין תנאי הניסוי (מנוחה, משימה קלה, או קשה), שליטה על טווח התדרים לסינון האותות (חלק מחישוב הPLV), בחירת תצוגה לאלקטרודה הנבחרת, הסתרה/הציגת של שמות האלקטרודות בגרפים, והעלאת קובץ מידע מסווג *mat*..

עמודה אמצעית - מודל תלת-ממדי, אינטראקטיבי, של המוח.

ניתן לסובב את המודל ב 360 מעלות בצורה חופשית עם העכבר. (drag and drop) המשיק המובנה של מערכת הצירם בمطلوب מופיע בצורה אוטומטית (בפינה הימנית עליונה) כאשר העכבר מרוחף (hover) מעל המודל ומאפשר ליצא תמונה שלו בקלות, ובכל פורמט. המודל מציג את מיקומי האלקטרודות על גבי המוח, ואת עצמת הקישוריות ביניהן. (לפי ערכי הPLV) כבירות מחדר, אחרי שעיבוד המידע הושלם בהצלחה ולפניהם בחירת אלקטרודה ספציפית, יוצג החיבור הcy חזק (ערך הPLV הגבוה ביותר) של המטופל.

בעת בחירת אלקטרודה ספציפית, יעדכן המודל וייצגו עצמות הקישוריות בין האלקטרודה הנבחרת לכל האלקטרודות האחרות בgraf.

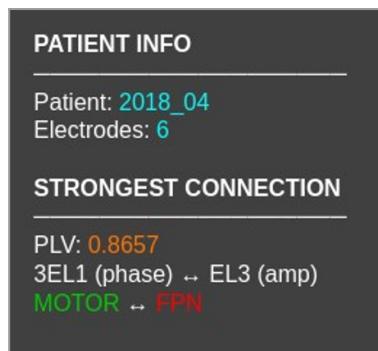
המשתמש יוכל לקבוע האם האלקטרודה הנבחרת תחשב לפ' *amp* / *phase* באמצעות תפריט הרדיו בפאנל השמאלי.

צבעי האלקטרודות השונים מעידים על הרשות המוחית עליה הן ממוקמות.

עמודה ימנית - גраф דו-ממדי שלם⁸, המציג את הקישוריות בין כל האלקטרודות, ופאנל מידע טקסטואלי.

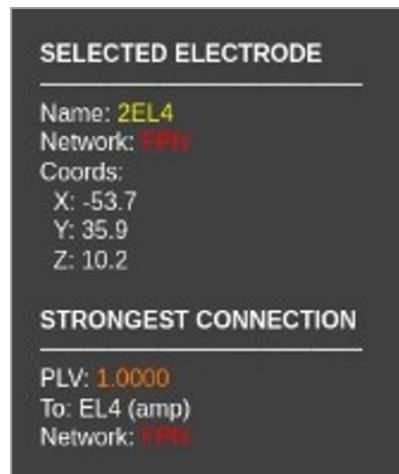
העמודה הימנית מוצגת רק לאחר שעיבוד המידע הושלם בהצלחה. היא מציגה גраф דו-ממדי המציג את כל האלקטרודות ואת עצמת הקישוריות ביניהן בצורה שטוחה וקלה להבנה. המשיק המובנה של מערכת הצירם בمطلوب מופיע בצורה אוטומטית כאשר העכבר מרוחף (hover) מעל הgraf ומאפשר ליצא תמונה שלו בקלות, ובכל פורמט.

בנוסף, מופיע תחתיו מקרה צבעים (colorbar) המשמש לניתוח ערכי הPLV בgraf. תחתיו מופיע פאנל מידע טקסטואלי המתעדכן לפי האלקטרודה הנבחרת. במצב ברירת המחדל, לפני שנבחרת אלקטרודה ספציפית, הוא מציג מידע כללי את: "שם" המטופל (מלך שם קובץ המידע), מספר האלקטרודות, החיבור הcy חזק (ערך הPLV הגבוה ביותר), בין אילו אלקטרודות הוא מתרחש ולפי איזה תדריות סוננו, ואת הרשותות בהן נמצאות.



*פאנל מידע כללי, לפני שנבחרת אלקטרודה ספציפית. (Fig.4)

לאחר לחיצה של המשתמש על אלקטרודה ספציפית בגרף הדו-מימדי, פאנל המידע מתעדכן ומוצג המידע עלייה, כגון: שם האלקטרודה, הרשת בה היא נמצאת, קורדינטות תלת-מימדיות בפורמט NIM, החיבור הכי חזק שלה (לפי ערך הPLV), לאיזו אלקטרודה, באיזה תדריות היא מסוננת והרשת בה היא נמצאת.



*פאנל מידע של אלקטרודה נבחרת. (Fig.5)

אצ"נ כי Fig.5 נלקח מתוך הרצה על דатаה רנדומלי שנוצר רק בשbill לבודוק את הכל. כמובן שערך הPLV של דטה ארגני לא אמרור להגיא לערך 1. (הסביר מרווח וניתוח עמוק בפרק 6.)

הסיבות לבחירה

היתרונות של הצגה גרפית של נתוני המעבדה הם רבים, וביניהם: יכולת לזהות דפוסים, מגמות וחירוגות שקשה או בלתי אפשרי לגלוות בעת בחינת מידע גלומי, יכולת לצפות בשינויים לאורך זמן, להשוות בין מצבים שונים, לזהות קוורלציות בין משתנים רבים, מסיע בהחשת אזורים המוח הפעילים, עוצמת הפעולות ואופן התקשרות בין אזורים שונים, הנגשה של ממצאים מדעיים מורכבים גם לקהלים שאינם מומחים, ומקלה על קבלת החלטות מושכלות בסביבות קליניות ומחקריות.

הסיבה שבחרנו בניתוח הקישוריות בין הרשותות ע"י חישוב ערך ה **Phase Locking Value** (PLV) היא משומם שהוא מספק מדד ישיר, פשוט וברור של סynchron בין אלקטרודות - מדד שנמצא

רגיש לשינויים בזמן-אמת ומתאים במיוחד לנוטרי ECoG בעלי רזולוציית זמן גבוהה. בנוסף, PLV מאפשר ניתוח ממוקד של תדרים ספציפיים וקל לשילוב באופן יעיל וברור ויזואלית על גבי המודל התלת-ממדי (ערך סקלרי בלבד), בניגוד לממדים מורכבים יותר הדורשים חישוב או פרשנות סטטיסטיות מתקדמת.

לבחירה **MATLAB** יש מספר סיבות.

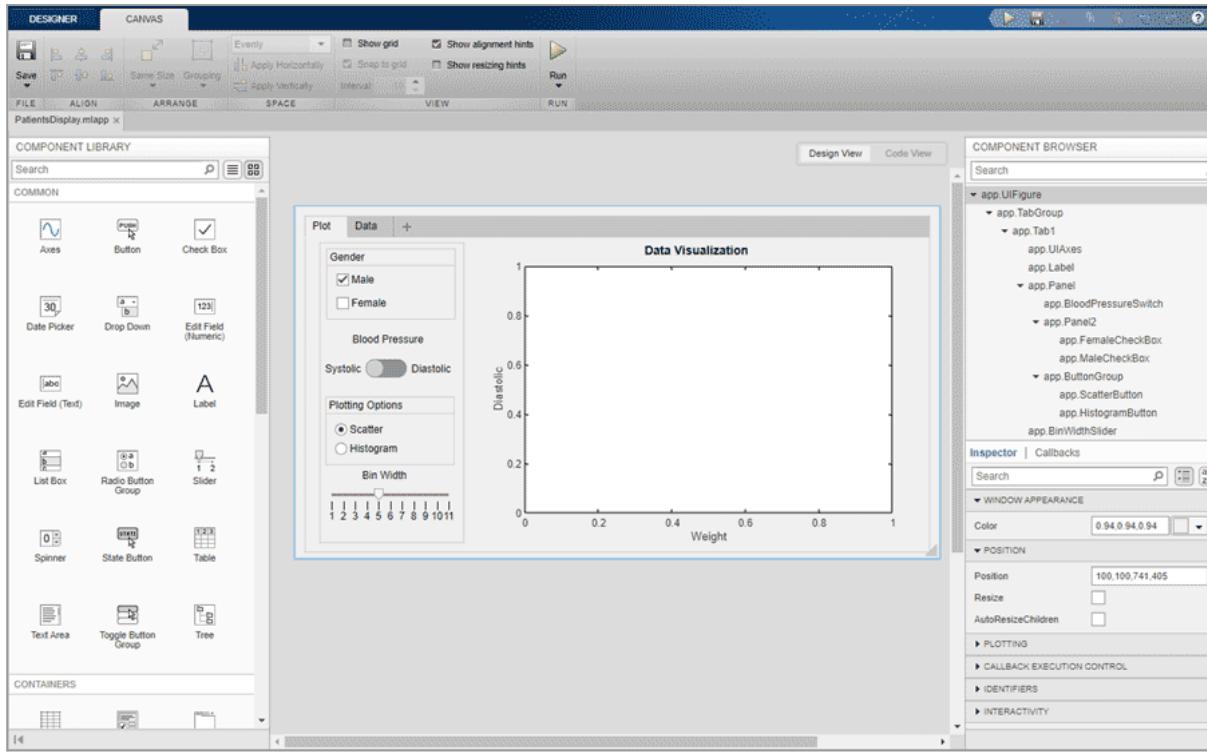
ראשית MATLAB מאפשרת עיבוד אוטומטי מתקדם בעזרת ספריות מובנות (כמו Signal Processing Toolbox) ומותאמת במיוחד לעבודה עם מטריצות, כנדרש בפרויקט זה. מטריצת PLV מכילה את ערכי הקישוריות בין כל זוגות האלקטרודות, והעבודה איתה ב-MATLAB היא טבעיות וקלה. הת לחבר פשוט וברור לאינדוקס של מטריצות, הביצועים מעולים למטריצות גדולות בזכות שימוש בספריות מהירות מאוד, והשפה יכולה לבנייה סבב החשיבה המטריציונית. זה עשו את הקוד קרייא יותר ופחותழות מועעד לטיעוות בהשוואה לשפות אחרות.

בנוסף, MATLAB מצטיינת במיוחד בייזואלייזציה תלת-מימדית ומאפשרת לרנדר משטחים תלת-ממדיים עם הגדרות תאורה מתקדמת, שקיפות, ויבוב חופשי בעכבר - הכל ללא צורך בהבנה עמוקה של גרפיקה ממוחשבת, ולא ספריות חיצוניות מורכבות כמו בשפות אחרות.

יתרונו נוסף הוא שהרבה מעבדות בתחום הביוירולוגיה, כולל המעבדה של ד"ר יערה ארץ, משתמשות ב-MATLAB כסטנדרט ויש תאיימות מובנית לפורמטים רפואיים (EEG, ECoG,...) כמו קבצי mat... לבסוף, אם היינו בונים את הפרויקט הזה ב-Python, היינו צריכים להתקין ולהגדיר מספר רב של ספריות חיצונית כמו numpy, scipy, matplotlib, PyQt, ו-Three.js. והוא יתאפשר יוטר מרכיבת ודורשת שימוש בכלים כמו VTK או Mayavi. ב-WebGL ופיתוח web, היה כמעט בלתי אפשרי לבצע את העיבוד המתמטי המורכב באופן יעיל, והיינו צריכים ללמוד להשתמש ב-WebGL ו-Three.js רק בשביב המודל התלת-מימדי. לעומת זאת, MATLAB מספק את כל הכלים האלו מובנים, משולבים, וモתאמים בדיקן לסוג העבודה הזה.

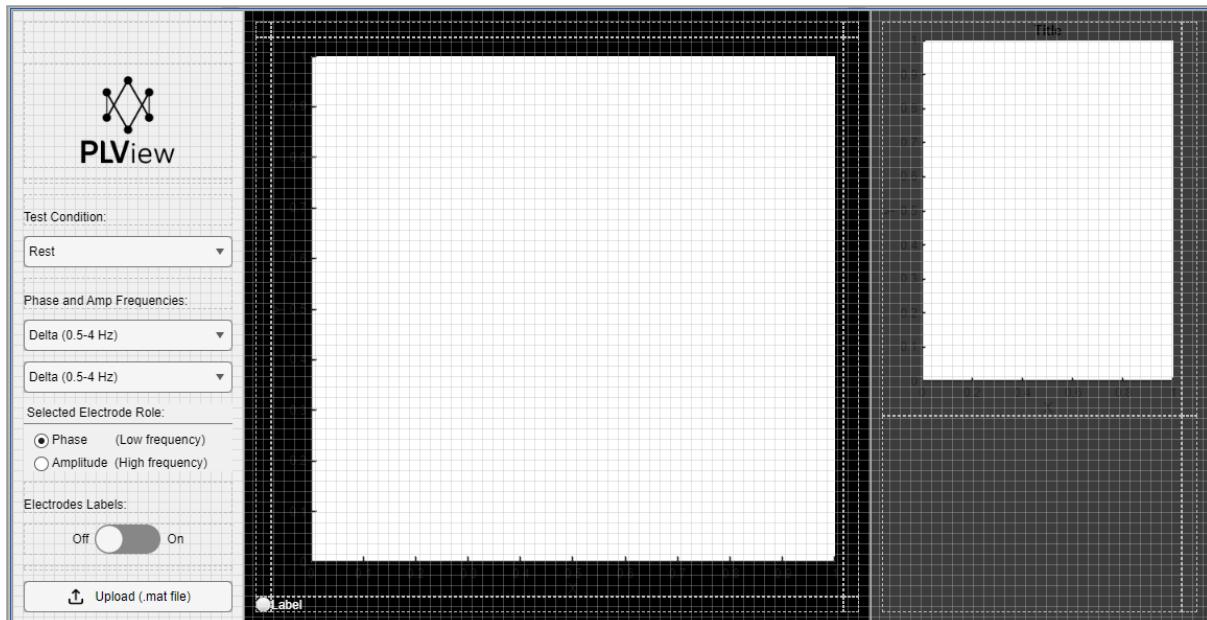
הסיבה לבחירה בסביבה **App Designer** היא קלות עיצוב המשקגרי, וביצוע שלב העיבוד המוקדים (Pre-processing) של המידע הגלומי באופן מובנה בתוך הכליל, בעת טעינת המידע. App Designer מאפשר לבנות משקՄ שמשתמש גרפי מלא ללא צורך בכתיבה קוד HTML.

סביבת עבודה זו מאפשרת עבודה במקביל ב-2 חיצות: משק גרפי, וקוד. עיצוב המשק הגרפי, ב-"View", "Design", נעשית ע"י גירירה של אובייקטים (Components) מוכנים מראש אל תוך חלון התוכנה, ומיקוםם ב"יד חופשית". כל אובייקט מקבל שם משתנה גלובלי ייחודי אליו אפשר לפנות בחלק הקוד. (לדוגמה: App.objectName)



(Fig.6) - חלונית הממשק הגרפי של ⁷ .MATLAB App Designer - Design View*

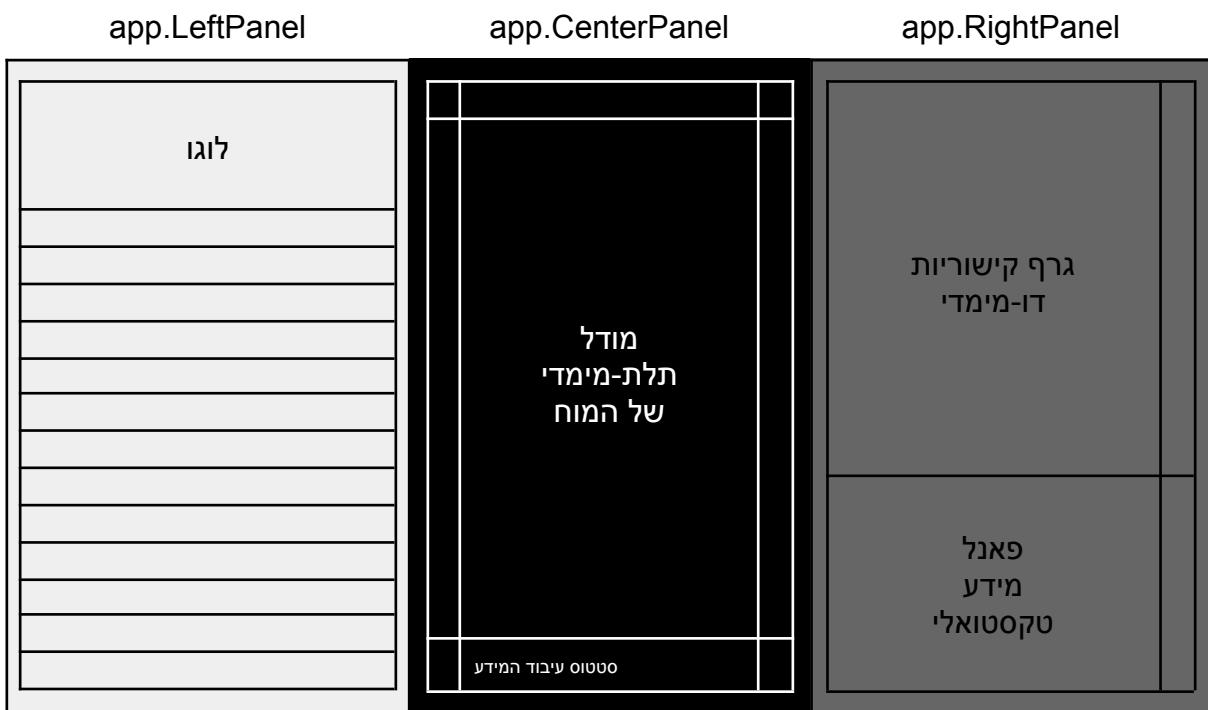
את חלון התוכנה אפשר לחלק ל"תאים" על ידי רשת (grid layout) אותה אפשר לחלק באופן דינامي (גודל יחסית לתאים אחרים, לדוגמה, תא אחד ברוחב x1 ותא שני ברוחב x0.5 ביחס אליו) או ידני. (רוחב/אורך קבועים לפי גודל בפיקסלים) זה מאפשר לייצר חלון רספונסיבי, המשנה את גודלו באופן דינמי למצוות הלקוח או גודל המשתמש, ושומר על היחסים בין האובייקטים המוצגים בו.



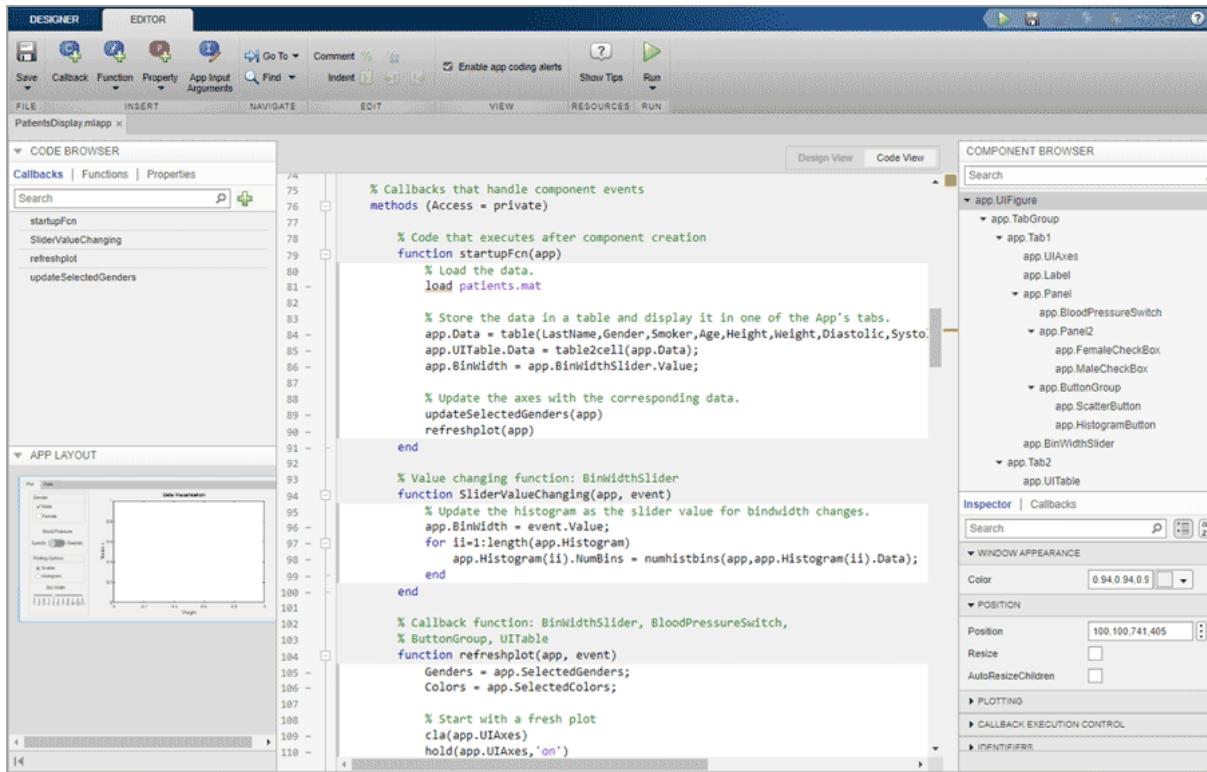
(Fig.7) .App Designer של הכלי PLView בממשק ה Design View *

בכדי לבנות את הכלי לא מספיק (או אפשרי) רק לכתוב את כל הקוד ב "Code View" מכיוון שיש מקטיעי קוד נעלים שסבירת העבודה לא מאפשרת לשכתב, אז תחילה נדרש לבנות את הממשק הגרפי ידנית.

תרשים כללי של מספר השורות והעמודות ב `grid layout` ויחסו הגודלים בתוכם (בקירוב):



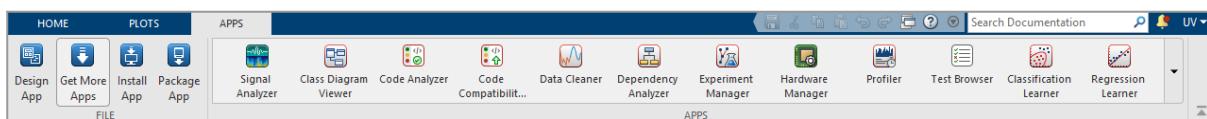
כתיבת הקוד המשלים את הממשק הגרפי נעשית בחלונית ה "Code View".
שם ניתן לכתוב קוד המתממשק ישירות עם האובייקטים הגרפיים ומעבד את המידע ברקע על ידי
 כתיבת פונקציות ומשתנים בשפת MATLAB.
צורת הפניה לאובייקטים השונים ולהגדրותיהם נעשית בצורה פשוטה להבנה ואינטואיטיבית
שחוזרת על עצמה ברוב המקרים.
בנוסף, ישנו חלקים בקוד התכנית שאינם ניתנים לעריכה חופשית, מה שעזר לשמר על "גבולות
גזרה" ברורים, ולהמנع מביעות בהרצה.



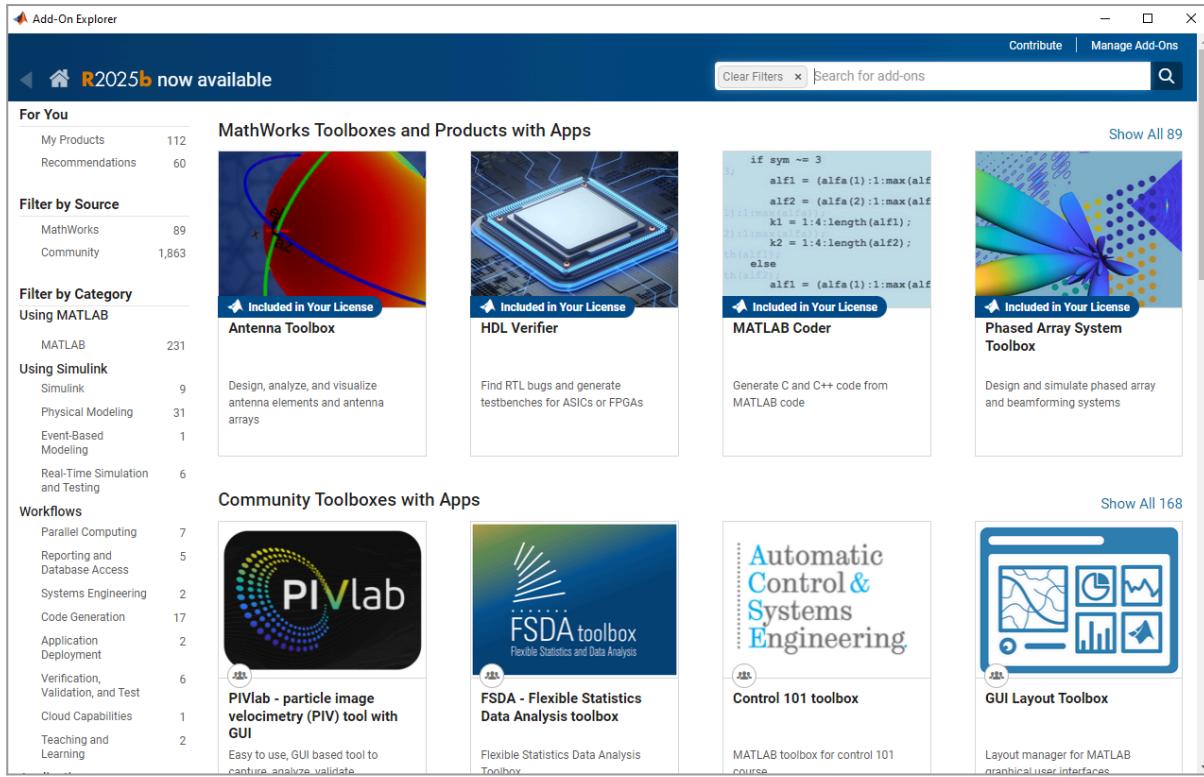
(Fig.8) .MATLAB App Designer לכתיבת קוד של⁷ . - Code View*

בנוסף, סביבת ה App Designer כוללת את פלטפורמת שיתוף הקבצים (MATLAB's File Exchange⁶) המובנית של MATLAB המאפשרת שיתוף קל של אפליקציות, ניהול גרסאות, והתקנות.

זה הקל מאד על העבודה על הכלי, שיכלה להתבצע מכל מחשב שהוא ולבסוף כל שנדרש הוא להכנס למילב דרך מחשב הענן של המעבדה ולהתקין אותה דרך תפריט ה Add-On Explorer. אופצייה נוספת לשיתוף הכלי היא האפשרות ב MATLAB ליצא את האפליקציה כקובץ הפעלה עצמאי שיכל לרוץ על כל מחשב גם ללא התקנת MATLAB. (בעזרת MATLAB Compiler).



*הגישה לדרך Add-On Explorer ול App Designer



***פלטפורמת ה

עדיין
"** בוסףת העליין. (Fig.10) MATLAB's File Exchange

***עמוד הכל' הפומבי ב** .MATLAB's File Exchange (Fig.11)

מגבלות

המגבלה העיקרית היא **הקוד הנעול**. סביבת העבודה App Designer נעלמת חלק מהקוד בפני ערכיה חופשית של המשתמש. דוגמה אחת שהקשטה על העבודה בפרויקט זה היא הגדרת פונקציות ה-**callback** שמתחרבות לכפתורי הממשק. במקרה להגדיר פונקציה מסווג זה יש להוסיף אותה דרך סביבת העבודה הויזואלית "Design View" בתפריט ההגדרות ימני, ורק אז הפונקציה נוספת לקוד במקום המיועד לכך וניתן לעורר רק את תוכנה בלבד, ללא אפשרות לעורר את שם הפונקציה או הקלט שלה.

חיסרון נוסף הוא **שינוי דברים בין גירסאות**, ולפעמים קוד שעבד בגירסה אחת לא עובד בגירסה אחרת. הם מפסיקים תמיינה בפונקציות ישנות, משנים התנהגות ושמות משתנים של פונקציות קיימות, או מוסיפים דרישות חדשות. זה אומר שאם נחזיר את הפרויקט הזה בעודו כמה שנים עם גירסת MATLAB חדשה, הוא עשוי לא לעבוד ונאלץ להשקיע הרבה זמן על תיקונים. בנוסף, קשה לשתף פרויקט בין אנשים שיש להם גירסאות שונות של MATLAB. (*פרויקט זה נבנה בMATLAB בגרסת b R2024b התואמת את הגירסה המותקנת בענן המעבדה, ומותאם לעבוד גם עם גירסה b (R2022b)

מגבלה נוספת היא **העיצוב המיוון של האובייקטים המובנים**, כמו כפתורים ותפריטים, וחוסר יכולת למקם אובייקטים מחוץ לתא ה-`layout` grid. על אף שניתן לשנות חלק מההגדרות הגרפיות של האובייקטים השליטה עדין מוגבלת ולא חופשית לגמרי. לדוגמה, לא ניתן להשפיע על שיקיפות של אובייקטים, או להעניק לאובייקטים אפקט צל או תוארה חיצונית. כמו כן, על כל אובייקט לשבתetur תא אחד `layout` grid שמאגדים. אובייקט לא יכול לשבת על שטח של 2 תאים, או לחרוג מאיזור התא המיועד לו.

חיסרון נוסף ממשמעותו של MATLAB הוא **ההရישון יחסית יקר**. זה מגביל את השימוש של חוקרים עצמאיים ומיעיד את השימוש בכלים למעבדות ממושדות.

ולבסוף, **הקוד של הסि�פירות המובנות/חיצונית סגור**. אזי אם מתרחשת שגיאת חישוב בתוך פונקציה השיכת לסיפריה תחילך הדיבאג (debugging) יכול להיות מסובך.

6. בדיקת השיטה, תוצאות וניתוח

לבדיקה השיטה נשתמש במקרה בוחן של כמה מטופלים שונים מהמעבדה:

שם המטופל	מספר אלקטroduות	מספר רשות
2018_01	3	3
2018_08	5	3
2018_07	9	3

הגדרות התפריט עברו כל המטופלים:

Test Condition	Frequency #1	Frequency #2	Electrodes Labels
Rest / Easy Task / Hard Task	Beta (12-30 Hz)	High Gamma (70-250 Hz)	On

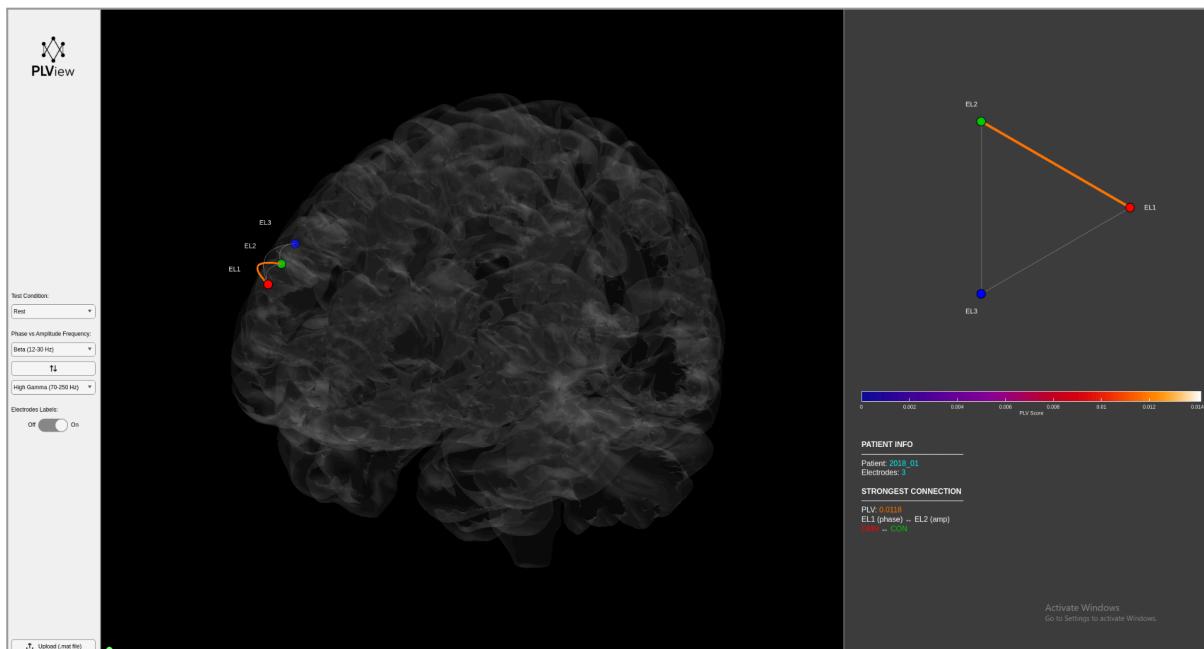
כפי שהוסבר קודם, נצפה לראות:

- ערכי VLP גבוהים יותר בתוך רשת ה- NPN בזמן משימות הדורשות שליטה קוגניטיבית (משימה קשה (Hard) - ספירה של אותיות ומספרים) לעומת מצב מנוחה (Rest)
- וכן ירידה בקשרויות בין רשותות שונות (כגון FPN ו- DMN) כאשר מתבצעת משימה הדורשת מיקוד גבוה. (משימה קלה (Easy) - ספירה פשוטה של מספרים)

(המשך בעמוד הבא...)

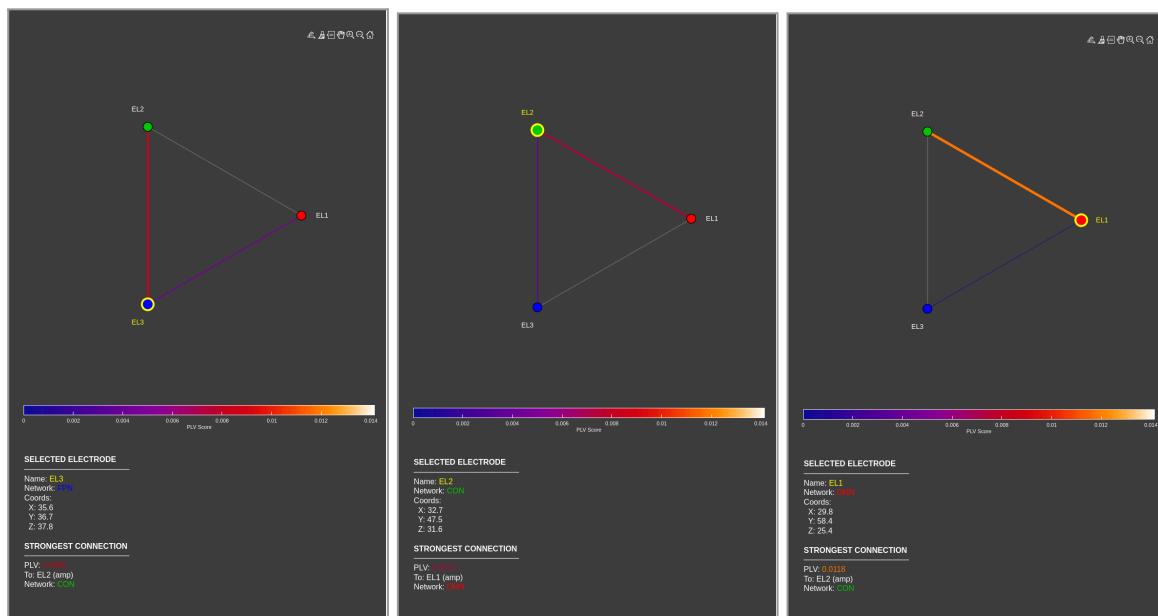
תוצאות לאחר טעינת הקובץ של מטופל 01 2018

Test Condition: Rest



(Fig.12)

תצוגה דו-מימדית עבור כל אלקטרודה: (פאנל ימני)

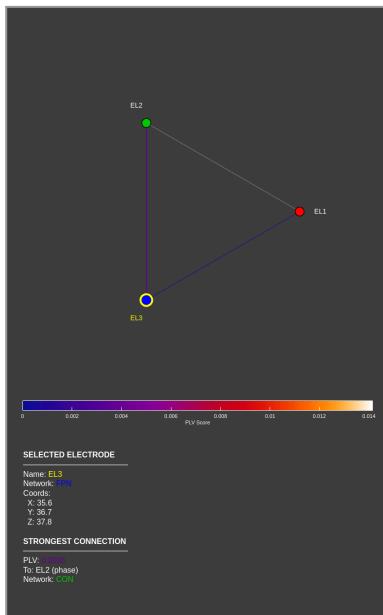


(Fig.15)

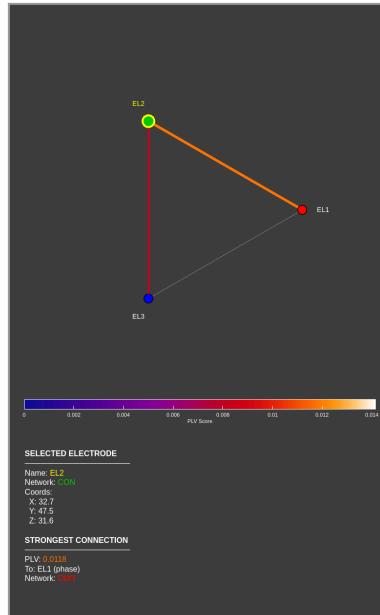
(Fig.14)

(Fig.13)

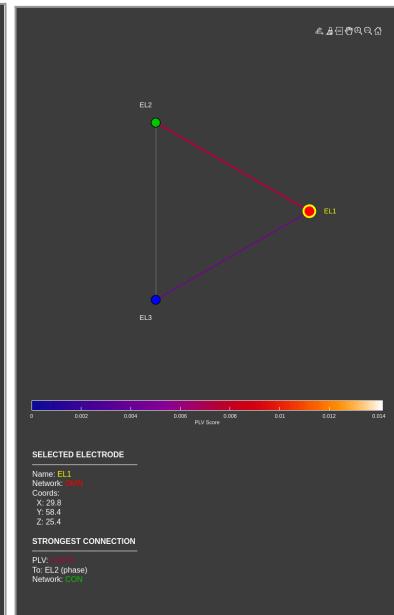
לאחר היפוך תפקדים: (amp של האלקטרודה הנבחרת וכל השאר phase)



(Fig. 18)

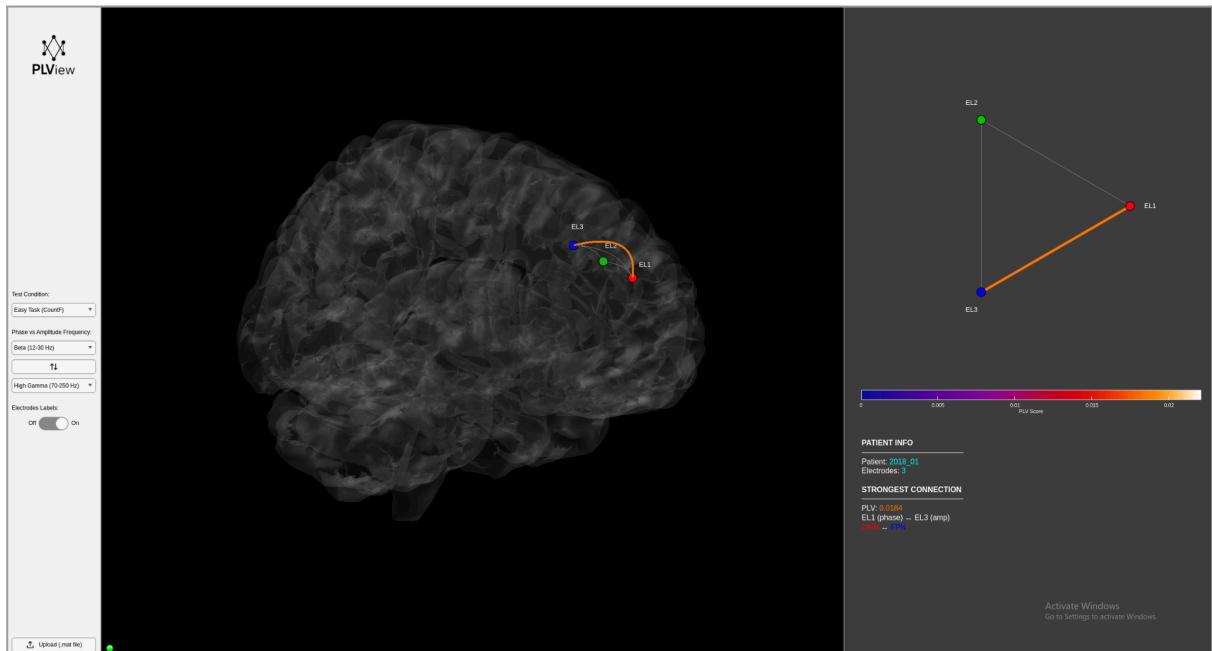


(Fig. 17)



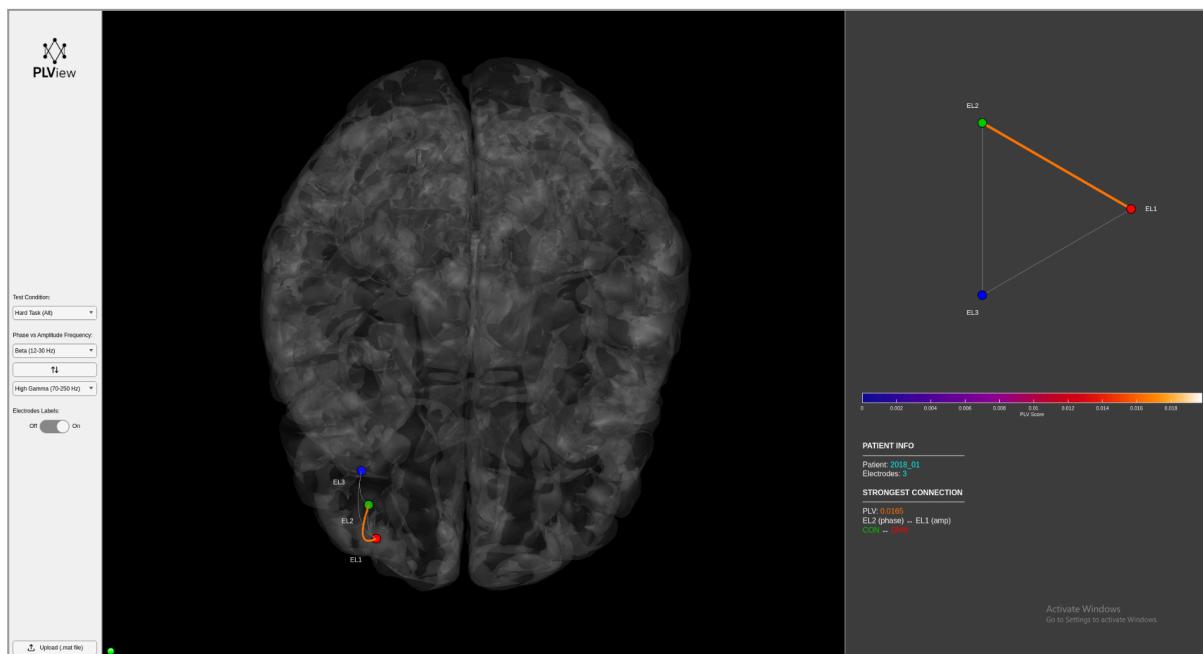
(Fig. 16)

Test Condition: Easy Task



(Fig. 19)

Test Condition: Hard Task



(Fig.20)

ערכי החיבורים החזקים ביותר עברו כל אלקטודה בכל מצב ניסוי:

האלקטודה הנבחרת	הרטה שלה	האלקטודה ה"מיושרת"	הרטה שלה	ערך PLV
Rest				
EL1 (phase)	EL2 (amp)	DMN	CON	0.0118
EL2 (phase)	EL1 (amp)	CON	EL2 (phase)	0.0074
EL3 (phase)	EL2 (amp)	FPN	CON	0.0085
EL1 (amp)	EL2 (phase)	DMN	CON	0.0074
EL2 (amp)	EL1 (phase)	CON	EL2 (phase)	0.0118
EL3 (amp)	EL2 (phase)	FPN	CON	0.0038
Easy Task				
EL1 (phase)	EL3 (amp)	DMN	FPN	0.0184
EL2 (phase)	EL1 (amp)	CON	DMN	0.0182
EL3 (phase)	EL2 (amp)	FPN	CON	0.0058
EL1 (amp)	EL2 (phase)	DMN	CON	0.0182

0.0058	FPN	EL3 (phase)	CON	EL2 (amp)
0.0184	DMN	EL1 (phase)	FPN	EL3 (amp)
Hard Task				
0.0108	FPN	EL3 (amp)	DMN	EL1 (phase)
0.0165	DMN	EL1 (amp)	CON	EL2 (phase)
0.0063	DMN	EL1 (amp)	FPN	EL3 (phase)
0.0165	CON	EL2 (phase)	DMN	EL1 (amp)
0.0092	DMN	EL1 (phase)	CON	EL2 (amp)
0.0108	DMN	EL1 (phase)	FPN	EL3 (amp)

ניתוח תוצאות מטופל 01_2018:

מכיוון שבמטופל זה ישן רק 3 אלקטטרודות רלוונטיות (על רקמה בריאה) המונחות על 3 רשתות שונות, אי אפשר להסיק לגבי קישוריות פנימית בתחום כל רשת, אך כן ניתן להשוות בין תנאי הניסוי השוניים.

ניתן לראות כי בתנאי המנוחה (Rest) - ערכי הקישוריות הם הנמוכים ביותר. כמו כן, הקישוריות בין רשתות DMN ↔ CON חזקות מהקישוריות לרשת ה-FPN. זה מחזק את הטענה שבעת מנוחה אין שליטה קוגניטיבית פעילה ולכן ה-FPN כמעט ואינה שולטת בקישוריות. בנוסף, הרשת הפעילה ביותר כאן היא DMN (Default Mode Network) הידועה כפעילה במנוחה ובמצבים של עיבוד פנימי.

בתנאי המשימה הקללה (Easy) - הדרשות מיקוד גבואה, נראה עלייה משמעותית בסyncron של FPN, והקשרים הבולטים הם בין DMN ↔ CON ו- FPN ↔ CON.

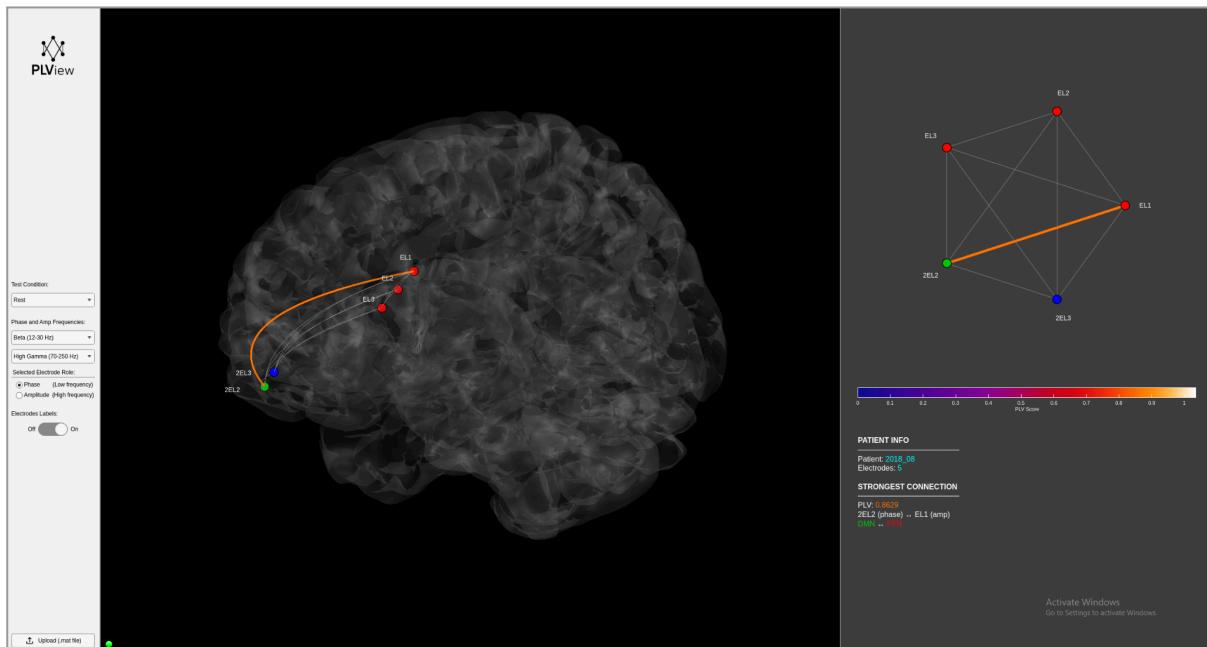
בתנאי המשימה הקשה (Hard) - לכואורה הינו מצפים לראות עלייה חזקה עוד יותר של FPN ושקיעה של DMN - אך יתכן שעקב מספר האלקטרודות המוגבל (3), לא נלכדה מלאה הפעולות הפרונטליות.

התוצאות מצביעות על מעורבות רחבה יותר של רשתות מרובות בזמן המשימה הקשה - דפו שתוואם את ההנחה שהמאיץ הקוגניטיבי דורש תיאום בין רשתות ולא רק הפעלה של FPN.

לסיכום, השינויים בערכי ה-PLV מדגימים את המעבר מדומיננטיות של DMN במנוחה לשוליטה של FPN ביצוע משימות - בדוק כפי שמתואר במאמר.

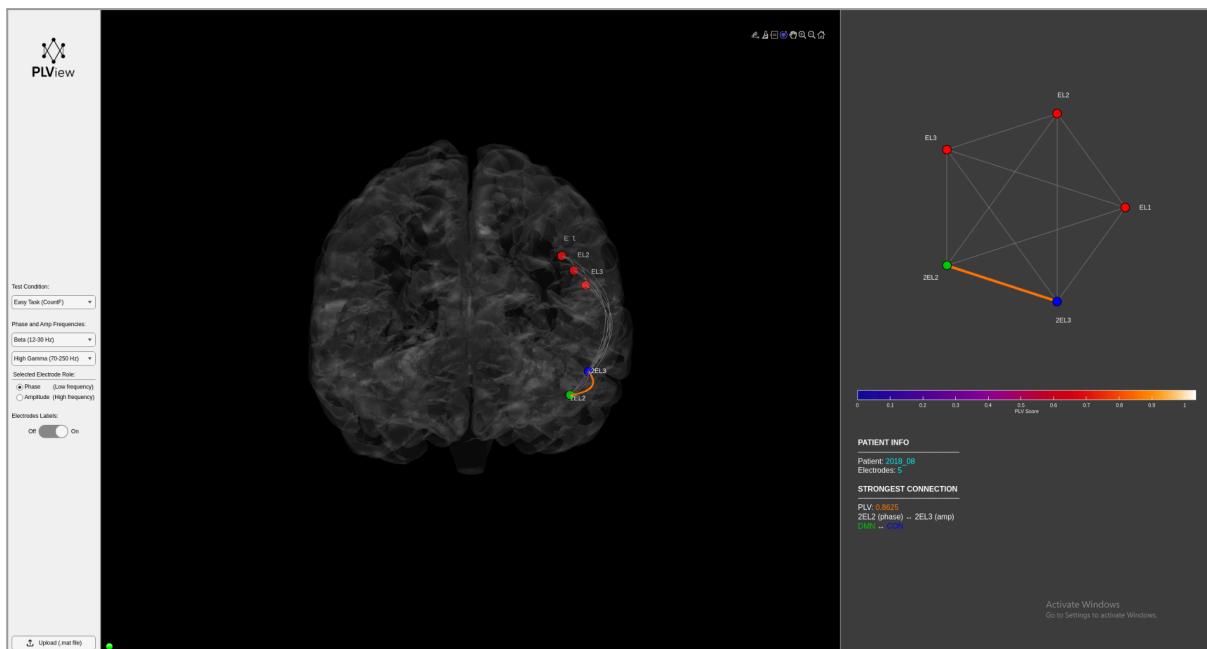
תוצאות לאחר טעינת הקובץ של מטופל 08_2018

Test Condition: Rest



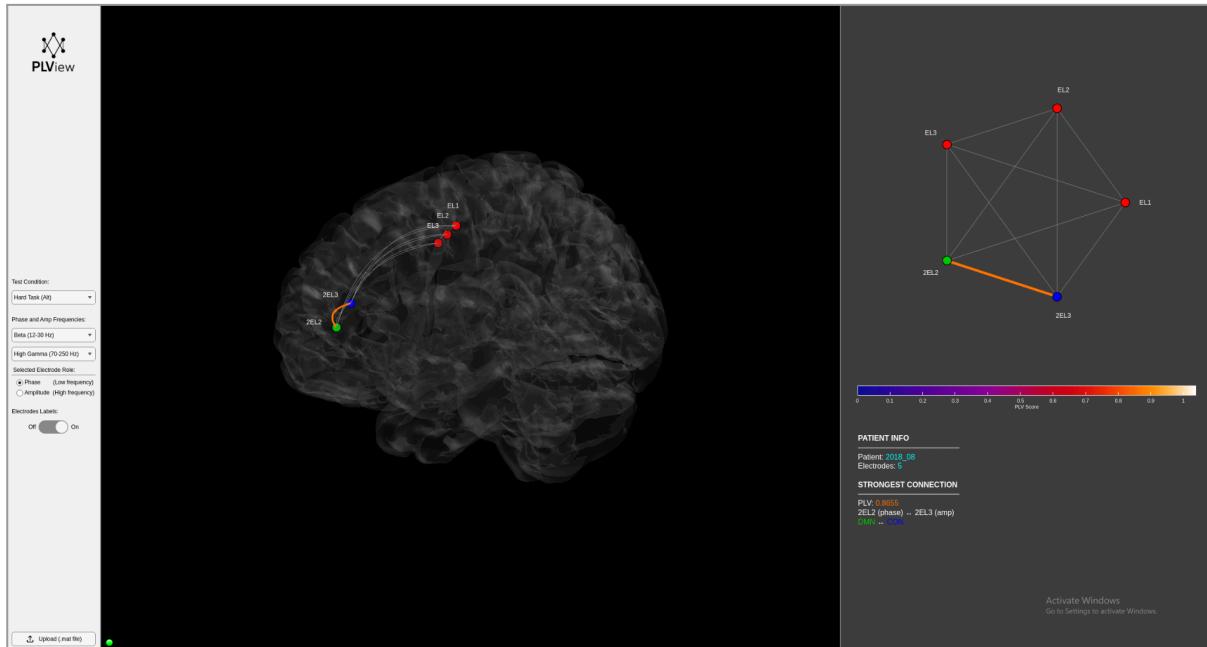
(Fig.21)

Test Condition: Easy Task



(Fig.22)

Test Condition: Hard Task



(Fig.23)

ניתוח תוצאות מטופל 2018_08:

בתנאי המנוחה (Rest) - הפעולות הדומיננטיות במנוחה היא של DMN, רשת ברירת המחדל, המראה קישוריות גבוהה. לעומת FPN ו-CON מראות סנכרון חלש, מה שמעיד על חוסר מעורבות של מנגמוני שליטה קוגניטיבית. התוצאה הזאת משקפת את המודל של Assem & Erez (2023) - במנוחה, האזוריים הקשורים לשליתה קוגניטיבית אינם פעילים באופן מתואם, בעוד ש-DMN פעילה ומקורתה חזק.

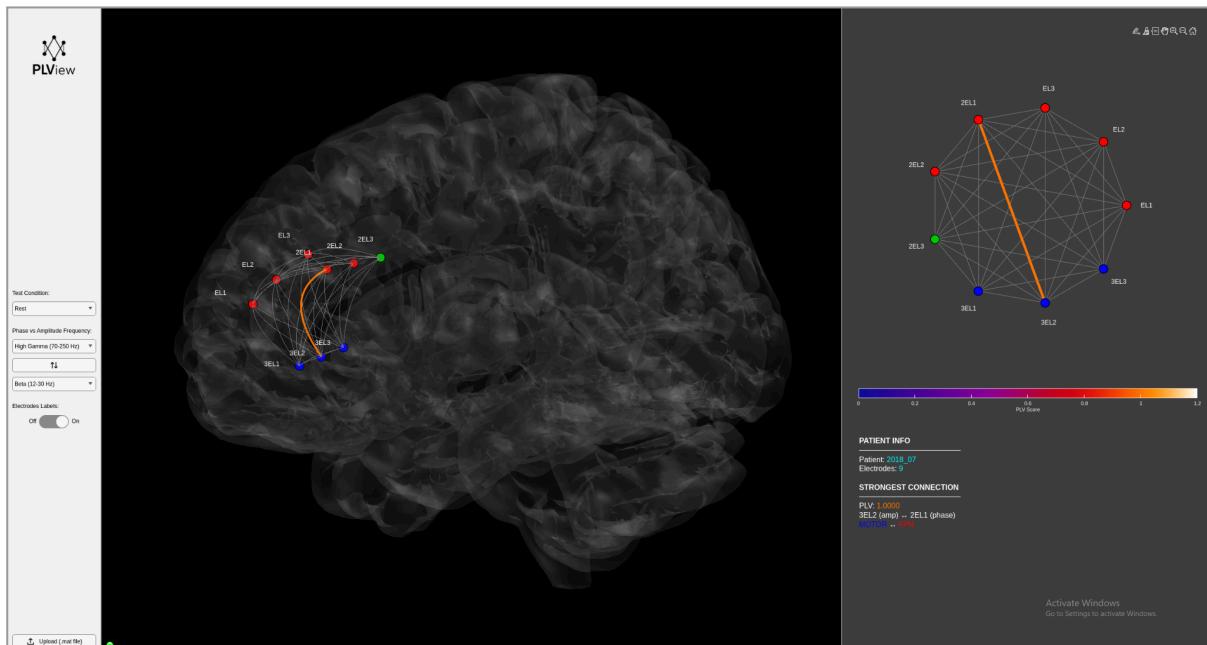
בתנאי המשימה הקללה (Easy) - המעבר ממשימת מנוחה למשימה קללהגורם להפעלה ראשונית של FPN ולתיאום עם DMN, כנראה לצורך מיוקוד קשב. הדפוס הזה מצביע על גיוס הדרגי של רשתות שליטה קוגניטיביות, תחילה שתואר במאמר כעליה הדרגתית בפעולות (High Gamma) באזורי הפונטליים.

בתנאי המשימה הקשה (Hard) - ערכי PLV בינוניים (~0.02) בין NPN ↔ FPN, המעידים על סנכרון מוגבר בתווך הרשת. ניכרת קישוריות גבוהה מאוד (~0.86) בין CON ↔ DMN ו- NPN ↔ DMN. ככלומר, בזמן העלייה בעומס הקוגניטיבי, כל שלוש הרשתות משתתפות, עם סנכרון גבוה גם בין רשתות שונות ולא רק בתווך כל אחת מהן.

כלומר, ממד ה-PLV מספק הוכחה חשובה לכך שהמוח משנה את דפוסי הסנכרון בין הרשתות בהתאם לעומס הקוגניטיבי - כפי שהמאמר הראה באמצעות מדדי High Gamma.

תוצאות לאחר טעינת הקובץ של מטופל 2018_07

Test Condition: Rest



(Fig.24)

ערći החיבורים החזקים ביותר עברו כל האלקטרודה:

ערך PLV	הרשת שלה	האלקטרודה הכי "מיושרת"	הרשת שלה	האלקטרודה הנבחרת
Rest				
0.0019	FPN	2EL2 (amp)	FPN	EL1 (phase)
0.0024	FPN	EL3 (amp)	FPN	EL2 (phase)
0.0027	MOTOR	3EL1 (amp)	FPN	EL3 (phase)
1	MOTOR	3EL2 (amp)	FPN	2EL1 (phase)
0.0019	FPN	EL2 (amp)	FPN	2EL2 (phase)
0.0020	FPN	EL2 (amp)	DMN	2EL3 (phase)
0.0018	FPN	2EL2 (amp)	MOTOR	3EL1 (phase)
1	FPN	2EL1 (amp)	MOTOR	3EL2 (phase)
0.0025	FPN	EL3 (amp)	MOTOR	3EL3 (phase)

0.8322	FPN	2EL1 (phase)	FPN	EL1 (amp)
0.8150	FPN	2EL1 (phase)	FPN	EL2 (amp)
0.8155	FPN	2EL1 (phase)	FPN	EL3 (amp)
1	MOTOR	3EL2 (phase)	FPN	2EL1 (amp)
0.8457	FPN	2EL1 (phase)	FPN	2EL2 (amp)
0.8433	FPN	2EL1 (phase)	DMN	2EL3 (amp)
0.8277	FPN	2EL1 (phase)	MOTOR	3EL1 (amp)
1	FPN	2EL1 (phase)	MOTOR	3EL2 (amp)
0.8352	FPN	2EL1 (phase)	MOTOR	3EL3 (amp)

ניתוח תוצאות מטופל 07_2018:

נחלק את התוצאות ל-3 קבוצות לפי ערכי PLV:

ערכים נמוכים (0.0018-0.0027), ערכים גבוהים (0.8150-0.8457), ושייכת (1).

ערכים נמוכים (0.0018-0.0027)

ערכים PLV נמוכים, הקרובים ל-0, מעידים על חוסר סינכרון מוחלט בין האלקטרודות. נשים לב למבחן המשותף בכל המקרים הללו - האלקטרודה הראשונה (phase) מסוננת לתדר Beta והשנייה (amp) ל-High Gamma, שתי האלקטרודות שייכות ל-2 סטריפים שונים, קלומרם, ממוקמות באיזורים יחסית רחוקים, ובנוסף, ברוב המקרים האלקטרודה המספקת את הפעזה (phase) שייכת לרשף הFPN.

המצאים מצביעים על כך שלא מתרחשת תקשורת מוחית בין איזורים אלה, וזה הגיוני מכיוון שבמצב מנוחה (Rest) אין משימה הדורשת תיאום בין הרשותות, אבל בתחום הרשות (FPN) אכן יש סינכרון יחסוי.

ערכים גבוהים (0.8150-0.8457)

ערכים PLV גבוהים, הקרובים ל-1, מעידים על סינכרון גבוה בין האלקטרודות. המבחן המשותף בקבוצה - האלקטרודה הראשונה (amp) מסוננת לתדר High Gamma והשנייה (phase) ל-Beta, ובכולם האלקטרודה אליה יש את הקישור החזק ביותר היא 2EL1, השיכת לרשף FPN. זה מצביע על כך שתדר ה-Beta של האיזור בו ממוקמת האלקטרודה 2EL1 משפייע על אמפליטודת ה-High Gamma של כל האלקטרודות האחרות. הסבר אפשרי אחד הוא שלאיזור המוח עליו מונחת האלקטרודה יש תפקיד מركצי בתיאום פעילות עצבית באיזורים אלה, אפילו במנוחה.

שגיאה (1)

ישן 4 שורות עם הערך $1 = PLV$, ובכלן מיצג הקישור בין 2 אלקטטרודות: 3EL2 (רשות MOTOR) ו- 2EL1 (רשות FPN).

משמעותו הערך 1 הוא **סינכרון מושלם** בין שתי האותות לאורך כל ההקלטה - ממצא לא סביר כמשמעותו במדעי ביולוגי, כמו זה הנמדד מהמוח, ולזה יכולם להיות מספר הסברים:

1. בעיה טכנית בהקלטה -

הסביר אפשרי אחד הוא ש-2 האלקטרודות הקליטו בדיק את אותן אותות. זה יכול לקרות אם 2 האלקטרודות נגעו או היו קרובות אחת לשניה, דבר שלפי המדע על מיקומי האלקטרודות נראה פחות סביר, מכיוון שהן ממוקמות על 2 רשתות שונות ומה שנראה מרחוק גדול יחסית.
(בנוסף, אפשרות זו נשלה ע"י בדינה ידנית של הדטה)

2. אלקטטרודת ייחוס משותפת -

אם 2 האלקטרודות חילקו את אותה אלקטטרודת ייחוס בתהיליך re-referencing אז הן יכולות להציג קורלציה מושלמת. במיוחד אם אלקטטרודת הייחוס הייתה יחסית רועשת/אקטיבית, זה יכול לייצר מתאם מלאכותי בשתי העורצים.

3. רעש חיצוני -

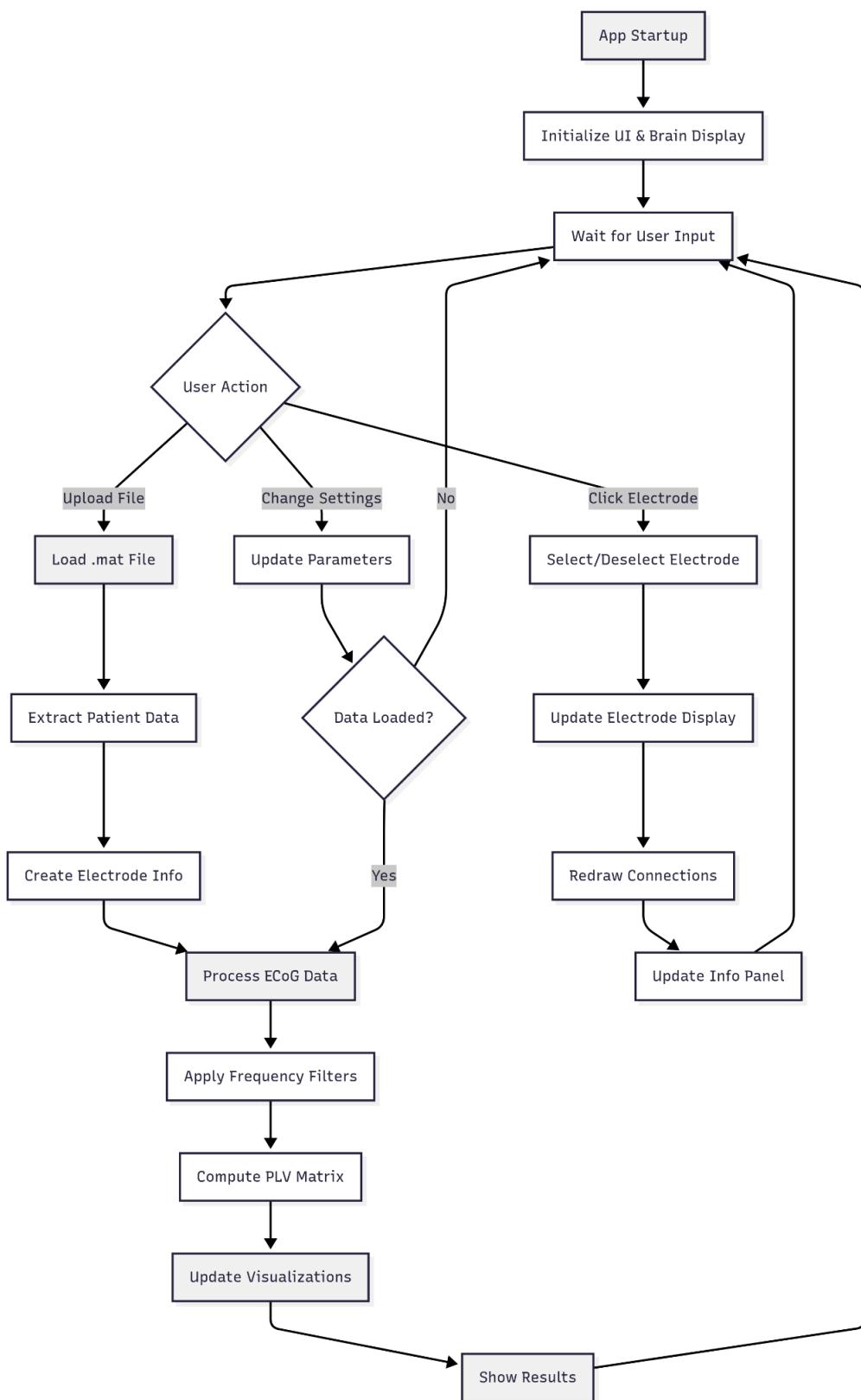
יתכן שרעש חיצוני השפיע על האלקטרודות האלו בצורה זהה.

4. בעיה בעיבוד הנתונים -

יש אפשרות שהבעיה נוצרה בשלב עיבוד המידע בכלל ובחירת העורצים הרלוונטיים מຕוך הקובץ. לדוגמה, שימוש במסנן bandpass חזק מדי יכול להוביל למataם מלאכותי. אופציה נוספת היא שהנתונים אחורי שלב הקיצוץ (trimming) היו קצרים מדי. (שלلت ע"י מעבר יידי על ההקלטות) ולבסוף, קיימת היכנות שהאלגוריתם יצר העתק של אחד העורצים האלה בטעות בעת סינון העורצים הרלוונטיים לפי ההוראות בקובץ המידע הסטטי של המעבדה. (Elecs_w_network.mat)

7. תרשימים מלכניים

(Fig.25) - **תרשימים זרימה לפי פעולות המשתמש** | User Flow



שלבי התרשים והפונקציות הראשיות המופעלות בכל שלב

1. App Startup - a. startupFcn(app)
2. Initialize UI & Brain Display - a. customLayoutControl(src, evt) b. customColormap() c. updateBrainModel() d. createInfoPanel()
3. Wait for User Input
4. User Action
5. Upload File
6. Load .mat File - a. UploadMATFileButtonPushed(event) b. setupLoadingAnimation()
7. Extract Patient Data - a. processDataWithAnimation(filePath, file)
8. Create Electrode Info - a. createElectrodeInfo() b. createNetworkColorMap()
9. Process ECoG Data - a. updateCurrentConditionData() b. processAndDisplayData()
10. Apply Frequency Filters
11. Compute PLV Matrix - a. processECOGData(ecogData, freqBand1Str, freqBand2Str) b. calculateStrongestConnections()
12. Update Visualizations - a. updateBrainModel()

b. updateCircularGraph()
c. plotConnectivityLines2D()
d. updateInfoPanel(selectedElectrodeIdx)
13. Show Results
14. Change Settings
15. Update Parameters -
a. ConditionDropDownValueChanged(event)
b. Freq1DropDownValueChanged(event)
c. Freq2DropDownValueChanged(event)
d. ElectrodeRoleChanged(event)
e. ShowElectrodeLabelsSwitchValueChanged(event)
16. Data Loaded?
17. Click Electrode
18. Select/Deselect Electrode -
a. onElectrodeClicked(src,~)
19. Update Electrode Display -
a. updateElectrodeAppearance()
20. Redraw Connections
21. Update Info Panel

שלבי התרשימים כולל הסבר על כל פונקציה

1. App Startup		
startupFcn(app)	הfonkzia:	
(...הרשות גישה כוללת לאפליקציה (כל הcptורים, הגדרות, משתנים - app * מועבר אוטומטית לכל הפונקציות בתוכנית ולכן לא נכתב אותו יותר באפ fonkzia	קלט:	
fonkzia הראשונה שמופעלת אוטומטית בהרצה התוכנית.	תיאור כללי:	

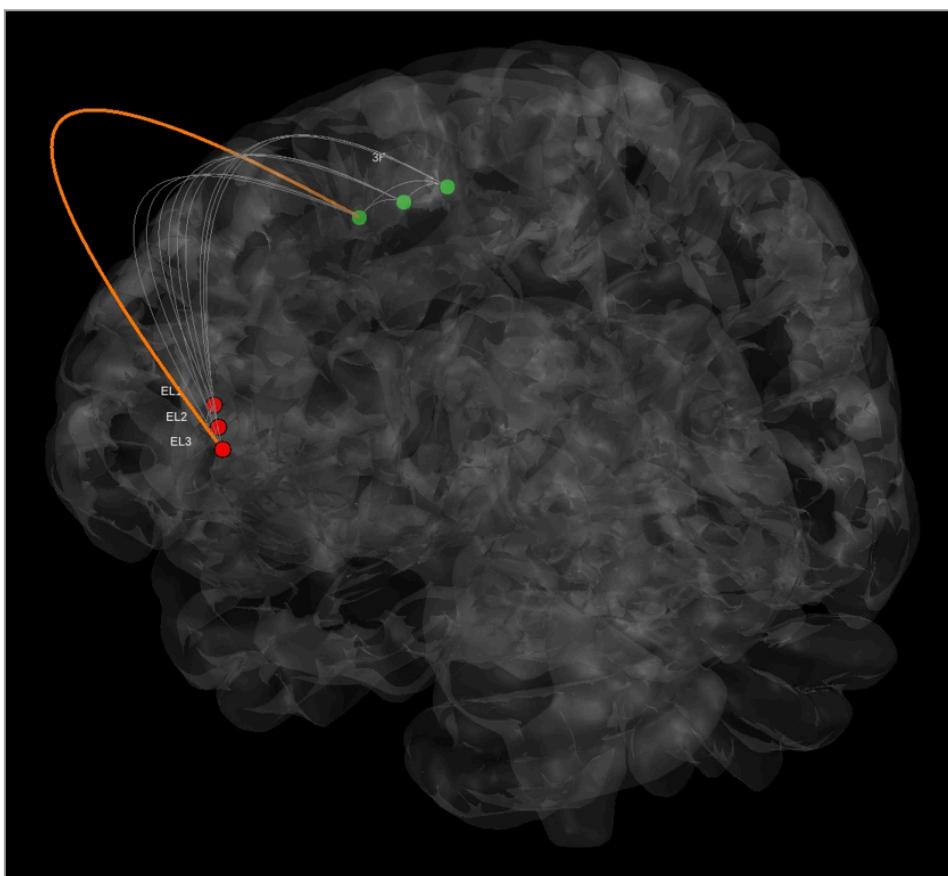
תחילה היא מגדרה משתני ברירות מיוחד כגון: ערכים לתפריט, הסירה/חישפה של האלקטרודות, מספר אלקטרודות, תדר דגימה (מוגדר תמיד ל-2000 Hz לפי ערך הדגימות של המעבדה), וכו'...
משיך באתחול הגדרות אינימצית הסיבוב של המוח בתלת מימד כגון: האם יש מידע טעון, האם מודיע מכמה לעיבוד, האם העיבוד מתבצע כתע, ואייפוס הטימר לזמן הטעינה.

2. Initialize UI & Brain Display	
customLayoutControl(src, evt)	הfonקציה: קלט: האובייקט שקרה לפונקציה, במקרה זה, החלון הראשי - src מידע על מה האירוע שהפעיל את הפונקציה, במקרה זה, שינוי בגודל - evt החלון הראשי ומיקומו הנוכחי.
customColormap()	תיאור כללי: במהרש מתבצע אתחול להגדרות של הממשק הגרפי כגון: הסירה של הפאנל הימני בו מופיע הגרף הדו-מימי עד לעיבוד המידע, אייפוס קווי החיבור בין האלקטרודות ואייפוס שמות האלקטרודות. הפונקציה customLayoutControl נקראת ידנית במטרה להחליף (override) את הפונקציה הדיפולטיבית של מطلب הקובעת את גודל הפאנלים של האפליקציה ואיזה מהם יופיעו או יוסתרו בהתאם לגודל החלון. מכיוון שבעת הפעלת התוכנית, לפני טעינת מידע, אין צורך בגרף הדו-מימי המציג את החיבורויות בין האלקטרודות נרצה להסתיר אותו. אזי בפונקציה נעשית הבדיקה האם יש מידע שנטען (לפי המשתנה הבינארי глובלי IsDataLoaded), במידע יש נקרא לפונקציה הדיפולטיבית של מطلب שתציג את כל הפאנלים ותתאים את גודלם לגודל החלון, אחרת, נקבע את רוחב הפאנל הימני ל-0 במטרה להסתיר אותו.
	תיאור כללי: יצירת מפת הצבעים הספציפית (לא מבירתה המחדל של המطلب). משיקולים עיצוביים נבחרה מפת צבעים ייחודית לתוכנה אשר משומשת להמחשת עצמת החיבוריות (ערך הPLV) בין אלקטרודות. גרדיינט הצבעים הולך לפי הסדר הבא מהנמוך לגבוהה: (0=) כחול כהה, סגול, אדום, כתום, לבן. (1=, הכי חזק)

משתנה המיקום (positions) קובע את מיקומי הצבעים בגרדיאנט, בין הערך 0 ל-1, ומשתנה הצבעים (colors) קובע את סדר הופעתם. כל צבע בمطلوب מיוצג על ידי ווקטור של 3 צבעים, בשיטת RGB, ויצירת הגרדיאנט דורשת "עירובוב" של כל אחד מהערכים האלה במרחב הלינארי של מפת הצבעים. לבסוף נוצר משתנה גלובלי cmap המכיל את ווקטור מפת הצבעים הסופית. במידה ומתרחשת שגיאה בכל אחד מהשלבים האלה התוכנה תשתמש במפת הצבעים הדיפולטיבית של מطلب מסווג parula, המכילה 256 גוונים. ההשראה העיצובית לקוחה מתוך המאמר "Visualizing effective connectivity in the human brain" ⁴.

updateBrainModel()

הfonקצייה:



תמונה:
(Fig.26)

תיאור כללי: ייצור מודל המהה תלת מימדי והפעלת האפשרות לסובב אותו בחופשיות בעזרת העכבר. המודל התלת-מימדי של המוח יושב בתוך אובייקט של מערכת צירים (app.BrainAxes)

תחילה מופיעים את המודל התלת-מימדי האחרון שנטען במטרה לנוקות/לאפס את המידע שמופיע עליו.

קובעים את הרקע לשחור בכך שלא יופיע הרקע הלבן הדיפולטיבי של מערכת היצירם.

השלב הבא הוא טיענת המודל התלת-מימדי המוכן מראש של המוח ששמור בקובץ head3d.mat. (נקח מתוךuproject⁵ "muse brain display", לא ברור אם הקובץ הוכן במיוחד לשימוש בפרויקט או נלקח מתוך מאגר open source אחר) משתמשים thereby בצד לחשוף את הקובץ בספריית הפרויקט.

הקובץ מכיל 2 סוג מידע: קודקודים ופאות. לכל קודקוד יש קורדינטות תלת-מימדיות בפורמט Montreal Neurological Institute space (MNI) וכל פאה מחברת בין 3 קודקודים.

הקובץ נתון למטרב ונשמר במשתנה הגלובלי BrainModelHandle.app, בו הוא מקבל את הגדרות הגראפיות כמו: צבע שטח הפנים, שקיפות, ותאורה.

אם איתור הקובץ או טעינתו נכשלים, נתון מודל תלת-מימי של כדור. לאחר טיענת המודל, מתאפשרות הגדרות מערכת היצירם למרכז את המודל, להסתייר את הקורדינטות ואת היצירם.

בשלב הסופי נערך בדיקה האם יש מידע טעון ומוכן להציג. במידה וכןו האלקטרודות, שמוטהן, וקיים החיבור ביניהן.

"יצירת קווי החיבור בגראף התלת-מימי נוצרת ע"י 3 פונקציות: drawBackgroundLines3D - מייצרת את קווי החיבור של האלקטרודות ללא "לחיצות" בצביע אפור דק. (שהמשתנה לא לחץ עליו) drawStrongestConnection3D - מייצרת את הקווים החזקים ביותר. (ערך הPLV הגבוה ביותר)

drawSelectedElectrodeLines3D - מייצרת את קווי החיבור של האלקטרודה שהמשתמש לחץ עליה, בצביע, עובי וגובה דינמיים לפי עצמת החיבורם.

הן דומות לפונקציה המיצרת קווי חיבור בגראף הדו-מימי (כתוב בהרחבה בשלב 12). עם מספר שינויים.

הרווח המינימאלי הוא 0.2 והמקסימלי 3.

ההבדל העיקרי הוא שקווי החיבור הם תלת-ממדיים והם מחברים בין כל 2ALKTRODOT בצורת קשת היוצאת מהמוח התלת-מימי.

בנוסף לעובי והצביע הדינامي, ככל שערך הPLV יותר גבוהה כך גם גובה הקשת.

<p><code>createInfoPanel()</code></p> <p>ולבסוף נוצר ומוסתר פאנל המידע שמוופיע מתחת לגרף הדו מימי המכיל את המידע הטקסטואלי של הדגימה.</p> <p>בפונקציה מוגדר המיקום של פאנל המידע: בפאנל הימני מתחת לגרף הדו-ימיidi.</p>	<p>הfonקציה:</p> <p>תיאור כללי:</p>
---	-------------------------------------

<p>3. Wait for User Input</p>	<p>בשלב זה התוכנה מחכה לאינפוט מהמשתמש.</p>
-------------------------------	---

<p>4. User Action</p>	<p>3 אפשרויות: 1. Upload File - להעלות קובץ. 2. Change Settings - שינוי הגדרות בתפריט. 3. Click Electrode - לחיצה על אלקטרודה.</p>
-----------------------	---

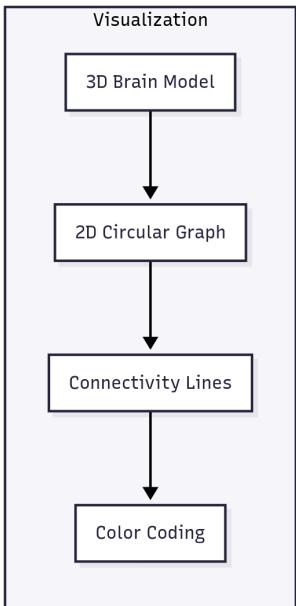
<p>5. Upload File</p>	
<p>6. Load .mat File</p>	
<p><code>UploadMATFileButtonPushed(event)</code></p>	<p>הfonקציה:</p>
<p>האירוע שקרה לפונקציה, במקרה זה, לחיצה על הכפתור - event</p>	<p>קלט:</p>
<p>המשתמש לוחץ על הכפתור "mat file (.mat file) Upload" ומעלה קובץ לתוכנה דרך החלון חיפוש קבצים שkopatz.</p> <p>כבריתת מחדל, החלון חיפוש הקבצים נפתח מאחוריו ומוסתר על ידי החלון התוכנה המרכזית. בכך ניתן לפותר בעיה זו, בעת לחיצה על כפתור "Upload" החלון התוכנה המרכזית מוסתר עד לשיום האינטראקציה עם החלון חיפוש הקבצים.</p> <p>לאחר שנבחר קובץ mat מתאים מופעלת אינימצית סיבוב המוח התלת-ימיidi, ומתחילה עיבוד המידע.</p>	<p>תיאור כללי:</p>
<p><code>setupLoadingAnimation()</code></p>	<p>הfonקציה:</p>
<p>הfonקציה מתחילה את המשתנים לפני הפעלת האינימציה של סיבוב המוח.</p> <p>תחילה היא מעדכנת את המשתנה הגלובלי IsProcessing ל-true ו"מכבה"</p>	<p>תיאור כללי:</p>

<p>את כל התפריטים בפאנל השמאלי כך שהמשתמש לא יוכל לשנות עוד הגדרות עד לסיום עיבוד המידע.</p> <p>לאחר מכן, היא מעדכנת את צבע נורת הסטטוס לצהוב עם הכיתוב "...Processing".</p> <p>ולבסוף היא "מנקה" את כל המידע הישן שמודיע על המודל התלת-מימדי של המוח. (אלקטרודות וקווי חיבור)</p>	
<p>Status lamp:</p> 	<p>תמונה: (Fig.27)</p>
<h3>7. Extract Patient Data</h3>	
<p>processDataWithAnimation(filePath, file)</p> <p>filePath מלאה לקובץ המידע שהמשתמש בחר - שם הקובץ בלבד - file</p>	<p>הfonקציה: הקלט:</p>
<p>הfonקציה הבאה שמופעלת מתחילה את שלב עיבוד המידע והפעלת האנימציה.</p> <p>מאותחל ומתחיל לפעול טיימר של 0.05 שניות לכל שינוי בזווית של המוח התלת-מימי ב כדי לחת את התחזשה של סיבוב חלק בזמן.</p> <p>cutת מתחיל ניסיין בעיבוד המידע המעודכן.</p> <p>כל המידע בקובץ נשמר במשתנה הגלובלי PatientData.app.</p> <p>שם המטופל מחולץ משם הקובץ הנבחר. המוסכמה במעבדה היא שם המטופל מופיע בסוף השם של הקובץ, لكن מועתקות 7 האותיות האחרונות, לדוגמה: "2017_02".</p> <p>בשלב הבא מתחיל עיבוד המידע כאשר נאסף המידע המעודכן על האלקטרודות ואז מועבד לפי תנאי הניסוי הנוכחי.</p> <p>אם המידע מועבד בהצלחה, האנימציה נפסקת והמידע מוצג במסך הגרפי.</p>	<p>תיאור כללי:</p>
<h3>8. Create Electrode Info</h3>	
<p>createElectrodeInfo()</p>	<p>הfonקציה:</p>
<p>איתחול המשתנה הגלובלי ElectrodeInfo.app המכיל את המידע עבור כל אלקטroduה.</p> <p>משיכת המידע עבור המטופל הנבחר במאגר המידע הסטטי של המעבדה "elecs_n79".</p>	<p>תיאור כללי:</p>

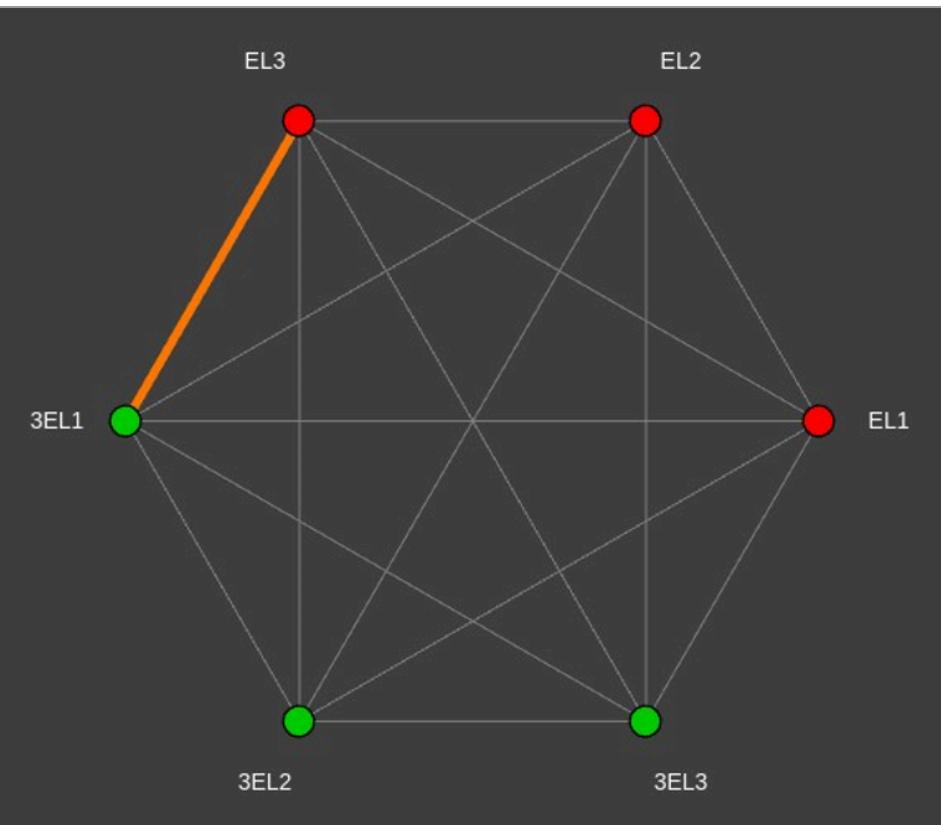
<p>במידה ונמצא המידע, השלב הראשון יהיה לפfilter רק את האלקטרודות הרלוונטיות המופיעות בקובץ "elecs_79_ch.elecs". (רק על הרקמה הבריאה) גם אלקטרודות (ערציז) הייחוס (רפנס) מסומנים בכך שלא ישפיעו על חישוב החיבוריות. (PLV) זה קורה באמצעות סימון של כל ערוץ רביעי ברצועה (טירוף אלקטרודות). בטלת מידע של מצב ניטוי בודד (לדוגמה rest_all) יכולות להופיע יותר מרצועה אחת, כולל יותר מ-4 ערוצים (שורות). עברו כל אלקטרודה נשמרות הקורדיינטות התלת-מימדיות שלה בפורמט NII. כמו כן, נשמר השם של כל אלקטרודה, הרשות בה היא ממוקמת, מאותחל משתנה השומר את החיבור הכי חזק שלא לאלקטרודה אחרת, וציב האלקטרודה לפי הרשות.</p>	
createNetworkColorMap()	הפונקציה:
פונקציה המתאימה צבע שונה לכל סוג רשת שונה בDATA. ماוחסן במשתנה גלובלי <code>app.NetworkColors</code> מסוג מילון. (dictionary)	תיאור כללי:

9. Process ECoG Data	
updateCurrentConditionData()	הפונקציה:
הפונקציה מכינה את המידע לעיבוד עבור תנאי הניסוי הנבחר. תחילה בודקת אם תנאי ניטוי נבחר ו"קוצצת" את ההקלטה לפי הוראות המעבדה: "לפני rest / countF (משימה קלה) יש להוריד שנייה מהוסף ושליש מההתחלתה. לפני alt (משימה קשה) להוריד שנייה מהוסף ו 3 שניות מההתחלתה". כתע הפונקציה מחברת בין כל ההקלטות בתנאי הניטוי להקלטה אחת ארוכה ומוחקת את ערוציז הייחוס. (רפנס)	תיאור כללי:
processAndDisplayData()	הפונקציה:
הפונקציה המפעילה את עיבוד המידע ומרעננת את התצוגה הגרפית. לבסוף מעדכנת את פאנל המידע הטקסטואלי להראות מידע כללי. (לפני שהמשתמש לוחץ על אלקטרודה ספציפית)	
10. Apply Frequency Filters	
11. Compute PLV Matrix	
processECOGData(ecogData, freqBand1Str, freqBand2Str)	הפונקציה:

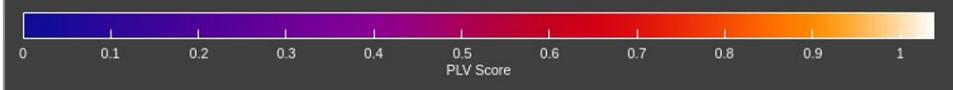
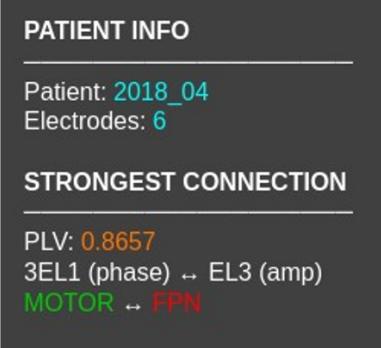
<p>המידע עבור תנאי המבחן הנבחר - <code>ecogData</code>, טווח התדרים הנבחר עבור האלקטרודה לשווואה - <code>freqBand1Str</code>, <code>string</code> וטווח התדרים הנבחר עבור שאר האלקטרודות - <code>freqBand2Str</code>, <code>string</code></p>	<p>הקלטן:</p>															
<pre> graph TD A[PLV Computation] --> B[Filter Signal Pairs] B --> C[Extract Phase & Amplitude] C --> D[Calculate Phase Differences] D --> E[Compute PLV Scores] </pre>	<p>תרשים זרימה: (Fig. 28)</p>															
<p>תיאור כללי: הפונקציה מבצעת את חישוב הPLV עבור כל צמד אלקטרודות. תחילה נקראת הפונקציה <code>getFrequencyBandLimits</code> המתרגמת את טווח התדרים מ-<code>string</code> לערבים מספריים. ואז חישוב הPLV בפונקציה מתבצע באופן הבא: <ul style="list-style-type: none"> - בשלב הראשון מוגדים טווח התדרים שהמשתמש בחר מסודר כך שהטווח הראשון הוא הנמוך (לו מחושבת הפאזה) והשני הוא הגבוה (לו מחושבת הפאזה של האמפליטודה). אם לא, נעשה תיקון אוטומטי של הסדר והמשתמש יעדכן בהודעה קופצת. - לולאה עברת על כל זוג אלקטרודות וממלאת מטריצה של ערכי PLV המיצגים את עצמת הסינכרון ביניהן. מטריצה של $3 \times N$ נראה כך: <table style="margin-left: auto; margin-right: auto;"> <tr> <td style="text-align: center;">E1</td> <td style="text-align: center;">E2</td> <td style="text-align: center;">E3</td> </tr> <tr> <td style="text-align: center;">E1</td> <td style="text-align: center;">x</td> <td style="text-align: center;">1p-2a</td> <td style="text-align: center;">1p-3a</td> </tr> <tr> <td style="text-align: center;">E2</td> <td style="text-align: center;">2p-1a</td> <td style="text-align: center;">x</td> <td style="text-align: center;">2p-3a</td> </tr> <tr> <td style="text-align: center;">E3</td> <td style="text-align: center;">3p-1a</td> <td style="text-align: center;">3p-2a</td> <td style="text-align: center;">x</td> </tr> </table> <p style="text-align: center;">$a =$ פאזה של האמפליטודה (amp) $c =$ פאזה של האות (phase)</p> <ul style="list-style-type: none"> - עבור כל צמד אלקטרודות החישוב מתבצע פעמיים: פעם אחת אלקטודה אחת מפולטרת לפי התדר הנמוך (phase) והשנייה בתדר </p>	E1	E2	E3	E1	x	1p-2a	1p-3a	E2	2p-1a	x	2p-3a	E3	3p-1a	3p-2a	x	
E1	E2	E3														
E1	x	1p-2a	1p-3a													
E2	2p-1a	x	2p-3a													
E3	3p-1a	3p-2a	x													

<p>הגבואה (amp) ואז הפוֹרַן.</p> <ul style="list-style-type: none"> - לאחר הפילטור מחשבים את ה"הבדל" (פער) בין הפאות של האותות. - ולבסוף מציבים בנוסחת הPLV לקבלת תוצאה בין 0 (אין סyncron) ל-1 (סינכרון מושלם). <p>בסוף הfonקציה נשמר ערך הPLV הגבואה ביותר לטובת הצגה בפאנל המידע הכללי (לפנוי שנבחרת אלקטטרודה ספציפית) וגם הערך הגבואה ביותר עבור כל אלקטטרודה.</p>	
calculateStrongestConnections()	הfonקציה: <p>fonקציה המוצאת ומאחסנת עבור כל אלקטטרודה את הפרטים הבאים: ערך הPLV הגבואה ביותר שלה, האינדקס של האלקטרודה אליה יש לה את הסינכרון החזק ביותר, שם האלקטרודה הצמודה, והרשת שלה.</p>
12. Update Visualizations	
 <pre> graph TD A[3D Brain Model] --> B[2D Circular Graph] B --> C[Connectivity Lines] C --> D[Color Coding] </pre>	תרשים זרימה: <i>(Fig. 29)</i>
updateBrainModel() → Written in step "2. Initialize UI & Brain Display"	הfonקציה:
updateCircularGraph()	הfonקציה:

תמונה:
(Fig.30)



תיאור כללי:
 יצירת הגרף הדו-מימדי המציג את החיבוריות בין האלקטרודות.
 הגרף הדו-מימדי יושב בתוך אובייקט של מערכת צירים. (app.CircleAxes)
 תחילה מופיעים את הגרף הדו-מימדי האחרון שנטען במטרה לנוקות/לאפס את
 המידע שמופיע עליו.
 האלקטרודות ממוקמות על גבי מעגל (ברדיוס 0.7 בצד שלא הגיעו לנקודה
 מערכת הצירים) ברוחחים שווים.
 תחילה מושפפים לגרף את קווי החיבורים באמצעות הפונקציה
`plotConnectivityLines2D`.
 כתת מיצרים את סימוני האלקטרודות והשמות שלהן.
 צבעי האלקטרודות מחולקים לפי הרשת בה הן נמצאות.
 אם יש אלקטרודה שהמשתמש לחץ עליה היא תוקף בקוו צהוב וגם השם שלה
 יסומן בצהוב.
 לבסוף, נסתיר את מערכת הצירים ונمرיץ את הגרף.
 הפונקציה מייצרת גם מקרה של בר צבע (colorbar) בתחתית הגרף הדו-מימדי
 שגבולותיו הם מ-0 עד ל-PLV היכי חזק כפול 2.1. (זאת בכך שאף קו חיבור לא
 קיבל את הצבע הלבן המקסימלי)
 בסיום, בסוף הפונקציה מתעדכן פאנל המידע לפי האלקטרודה הלחוצה.

Colorbar:		תמונה: (Fig.31)
plotConnectivityLines2D()	<p>הפונקציה מייצרת את קווי החיבור בין האלקטרודות בגרף הדו-מימדי.</p> <p>העובי של קווי החיבור נקבע באופן דינמי לפי עוצמת החיבור. (PLV)</p> <p>העובי המינימלי הוא 0.2 והמקסימלי הוא 4.</p> <p>מספר קווי החיבור הנוצרים הוא מספר האלקטרודות ברייבוע. (חיבורים בין כל האלקטרודות לכל האלקטרודות ח*ח)</p> <p>cut נקבע הצלע של קווי החיבור.</p> <p>במידה ואין אלקטרודה "לחוצה" (שהמשתמש בהר) אז צבע כל הקווים יהיה אפור בעובי מינימלי חוץ מהחיבור החזק ביותר בגרף, שהיה בצלע ועובי דינامي לפי עוצמת החיבור.</p> <p>חיבור החזק ביותר נבחר באופן גלובלי ממטריצת PLV.</p> <p>את החיבור החזק ביותר צובעים לפי מפת הצבעים שיצרנו בתחלת התכנית app.cmap.</p> <p>קווי החיבור נשמרם במשתנה app.CircleLinesHandles.</p> <p>במידה שיש אלקטרודה לחוצה שהמשתמש בהר בה, הפונקציה צובעת את כל החיבורים שלה עם שאר האלקטרודות באופן דינامي, וכל שאר החיבורים בין האלקטרודות הללו נבחרות מציריים באפור בעובי מינימאל.</p>	תיאור כללי: הfonkzia:
updateInfoPanel(selectedElectrodeIdx)		הfonkzia:
selectedElectrodeIdx	Aינדקס האלקטרודה שהמשתמש לחץ עליה -	הקלט:
General info panel:		תמונה: (Fig.32)
Selected electrode info panel:		

<div style="background-color: black; color: white; padding: 10px; width: 300px;"> <p>SELECTED ELECTRODE</p> <hr/> <p>Name: 2EL4 Network: FEM Coords: X: -53.7 Y: 35.9 Z: 10.2</p> <p>STRONGEST CONNECTION</p> <hr/> <p>PLV: 1.0000 To: EL4 (amp) Network: FEM</p> </div>	<p>(Fig.33)</p> <p>* הערכה: צילום המסך של פאנל המידע של אלקטרודה נבחרת נלקח מຕוך הרצה על דטה רנדומלי שנוצר רק בשבייל לבדוק את הכל. כМОון שערק PLV של דטה ארגני לא אמרו להגיע לערך 1.</p>
<p>תיאור כללי:</p> <p>הפונקציה מייצרת את המידע הטקסטואלי המוצג בפאנל המידע המוצג מתחת לגוף הדיו-מידם.</p> <p>הפונקציה מקבלת את האינדקס של האלקטרודה שהמשמש לחץ עליה, ואם אין אלקטרודה נבחרת אז מועבר האינדקס 0 ומוצג מידע כללי. (החיבור הכי חזק בגרף)</p> <p>תחילה נבדק באיזה תפקיד (phase/amp) המשמש רוצה להציג לאלקטרודה הנבחרת (ElectrodeRole) בכך לציין זאת בחיבור בין האלקטרודות.</p> <p>בפאנל המידע הכללי מוצגים:</p> <ul style="list-style-type: none"> - שם המטופל - מספר האלקטרודות - ערך PLV הגבוה ביותר - האלקטרודות בינהן הערך הגבוה ביותר והתדריות שלהן (phase/amp) - הרשותות שלהן <p>במידה ואלקטרודה לחוצה, מוצגים:</p> <ul style="list-style-type: none"> - שם האלקטרודה הנבחרת - הרשות שלה - הקורדינטות (X,Y,Z) <p>ערך PLV הגבוה ביותר של האלקטרודה הנבחרת</p> <p>האלקטרודה אליה יש לה את החיבור החזק ביותר והתדריות שלה</p>	

(phase/amp)	
- הרשות שלALKTRODת החיבור	

13. Show Results

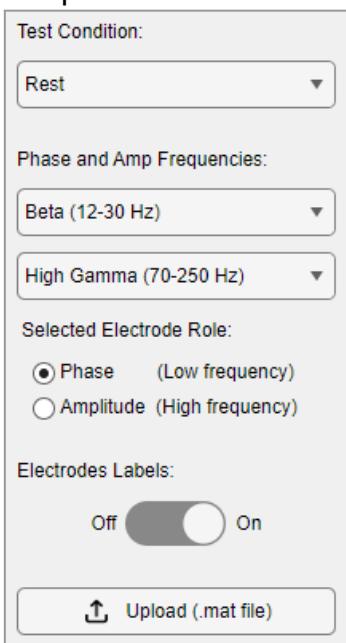
→ Go to step “3. Wait for User Input”

14. Change Settings

15. Update Parameters

<p>המשתמש יכול לשנות כל אחד מערכי התפריט הצדדי (פאנל שמאל) לאחר שינוי בכל תפריט (event) מופעלת הפקציה המתאימה לעדכון ההגדלה.</p>	תיאור כללי:
--	--------------------

Left panel menu:



תמונה:
(Fig.34)

ConditionDropDownValueChanged(event)

הפקציה:

האירוע שקרה לפונקציה, במקרה זה, שינוי בערך התפריט - event

הקלט:

תיאור כללי:
בחירה תנאי הניסוי.

תפריט בחירה:

Test condition • - תנאי הניסוי

Rest ○ - מנוחה

	<ul style="list-style-type: none"> <input type="radio"/> CountF - תנאי קל <input type="radio"/> Alt - תנאי קשה <p>תחילה נבדק האם יש מידע טען במשתנה app.PatientData</p> <p>אם כן, נורט הסטטוס בתחתית הפאנל האמצעי (הגרפי) משנה את הצבע לצהוב ומופיע לידה הכיתוב "...Processing".</p> <p>לאחר מכן המידע הטען מתעדכן לתנאי הניסוי החדש, המידע החדש מעובד ומוצג בפאנל האמצעי, ולבסוף במידה והטהליך מסתיים בהצלחה נורט הסטטוס משנה את צבעה לירוק. (אחרת, נורט הסטטוס מוארת באדום, מופיע לצדיה הכיתוב "Error" ו קופצת הודעה השגיאה המתאימה)</p>
Freq1DropDownValueChanged(event) Freq2DropDownValueChanged(event)	הfonkzia:
האירוע שקרה לפונקציה, במקרה זה, שינוי בערך התפריט - event	הקלט:
<p>בחירה טווח תדרים לחישוב הPLC עבור כל אלקטרוזדה.</p> <p>תפריט הבחירה הראשון הוא עבור האלקטרוזדה הנבחרת, והתפריט השני הוא עבור ההשוואה עם כל שאר האלקטרוזדות.</p> <p>בין 2 התפריטים קיימים כפתורים המאפשרים החליף בין טווחי התדריות לאלקטרוזדות בלחיצת כפתור.</p> <p>תפריט בחירה:</p> <ul style="list-style-type: none"> • Phase vs Amplitude Frequency (Delta (0.5-4 Hz) <input type="radio"/> (Theta (4-8 Hz) <input type="radio"/> (Alpha (8-12 Hz) <input type="radio"/> (Beta (12-30 Hz) <input type="radio"/> (Gamma (30-70 Hz) <input type="radio"/> (High Gamma (70-250 Hz) <input type="radio"/> <p>תחילה נבדק האם יש מידע טען במשתנה app.PatientData</p> <p>נורט סטטוס מתעדכנת לצהוב.</p> <p>המידע החדש, לפיבחירה המשמש, מעובד ומוצג.</p> <p>אם התהלהיך מצליח הנורה הופכת לירוקה אם לא אז אדומה.</p>	תיאור כללי:
ElectrodeRoleChanged(event)	הfonkzia:

האירוע שקרה לפונקציה, במקרה זה, שינוי בcpfotori הרדיו - event	הקלט:
<p>כפטור רדיו המחליף בין תפקיד האלקטרודה הנבחרת בחישוב PLV:</p> <ul style="list-style-type: none"> • Phase (Low frequency) • Amp (High frequency) <p>אם המשתמש בוחר ב"Phase" אז אותן מהאלקטרודה הנבחרת יsoon בטוויה התדריות הנמוך ותחושב עבורה הפאזה (phase) ביחס לפאזה האמפליטודה (amp) של האלקטרודות האחרות, שיסוננו בטוויה התדריות הגבוהה. אם המשתמש בוחר ב"Amp" אז יציג הפור, פאזה האמפליטודה של אותן מהאלקטרודה הנבחרת לעומת הפאזה של אותן מכל שאר האלקטרודות.</p> <p>מייצג משתנה בינהרי גלובלי app.ElectrodeRole.</p> <p>כאשר הפונקציה מופעלת המשתנה ElectrodeRole מעודכן, שתי הגרפים מתעדכנים, ופאנל המידע מתעדכן רק אם יש אלקטרודה "לחוצה". אחרת הוא ממשיך להציג את המידע הכללי.</p>	תיאור כללי:
ShowElectrodeLabelsSwitchValueChanged(event)	הפונקציה:
האירוע שקרה לפונקציה, במקרה זה, לחיצה על המטג - event	הקלט:
<p>מטג המדליך/מכבה את התצוגה של שמות האלקטרודות על הגרפים.</p> <p>יש לו 2 ערכים טקסטואליים: 'off' / 'on'.</p> <p>לחיצה עליו בכל שלב בתוכנית תציג/תסתיר את שמות האלקטרודות.</p>	תיאור כללי:
16. Data Loaded? No → Go to step “3. Wait for User Input” Yes → Go to step “9. Process ECoG Data”	

17. Click Electrode	
לחיצה על כל אחת מהאלקטרודות בגרף הדו-ימידי תציג את עצמת הסינכרון (PLV) שלו עם כל שאר האלקטרודות, ובפאנל המידע יופיע המידע על הסינכרון הכי חזק.	תיאור כללי:
18. Select/Deselect Electrode	
onElectrodeClicked(src,~)	הפונקציה:

<p>האובייקט שקרא לפונקציה, במקרה זהה, האלקטרודה הנבחרת - <code>src</code> משתנה מיוצר שלא משתמשים בו בפונקציה. מה שנשלח במקרה לפונקציה - ~ הוא האירוע של הלחיצה על האלקטרודה</p>	<p>הקלט:</p>
<p>פונקציה המתארת לחיצה על אלקטרודה ומרעננת את התצוגה של הגרפים. תחילה מ Chapman במשתנה השמור בתוך אובייקט האלקטרודה שנקרא <code>UserData</code>, אם לא מוצא, אם לא מוצא, הולך לחפש בגוף התרטט-מיידי, אם לא מוצא הולך לחפש בגוף הדו-מיידי. ברגע שמאתר את אינדקס האלקטרודה הנבחרת (<code>Select</code>), מעדקן את פאנל המידע. אם האינדקס דומה לאינדקס הקודם שנבחר אז מבטל (<code>Deselect</code>) את הבחירה באלקטרודה ומציג את המידע הכללי בפאנל המידע.</p>	<p>תיאור כללי:</p>
<h3>19. Update Electrode Display</h3>	
<p><code>updateElectrodeAppearance()</code></p>	<p>הפונקציה: תיאור כללי:</p>
<p>נראות האלקטרודה הנבחרת משתנה להיקף צהוב, שמה מתעדכן לצהוב, והיא גדולה.</p>	
<h3>20. Redraw Connections</h3>	
<p>לאחר בחירה/ביטול בחירה באלקטרודה, קווי החיבור מתעדכנים. בבחירה אלקטרודה קווי החיבור יראו את כל החיבורים של האלקטרודה עם כל שאר האלקטרודות. בביטול בחירה, הם יחזורו למצב הכללי המציג את החיבור החזק ביותר בלבד. עבור הגוף התרטט-מיידי, רק קווי החיבור יתעדכנו ע"י הפונקציות המתאימות שתוארו בפונקציה <code>updateBrainModel</code> (<code>updateBrainModel</code>) בשלב 2. עבור הגוף הדו-מיידי, כל הגוף יתעדכן, כולל קווי החיבור. זה קורה בגלל שהגוף התרטט-מיידי כבד יותר, וליצור את כלו מחדש דורש יותר זמן וчисובים. יצירת הגוף הדו-מיידי מחדש כולל קווי החיבור הוא מהיר ופשוט אזי עדיף פשוט לרענן אותו עם המידע החדש.</p>	<p>תיאור כללי:</p>
<h3>21. Update Info Panel</h3>	
<p>לבסוף פאנל המידע יתעדכן לפי האלקטרודה הנבחרת/مبוטלת. כתוב בהרחבה בתיאור הפונקציה <code>updateInfoPanel</code> (<code>updateInfoPanel</code>) בשלב 12.</p>	<p>תיאור כללי:</p>

→ Go to step “3. Wait for User Input”

פונקציות תמייה נוספת שלא תוארו בשלבי התרשימים

1. Graphics -
 - a. continuousRotation()
 - b. stopLoadingAnimation()
 - c. hexColor = rgbToHex(rgbColor)
 - d. hexColor = getPLVColor(plvValue)
 - e. hexColor = getNetworkColor(networkName)

2. Processing -
 - a. $y = \text{hilbert_alt}(x)$
 - b. [low_freq, high_freq] = getFrequencyBandLimits(bandName)

3. Info Panel -
 - a. setInfoPanelVisible(visible)

4. Electrodes & Lines -
 - a. [xArch, yArch, zArch] = create3DArch(point1, point2, numPoints, archHeight)

הסבר על פונקציות התמייה

1. Graphics	
continuousRotation()	<p>הfonקציה: continuousRotation()</p> <p>תיאור כללי: הפונקציה אחראית על סיבוב רוחבי של המודל התלת-מימדי של המוח בזמן עיבוד המידע. ("אנימציית המתנה")</p> <p>המשתנה הראשון שמודגזר הוא rotationAngle, והוא מסוג persistent. כלומר, הערך שלו נשמר בין הקריאה השונות של הפונקציה. הוא מתחילה מזוויות 0.</p> <p>cutת מתחרשת בדיקה האם עדין מעובד מידע, במידה וכן, זווית התצוגה של המוח מתעדכנת לזוית הנוכחית (הקודמת + 3 מעליות סיבוב רוחבי).</p>

<p>הערך 20 הוא הזרoit האנכית שנשארת קבועה לאורך כל הריצת הפונקציה. (ורק הזרoit הרוחבית משתנה)</p> <p>הדבר האחרון שקרה הוא שמתעדכנת התצוגה באופן יידי (drawNow) אך עם "האטה" קטנה (limit) בכך להמנע מרענון מהיר מדי לפני שההתצוגה מתעדכנת.</p>	
<p><code>stopLoadingAnimation()</code></p>	<p>תיאור כללי:</p> <p>הפונקציה האחראית להפסקת אнимציית סיבוב המוח.</p> <p>ראשית, המשתנה הבינארי הגלובלי המסמן שמתறש עיבוד מידע משנה את ערכו <code>false</code>.</p> <p>ואז משתנה הטימר נמחק ומתאפשר.</p> <p>ולבסוף כל אובייקטי התפריט "משתחררים" לשימוש חופשי של המשתמש, ונורת הסטוס מחליפה את צבעה לירוק.</p>
<p><code>hexColor = rgbToHex(rgbColor)</code></p>	<p>תיאור כללי:</p> <p>הקלט:</p> <p>hexColor: משתנה טקסט (סטרינג) המתאר צבע בקוד הקסאדיימלי, = '#000000' זה שחור</p>
<p>הפונקציה ממירה מקוד צבע RGB לקוד צבע הקסאדיימלי HEX.</p> <p>הסיבה שאנו צריכים את הקוד HEX היא בשbill לשנות את צבע הטקסט בפאנל המידע לפי צבע האלקטרודה (הרשות בה היא נמצאת) בקוד HTML.</p> <p>קוד RGB הוא וקטור המכיל 3 מספרים בין 0-255 אחד מהם מתאר את צמות הצבע האדום (Red), ירוק (Green), וכחול (Blue) בצבע הנבחר.</p> <p>קוד HEX הוא קוד טקסטואלי המורכב מסולמית '#' ואחריה 6 ספרות שכל צמד בהן מתאר צבע אדום, ירוק, כחול.</p> <p>הפונקציה ממירה כל מספר בקוד RGB לזוג ספרות HEX ומחייבת את המחרוזת.</p>	<p>תיאור כללי:</p>
<p><code>getPLVColor(plvValue)</code></p>	<p>תיאור כללי:</p>
<p>הקלט:</p> <p>plvValue PLV מספרי - ערך PLV</p>	
<p>הפונקציה ממירה כל מספר בקוד הקסאדיימלי = '#000000' לאחד צבעים (colorbar) לפי ערך הPLV.</p>	<p>תיאור כללי:</p>

<p>תחילה, אם ערך הLVPC לא תקין הצביע הדפוולי בי הוא אפור. (#696969) בהמשך, נורמל את ערך הLVPC לפי הנירמול שעשינו לבר הצביעים. (העלאנו את המקסימום שלו להיות ערך הLVPC הכי גובהה כפול 1.2, ב כדי שאך חיבור יהיה בצבע לבן)</p> <p>לבסוף נמיר את ערך הRGB שחוזר מ משתנה הקאומס באינדקס הצביע למחוזת HEX ע"י הפונקציה <code>rgbToHex()</code>.</p>	
<p><code>getNetworkColor(networkName)</code></p>	<p>הfonקציה:</p>
<p>שם הרשת (מחוזת) שבה נמצאת האלקטרודה - <code>networkName</code></p>	<p>הקלט:</p>
<p>משתנה טקסט (סטרינג) המתאר צבע בקוד הקסאדיימלי = <code>hexColor</code></p>	<p>הפלט:</p>
<p>הפונקציה מחזירה את צבע הרשת המתאים בקוד HEX.</p> <p>תחילה נבדוק האם שם הרשת מופיעה במיילון צבעי הרשותות (NetworkColorMap app) שהרכבנו בפונקציה <code>createNetworkColorMap()</code>.</p> <p>במידה ושם הרשת מופיעה, נוציא את הצבע המתאים מהמיילון בקוד RGB ונמיר לHEX ע"י הפונקציה <code>rgbToHex()</code>.</p> <p>במידה ולא קיים, נתונים צבע אפור לרשותות לא מזוהות. (#808080)</p>	<p>תיאור כללי:</p>

2. Processing	
<p><code>y = hilbert_alt(x)</code></p>	<p>הfonקציה:</p>
<p>הסיגנל עליו אנחנו רצים להפעיל טרנספורם הילברט - <code>x</code></p>	<p>הקלט:</p>
<p>הסיגל אחריו הפעלת הילברט = <code>y</code></p>	<p>הפלט:</p>
<p>הפונקציה מחשבת הילברט טרנספורם (hilbert transform) ידנית על סיגナル במידה והרחבת 'x' Signal Processing Toolbox' לא מותקנת על המחשב או שיש אליה בעיה בשרת.</p>	<p>תיאור כללי:</p>
<p><code>[low_freq, high_freq] = getFrequencyBandLimits(bandName)</code></p>	<p>הfonקציה:</p>
<p>מחוזות המתארת את טווח התדרים הנבחר מהתפריט - <code>bandName</code></p>	<p>הקלט:</p>
<p>גבול תחתיו (תדר נמוך = <code>low_freq</code>)</p>	<p>הפלט:</p>
<p>גבול עליון (תדר גבוהה = <code>high_freq</code>)</p>	

<p>הfonקציה מתרגם את טווח התדרים ממחוזת (string) לטווח מספרי. [תדר נמוך, תדר גבוהה]</p>	תיאור כללי:
---	--------------------

3. Info Panel	
<code>setInfoPanelVisible(visible)</code>	הfonקציה:
משתנה בינהר המתאר האם צריך להסתייר/להציג את פאנל המידע - <code>visible</code>	הקלט:
הfonקציה מציגה/מסתירה את פאנל המידע ומעדכנת את המשתנה הבינהר <code>global.InfoPanelVisible</code> .	תיאור כללי:

4. Electrodes & Lines	
<code>[xArch, yArch, zArch] = create3DArch(point1, point2, numPoints, archHeight)</code>	הfonקציה:
קורדינטות תלת-מימדיות (z,y,x) של נקודת התחלה - <code>point1</code> קורדינטות תלת-מימדיות (z,y,x) של נקודת הסוף - <code>point2</code> מספר נקודות לאורך הקשת - <code>numPoints</code> גובה הקשת - <code>archHeight</code>	הקלט:
מערך ערכי קורדינטות x של כל הנקודות של הקשת = <code>xArch</code> מערך ערכי קורדינטות y של כל הנקודות של הקשת = <code>yArch</code> מערך ערכי קורדינטות z של כל הנקודות של הקשת = <code>zArch</code>	הפלט:
הfonקציה מייצרת קשת (Arch) חיבור תלת-מימדית בין 2 אלקטודות. תחילה הfonקציה מגדרה ערכי דיפולט במידה והיא לא מקבלת את כל ערכי הקלט. שנית מוצאים את נקודת האמצע של הקשת ע"י ממוצע בין הנקודות התחלה וסוף ומחשבים את המרחק ביניהן. যוצרים מערך ראשוני שמייצג את הקשת (t) ע"י חלוקה של המרחב בין נקודות התחלה והסיום למקטעים כמספר הנקודות לאורך הקשת. (<code>numPoints</code>)	תיאור כללי:

לאחר מכן מוצאים את הכוון שיוצאה מהמוח, לפי חישור הווקטור התלת-מימדי של נקודת מרכז המוח (ראשית הצירים $[0,0,0]$) וווקטור נקודת האמצע של הקשת.
ולבסוף יוצרים את הקשת ע"י נסחה פרבוליית המענייקה לכל נקודה לאורך הקשת גובה משתנה. (עולה עד האמצע ואז יורדת)

8. מסקנות וסיכום

האם התוצאות מתאימות לניתוח הראשוני?

על פי הניתוח הראשוני (בהתאם לממצאי המאמר של¹ Assem & Erez (2023), צפינו לראות ערכי PLV גבוהים יותר בתוך רשת ה- FPN בזמן שימוש הדורשות שליטה קוגניטיבית (משימה קשה (Hard) - ספירה של אותיות ומספרים) לעומת משימה מנוחה (Rest), וכן ירידת בקישוריות בין רשותות שונות (כגון FPN ו- DMN) כאשר מתבצעת משימה הדורשת מיקוד גבוה. (משימה קלה (Easy) - ספירה פשוטה של מספרים)

התוצאות המוצגות בפרק "בדיקה השיטה, תוצאות וניתוחן" עלות בקנה אחד עם ממצאי המאמר, וניתנות לניתוח בצורה ויזואלית באמצעות הכליל בהצלחה.

יתרונות וחסרונות של הפתרון

בטבלה הבאה מ羅צים היתרונות והחסרונות של הכליל להציג גרפית.
ישנו מספר **מגבליות טכניות נקודתיות של הכליל** במצבו הנוכחי עלייהן כתבתי בהרחבה בפרק הבא. ("רעיון למשך")

חסרונות	יתרונות
<ul style="list-style-type: none">כיסוי מוגבל של המוח - רישומי ECoG מתבצעים רק מאזורים עליהם הונחו אלקטרודות לצורך הניתוח. משמעות הדבר היא שהנתונים אינם מספקים תמונה מלאה של רשותות המוח, אלא מבט מקומי ומוגבל למיקום ההשתלה.	<ul style="list-style-type: none">המבנה מרחבית של רשותות ואיזורי המוח - הייצוג התלת-ממדי מאפשר לזהות את המיקום המדויק של כל אלקטרודה על פני הקורטקס ואת הקשרים בין אזורים שונים. כך ניתן להבין כיצד רשותות מוחיות שונות מתחשרות זו עם זו ולזהות גבולות תפוקודים באופן חזותי ו ישיר.
<ul style="list-style-type: none">הטרוגניות בין מטופלים - מיקום ציפויות האלקטרודות משתנים ממטופל למטופל לפי הצרכים הקליניים, ולכן קשה להשוות ישירות בין נבדקים או להכליל תוצאות לרמה קבוצתית.	<ul style="list-style-type: none">הצגה טובה של מטופל בודד - הכליל מאפשר ניתוח מותאם אישית של הנתונים לכל נבדק, בהתאם לפריסת האלקטרודות הספציפית שלו. זהו יתרון משמעותי בניתוח קליני, שבו כל מטופל משמשו עליון בניתוח קליני, והוא מושפע מחלוקת המיקום והיקף היכסו של רשותות לרעש - אותן עשוים לכלול רעשים שמקורם בתנועות, גיריה

<p>חשמלית, או פעילות שרירית. סינון לא מושלם עלול לגרום להטיה בתוצאות החישובים של הקישוריות.</p> <ul style="list-style-type: none"> מורכבות חישובית ופרשנית - מודיעיני קישוריות כמו PLV רגשים לשינויים מתמטיים (כמו פילטרים, טווחי תדרים, והפרמטרים של הגלונים), ולכן יש צורך בזיהרות בפרשנות - לא כל שינוי במדד משקף בהכרח שינוי בתקשרות עצבית אמיתית. פרשנות חזותית סובייקטיבית - הציג גרפית של נתוני מוח מורכבים דורשת בחירה של ספים, צבעים, וטווחים. בחירות אלו משפיעות על האופן שבו המשתמש מפרש את המידע ועלולות להוביל למסקנות חזותיות מטעות. 	<p>האלektטרודות.</p> <ul style="list-style-type: none"> הציג אינטואיטיבית וברורה ללא צורך במידע עמוק - ויזואליזציה אינטרاكتיבית מפשטת את הנתונים המורכבים ומאפשרת גם לרופאים, סטודנטים או אנשי צוות שאינם מתחמנים בניתוח אוטומטי להבין את התמונה הכללית של הפעולות המוחית והקישוריות. הקלת בניתוח של פעילות רשתות - הציג עצמות הקישוריות בין האלקטרודות מאפשרת לזהות בצורה מהירה היכן קיימים קשרים חזקים יותר - האם בעיקר בתחום רשת תפוקודית, או בין רשתות שונות. כך ניתן להסיק על ארגון הרשתות ותפקודן בזמן שימושה או במנוחה. סביבת MATLAB המתאימה לשימוש במעבדה - MATLAB היא סביבת עבודה מוכרת ונפוצה בתחום חקר המוח. היא מאפשרת אינטגרציה נוחה עם קבצי נתונים קיימים, ביצוע עיבודים מתקדמים, ויכולת הרחבנה עתידית (למשל שילוב אלגוריתמים חדשים למדידת קישוריות או עיבוד בזמן אמת). יכולת התאמת אישית - המערכת ניתנת להתאמת בקלות לצרכים שונים: הוספה מדדים חדשים, שינוי הציג, או הצגת תדרים מסוימים בלבד.
--	--

סיכום

לסיכום, מטרת הפרויקט הייתה לבנות כלי שיעזר לאפ"ן קישוריות של קליפת המוח הקדמית עם סנסורים רבים שמונחים על המוח, ספציפית באיזורים שאחראים לתפקידים הקרייטיים. (Executive functions)

מטרה זו הושגה בעזרת ייצרת הכלי `PLView`, אשר מאפשר הצגה ויזואלית של האלקטרודות על גבי המוח וניתוח הקישוריות ביןיהן בעזרת חישוב `PL` בין כל זוג אלקטרודות והציג התוצאות.

התוצאות שנצפו בפרויקט תאמו את הניתוח הראשוני וענו על הצפויות לציג גרפיט ברורה, י呜ה, אוינטראקטיבית.

כעת ניתן להעמק ולהרחיב את יכולות הכלי, ולהמשיך להתאים אותו לצרכים המתפתחים של המעבדה ותחום המחקר.

בנימה אישית, למדתי והתפתחתי רבות במהלך העבודה על הפרויקט. בתחום ניתוח אוטומטי נירולוגיים, בתחום ויזואליזציה של מידע מורכב, ובתחום פיתוח כלים גרפיים המשלבים עיבוד וניתוח מידע מקדים. ההבנה החשובה ביותר אליה יצאתי מהפרויקט הוא שימוש הויזואליזציה של אוטומטי נירולוגיים הוא תחום שאשם לפתח ולהשתלב בו במסגרת קריירה או מחקר בעtid. כשהתחלתי לעבוד על הפרויקט ידעתי שאנה חלק העיצוב הויזואלי בו יש לי ניסיון עבר, אך היה לי חשוב לשלב גם חלק טכני, להבין לעומק את החישובים המתמטיים ואת שלב העיבוד המקדים של המידע לפני הציגו. הפרויקט אכן שילב בין התchromים, ואיתגר אותי ללמידה כלים חדשים בהם לא התנסתי מעולם, כמו MATLAB's App Designer.

אני גאה ומורוצה מההתוצר הסופי ומקווה שהוא יביא ערך למעבדה, ויעזר לקדם את המחקר בה.

9. רענוןת להמשך

לאחר הוכחת ההיתכנות הראשונית ובנית התשתיות הגրפית והפונקציונאלית,icut ניתן להמשך לפתח את הכלי ולהוסיף יכולות חדשות.

השאיפה העיקרית בפרויקט זה הייתה לייצר כלי שיעוד בעתיד לסייע למפתחים לזהות איזורים במוח שאחראים לתפקידים קרייטיים (Executive functions) בזמן-אמת בניוית מוח פתוח. היתרונות של כלי ויזואלי מסווג זה המ שיפור ההתמצאות המרחכית וצמצום הפגיעה באיזורי מוח קרייטיים בזמן הניוית. בעוד הכלי הנוכחי מעבד את המידע תוך מספר שניות בודדות ומציג אותו בצורה ויזואלית ברורה, המידע עליו הוא משתמש ע過ר בשלב עיבוד ראשוני עוד לפני הפעלת הכלי: מחושב מחדש הרפרנס ומסונן במסנן notch. (בשם הקובץ: "..._only_reref_notch50_79_...") בנוסף, הכלי עובד על הקלטות שלמות ומלאות לאחר ביצוע כלל תנאי הניסוי (מנוחה, משימה קלה, משימה קשה) כמספר פעמים. בעוד שהאוון אינו חסם מיד לשימוש בכלי בזמן-אמת, כיוון שנitin להשלים את המשימות ורק אז לקבל תמונה מצב של המוח, הכלי האידייאלי בעניין ידע לתת תמונה מצב דינמית, עדכנית, ומתעדכנת לכל אורך הניוית.

יכולת נוספת שנitin יהיה להוסיף לכלי בעתיד היא **הציג חישובים נוספים** לניטוח הקישוריות ואיפיון הרשותות המוחיות. כיום הכלי מציג את הקישוריות המוחית על פי חישוב LVPC בלבד. בהמשך ניתן יהיה להרחיב את יכולות הכלי כך שיינתח ויציג מגוון רחב יותר של מדדים מתוך אותן ה-GECO. מעבר למדידת הסנכרון בין האלקטרודות, ניתן יהיה לחשב מדדים נוספים המתארים את עצמת הפעולות החשמליות באזוריים שונים, תగובת האיזור לגירוי החשמלי בניוית, את הדינמיקה של הקישוריות לאורך זמן, ואת הכווניות של ההשפעה בין איזורי המוח. בנוסף, ניתן יהיה להפיק מדדים מרמת הרשת המוחית, כגון זיהוי איזורים מרכזיים וחלוקה לפי פונקציונליות. שילוב מדדים אלו בתצוגה תלת-ממדית ודו-ממדית יאפשר הבנה עמוקה יותר של דפוסי הפעולות המוחית ושל האינטראקציות המורכבות ביניהם.

בנוסף, הכלי יוכל לאפשר עיבוד והציג ויזואלית מידע של מטופל בלבד בלבד, בהמשך ניתן יהיה להרחיב את האפשרות ולאפשר עיבוד **מידע של מטופלים מרובים**, לחשב, ולהציג השוואות מעניינות. יכולת זו תאפשר לבצע חישובים קבוצתיים כגון ממוצע LVPC בין נבדקים, ניתוח שונות בין קבוצות, והציג מפות קישוריות ממוצעות או השוואתיות על גבי המודל התלת-ממדי. הרחבה צזו תאפשר לחוקרים ולרופאים לzechot דפוסים משותפים של קישוריות תפקודית, ולאחר חריגות של מטופלים אינדייבידואלים.

לבסוף, יכולת מעניינת נוספת שנitin יהיה להוסיף לכלי היא **הציג הקלטות של הפעולות המוחית בזמן**. מכיוון שהמידע כבר קיים בקבצים ניתן לבצע ניתוחים מעניינים בזמן, לקשר בין עצמת פעילות

ברגע ספציפי לגורם, ולראות באופן ויזואלי את הפעלת האיזורים השונים בmouth בעת ביצוע תנאי הניסוי השונים.

בנוסף להרחבת והוספת יכולות חדשות לכלי, ישנו מספר **שיעורים טכניים נקודתיים** היכולים לשפר את היכולות הקיימות, אותן לא הספקתי ליישם בפרויקט זה:

1. **הציג כל ערכיו של PLV עבור אלקטודה נבחרת** - הכלים הנוכחיים מציגים עבור כל אלקטודה את ערך הקישוריות הגבוהה ביותר בלבד ואת האלקטודה אליה היא מקושרת. שיפור מתבקש יהיה להוסיף, בנוסף טקסטואלי או ויזואלי, את ערכיו הקישוריות (PLV) עבור כלל הקשרים של האלקטודה הנבחרת. זהו שיפור פשוט יחסית מכיוון שכל המידע כבר קיים במטריצת PLV המחשבת בשלב עיבוד המידע, ורק נדרש להציגו.
2. **הוספה חישובים למידע מרקמה חוליה** - כיום האלקטודות המעובדות הן רק אלה הנמצאות על הרקמה הבריאת. (מפולטות לפי האלקטודות המופיעות במאגר המידע של המעבדה "79ch_elecs") בהמשך ניתן יהיה להוסיף לנתח גם אלקטודות על רקמה חוליה וליצור השוואות מעניינות נוספת בינהן. (המידע עליון כבר נמצא בקובץ המידע של המעבדה, הוא פשוט לא מעובד היום בכלל)
3. **איתור שם המטופל באופן דינامي** - בפרטן הנוכחי, שם המטופל נלקח בצורה אוטומטית מ-7 האותיות האחרונות של הקובץ המועלה. הצעה לשיפור תהיה למצאו את שם המטופל בקובץ בצורה יותר כולנית ופחות "ידנית" שתתמוך בצורה יותר גמישה בשמות של קבצים גנריים. (לא רק בפורמט השמות של המעבדה)
4. **הוספת תמיכה במידע גלומי** - כיום הכלים תומך במידע המסונן במסנן notch ועובד רি-רפרנסינג לפני העבודה בכלים. (בשם הקובץ: "..._only_reref_50_79_notch_...") ניתן לשפר את הכלים שיתמוך בעוד סוגים נוספים ("..._only_reref_50_79_notch_...") המודיע גלומי (איזה מסנן, באיזה תדרים, ואיזו שיטת ררי-רפרנסינג), ועוד ליתר את שלב העבודה המקדמים שנערך ביום מחוץ לכלי.
5. **אפשרות ליצוא גיליון תוצאות** - הוספה שימושית שעלה בעת כתיבת ספר הפרויקט (ספציפית בפרק של "בדיקה השיטה, תוצאות וניתוח") היא האפשרות ליצוא גיליון תוצאות מסודר בלחיצת כפתור אחד עבור כל ערכי הקישוריות ובכל תנאי הניסוי, דוגמה לatable המופיעה בפרק "בדיקה השיטה, תוצאות וניתוח". הפורמט המומלץ יהיה בטבלת אקסל (.csv) או טבלת CSV..

10. ביבליוגרפיה

1. Moataz Assem, Michael G. Hart, Pedro Coelho, Rafael Romero-Garcia, Alexa McDonald, Emma Woodberry, Robert C. Morris, Stephen J. Price, John Suckling, Thomas Santarius, John Duncan, Yaara Erez. High gamma activity distinguishes frontal cognitive control regions from adjacent cortical networks, Cortex, Volume 159, 2023, Pages 286-298.
<https://doi.org/10.1016/j.cortex.2022.12.007>.
2. Ayan S. Mandal, Moataz Assem, Rafael Romero-Garcia, Pedro Coelho, Alexa McDonald, Emma Woodberry, Robert C. Morris, Stephen J. Price, John Duncan, Thomas Santarius, John Suckling, Michael G. Hart, Yaara Erez. Tumour-infiltrated cortex participates in large-scale cognitive circuits medRxiv 2022.12.19.22283690.
<https://doi.org/10.1101/2022.12.19.22283690>
3. Crone N.E., Sinai A. & Korzeniewska A. (2006). High-frequency gamma oscillations and human brain mapping with electrocorticography. Progress in Brain Research 159, 275-295.
[https://doi.org/10.1016/S0079-6123\(06\)59019-3](https://doi.org/10.1016/S0079-6123(06)59019-3)
4. Alexander S. Atalay, Matteo Fecchioa, Brian L. Edlowa. Visualizing effective connectivity in the human brain. Volume 18, Issue 4, 1254-1256.
[https://www.brainstimjnl.com/article/S1935-861X\(25\)00266-9/fulltext](https://www.brainstimjnl.com/article/S1935-861X(25)00266-9/fulltext)
5. Arno Delorme, Muse Brain Display project.
https://github.com/arnodelorme/muse_brain_display
6. MATLAB's File Exchange.
https://www.mathworks.com/matlabcentral/fileexchange/?s_tid=gn_mlc_fx_file_s
7. MATLAB's App Designer.
<https://www.mathworks.com/products/matlab/app-designer.html#>
8. Complete Graph, Wikipedia.
https://en.wikipedia.org/wiki/Complete_graph

9. Electrocorticography, Wikipedia.

<https://en.wikipedia.org/wiki/Electrocorticography>

10. Large-scale brain network, Wikipedia.

https://en.wikipedia.org/wiki/Large-scale_brain_network

11. Frontoparietal network, Wikipedia.

https://en.wikipedia.org/wiki/Frontoparietal_network

12. Executive functions, Wikipedia.

https://en.wikipedia.org/wiki/Executive_functions

13. Jean-Philippe Lachaux, Eugenio Rodriguez, Jacques Martinerie, and Francisco J. Varela. Measuring Phase Synchrony in Brain Signals. Human Brain Mapping 8:194-208 (1999)

<https://pubmed.ncbi.nlm.nih.gov/10619414/>

14. Visualization of brain mapping with electrical stimulation and recordings of brain activity. Project number 112, Year 2025.

<https://engineering.biu.ac.il/node/13358>

11. נספחים

הקוד: (נכתב ב'Code View')

```
classdef PLView < matlab.apps.AppBase
    % Properties that correspond to app components
    properties (Access = public)
        UIFigure                               matlab.ui.Figure
        GridLayout                             matlab.ui.container.GridLayout
        LeftPanel                              matlab.ui.container.Panel
        GridLayout2                            matlab.ui.container.GridLayout
        TestConditionDropDownLabel            matlab.ui.control.Label
        SelectedElectrodeFrequencyLabel      matlab.ui.control.Label
        ElectrodesLabelsSwitchLabel         matlab.ui.control.Label
        Logo                                    matlab.ui.control.Image
        ConditionDropDown                     matlab.ui.control.DropDown
        Freq1DropDown                         matlab.ui.control.DropDown
        Freq2DropDown                         matlab.ui.control.DropDown
        SelectedElectrodeRole                matlab.ui.container.ButtonGroup
        AmpButton                             matlab.ui.control.RadioButtons
        PhaseButton                           matlab.ui.control.RadioButtons
        ShowElectrodesLabelsSwitch          matlab.ui.control.Switch
        UploadMATFileButton                  matlab.ui.control.Button
        CenterPanel                           matlab.ui.container.Panel
        GridLayout3                           matlab.ui.container.GridLayout
        ProcessingLabel                      matlab.ui.control.Label
        DataLoadLamp                          matlab.ui.control.Lamp
        BrainAxes                             matlab.ui.control.UIAxes
        RightPanel                           matlab.ui.container.Panel
        GridLayout4                           matlab.ui.container.GridLayout
        CircleAxes                           matlab.ui.control.UIAxes
    end
    % Properties that correspond to apps with auto-reflow
    properties (Access = private)
        onePanelWidth = 576;
        twoPanelWidth = 768;
    end
    %%%
    % Public properties (might need external access)
    properties (Access = private)
        % PUBLIC DATA
        PatientData             % Raw loaded .mat file data
        PatientName              % String: current subject identifier (example: "2017_02")
        CurrentConditionData     % Processed ECoG data for the current test condition
        PLVMatrix                % Computed PLV connectivity matrix
        % ELECTRODE CONFIGURATION
        ElectrodeCoords          % Nx3 matrix: 3D electrode positions (x,y,z)
        NumElectrodes            % Integer: number of electrodes

        % PROCESSING PARAMETERS
        SamplingFrequency        % [Hz], default: 2000

        % VISUALIZATION STATE
        ShowLabels               % Boolean: electrode labels on/off
        ElectrodeRole             % Boolean: selected electrode role (phase / amp)
    end

    % Private properties
    properties (Access = private)
        % ELECTRODES LAB'S DATABASE
        % ↴ Should remain static in the lab's drive and updated for new patients.
        ElectrodesData = load("/mnt/jane_data/DataFromMoataz/Elecs_n79_w_network.mat"); %
    end
    Table
        % ELECTRODES DATA
        ElectrodeInfo             % Struct: containing all electrode information
        NetworkColorMap            % Dictionary (containers.map object): color mapping for
        different networks
        SelectedElectrodeIdx       % Integer: index of currently selected electrode (0 = none)
        InfoPanel                 % Handle: to information panel
        InfoText                   % Handle: to information text
    end
```

```

InfoPanelVisible           % Boolean: tracks if floating info panel is visible

% PROCESSING FLAGS
IsDataLoaded              % Boolean: tracks if valid data is available
NeedsReprocessing          % Boolean: tracks if PLV needs recalculation
IsProcessing               % Boolean: tracks if processing is in progress
LoadingTimer                % Array: timer for rotating brain animation

% 2D VISUALIZATION HANDLES
CircleElectrodeCoords      % 2D coordinates for circular layout
CircleLinesHandles          % Graphics handles for 2D connectivity lines
CircleElectrodeHandles      % Graphics handles for 2D electrode points
CircleElectrodeLabels       % Graphics handles for 2D electrode text labels

% 3D VISUALIZATION HANDLES
BrainModelHandle            % Graphics handle for 3D brain surface
BrainElectrodeHandles       % Graphics handles for 3D electrode points
BrainLinesHandles           % Graphics handles for 3D connectivity lines
BrainBackgroundLines        % Graphics handles for persistent grey background lines
BrainSelectedLines          % Graphics handles for selected electrode lines
BrainStrongestLine          % Graphics handle for strongest connection highlight
BrainElectrodeLabels        % Graphics handles for 3D electrode text labels

% VISUAL STYLING
cmap                         % Custom colormap for PLV visualization
ElectrodeColors              % Current color matrix for electrodes
ColorbarMax                  % Maximum value for colorbar scaling (120% of max PLV)

end
%%%%%%%%%%%%%%% FUNCTIONS
%%%%%%%%%%%%%%%
methods (Access = private)

%%%%%%%%%%%%%%% Graphics %%%%%%
% Custom gradient colormap
function customColormap(app)
try
    % Number of colors
    numColors = 256; % Default size

    % RGB gradient colors (from blue to white)
    blue    = [0.05, 0.05, 0.6];    % dark blue
    purple  = [0.5, 0.0, 0.6];     % deep purple
    red     = [0.8, 0.0, 0.1];     % red
    orange  = [1.0, 0.5, 0.0];     % orange
    white   = [1.0, 1.0, 1.0];     % white

    % Positions of these colors along the gradient (0 to 1)
    positions = [0, 0.35, 0.6, 0.85, 1];

    % Color order
    colors = [blue; purple; red; orange; white];

    % Interpolate each RGB channel over numColors points
    xInterp = linspace(0, 1, numColors);
    rInterp = interp1(positions, colors(:,1), xInterp, 'pchip');
    gInterp = interp1(positions, colors(:,2), xInterp, 'pchip');
    bInterp = interp1(positions, colors(:,3), xInterp, 'pchip');

    % Combine into final colormap
    app.cmap = [rInterp(:,1), gInterp(:,1), bInterp(:,1)];
catch
    app.cmap = parula(256); % Alternative colormap
end
end
% Create networks colormap
function createNetworkColorMap(app)
    % Get unique networks
    uniqueNetworks = unique(app.ElectrodeInfo.Networks, 'stable');
    numNetworks = length(uniqueNetworks);

    % Create distinct colors for each network
    if numNetworks <= 10
        % Predefined distinct colors
        distinctColors = [
            1.0, 0.0, 0.0;    % Red

```

```

        0.0, 0.8, 0.0;      % Green
        0.0, 0.0, 1.0;      % Blue
        1.0, 0.65, 0.0;    % Orange
        0.8, 0.0, 0.8;     % Magenta
        0.0, 0.8, 0.8;     % Cyan
        1.0, 1.0, 0.0;     % Yellow
        0.5, 0.0, 1.0;     % Purple
        0.0, 0.5, 0.0;     % Dark Green
        0.8, 0.4, 0.0;     % Brown
    ];
    networkColors = distinctColors(1:numNetworks, :);
else
    % If many networks
    networkColors = hsv(numNetworks);
end

% Create mapping dictionary
app.NetworkColorMap = containers.Map();
for i = 1:numNetworks
    app.NetworkColorMap(uniqueNetworks{i}) = networkColors(i, :);
end
end
% Replace the automatic layout behavior
function customLayoutControl(app, src, evt)
    % Only allow responsive layout changes if data is loaded
    if app.IsDataLoaded
        % Call the original auto-generated updateAppLayout method
        app.updateAppLayout(evt);
    else
        % Keep 2-column layout during startup
        app.GridLayout.ColumnWidth = {200, '1x', 0};
        drawnow;
    end
end
% Setup loading animation UI changes
function setupLoadingAnimation(app)
    app.IsProcessing = true;

    % Hide UI controls during processing
    app.RightPanel.Visible = 'off'; % Hide the right panel
    app.ConditionDropDown.Enable = 'off';
    app.Freq1DropDown.Enable = 'off';
    app.Freq2DropDown.Enable = 'off';
    app.SelectedElectrodeRole.Enable = 'off';
    app.ShowElectrodesLabelsSwitch.Enable = 'off';

    % Show processing status
    app.DataLoadLamp.Color = 'yellow';
    app.ProcessingLabel.FontColor = 'yellow';
    app.ProcessingLabel.Text = " Processing...";

    % Clear any existing electrodes and connections
    if ~isempty(app.BrainElectrodeHandles)
        delete(app.BrainElectrodeHandles);
        app.BrainElectrodeHandles = gobjects(0);
    end
    if ~isempty(app.BrainLinesHandles)
        delete(app.BrainLinesHandles);
        app.BrainLinesHandles = gobjects(0);
    end
    app.updateBrainModel();
    drawnow;
end
% Process data with background animation
function processDataWithAnimation(app, filePath, file)
    % Start rotation timer
    app.LoadingTimer = timer('ExecutionMode', 'fixedRate', 'Period', 0.05, ...
        'TimerFcn', @(~,~) app.continuousRotation());
    start(app.LoadingTimer);

    try
        % Data processing
        app.PatientData = load(filePath);
        file = char(file); % convert to char array
        app.PatientName = file(end-10:end-4); % example: "2017_02"
    end
end

```

```

        app.createElectrodeInfo();
        app.updateCurrentConditionData();
        % Only proceed if data is valid
        if ~isempty(app.CurrentConditionData)
            app.processAndDisplayData();

            % Processing complete - stop animation
            app.stopLoadingAnimation();

            % Switch to 3-column layout
            app.IsDataLoaded = true;
            app.RightPanel.Visible = 'on';
            app.GridLayout.ColumnWidth = {200, '1x', '0.5x'}; % Show right panel
(2D)

            % Success messages
            uialert(app.UIFigure, 'Data loaded successfully!', 'Success', 'Icon',
'success');
            app.setInfoPanelVisible(true); % Show info panel
            app.updateInfoPanel(0); % (0) = General info
        else
            app.stopLoadingAnimation();
            uialert(app.UIFigure, 'Data loaded but no valid ECoG data found. Check
data structure.', 'Warning', 'Icon', 'warning');
        end

        catch ME
            app.stopLoadingAnimation();
            rethrow(ME);
        end
    end
    % Continuous rotation function for timer
    function continuousRotation(app)
        persistent rotationAngle;
        if isempty(rotationAngle)
            rotationAngle = 0;
        end

        if isvalid(app.BrainAxes) && app.IsProcessing
            view(app.BrainAxes, rotationAngle, 20);
            rotationAngle = rotationAngle + 3; % 3 degrees per 50ms
            if rotationAngle >= 360
                rotationAngle = 0;
            end
            drawnow limitrate;
        end
    end
    % Stop loading animation
    function stopLoadingAnimation(app)
        app.IsProcessing = false;

        % Stop timer
        if ~isempty(app.LoadingTimer) && isvalid(app.LoadingTimer)
            stop(app.LoadingTimer);
            delete(app.LoadingTimer);
            app.LoadingTimer = [];
        end

        % Re-enable UI controls
        app.ConditionDropDown.Enable = 'on';
        app.Freq1DropDown.Enable = 'on';
        app.Freq2DropDown.Enable = 'on';
        app.SelectedElectrodeRole.Enable = 'on';
        app.ShowElectrodesLabelsSwitch.Enable = 'on';
        app.DataLoadLamp.Color = 'green';
        app.ProcessingLabel.Text = "";
    end
    % Convert RGB values to hex color string for HTML
    function hexColor = rgbToHex(app, rgbColor)
        % Convert RGB [0-1] to hex string
        rgbColor = max(0, min(1, rgbColor)); % Clamp to [0,1]
        r = round(rgbColor(1) * 255);
        g = round(rgbColor(2) * 255);
        b = round(rgbColor(3) * 255);
        hexColor = sprintf('#%02X%02X%02X', r, g, b);
    end

```

```

    end

    % Get colorbar color for a PLV value
    function hexColor = getPLVColor(app, plvValue)
        if isempty(app.PLVMatrix) || isnan(plvValue)
            hexColor = '#696969'; % Grey for invalid values
            return;
        end

        % Normalize PLV value to the colorbar range (0 to maxPLV * 1.2)
        normVal = plvValue / app.ColorbarMax; % This will give values 0 to ~0.83 for
real PLV scores
        normVal = max(0, min(1, normVal)); % Clamp to [0,1]

        % Map to colormap index
        colorIdx = max(1, round(normVal * (size(app.cmap, 1) - 1)) + 1);
        colorIdx = min(colorIdx, size(app.cmap, 1));

        % Convert to hex
        hexColor = app.rgbToHex(app.cmap(colorIdx, :));
    end

    % Get network color as hex
    function hexColor = getNetworkColor(app, networkName)
        if isKey(app.NetworkColorMap, networkName)
            rgbColor = app.NetworkColorMap(networkName);
            hexColor = app.rgbToHex(rgbColor);
        else
            hexColor = '#808080'; % Gray for unknown networks
        end
    end
    %%%%%% Processing %%%%%%
    % Setup electrode data structure
    % ↳ Containing: Names, Coordinates, Network (+Color), Strongest Connection
    function createElectrodeInfo(app)

        % Initialize electrode info structure
        app.ElectrodeInfo = struct();

        % Get selected patient name from uploaded .mat file
        patientName = app.PatientName;

        try
            % Access table data
            electrodeData = app.ElectrodesData.elecs_n79;
            if ismember('subject', electrodeData.Properties.VariableNames)
                subjectData = electrodeData.subject;

                % Create patient mask
                if iscell(subjectData)
                    patientMask = strcmp(subjectData, patientName);
                else
                    patientMask = string(subjectData) == string(patientName);
                end
            else
                error('subject column not found in elecs_n79 table');
            end

            % Check if any electrodes were found for the subject
            if ~any(patientMask)
                error('No electrodes found for patient "%s". Check available subjects
above.', patientName);
            end

            % Filter out reference channels (contact == 4) early
            if ismember('contact', electrodeData.Properties.VariableNames)
                contacts = electrodeData.contact(patientMask);
                if iscell(contacts)
                    nonRefMask = cellfun(@(x) x ~= 4, contacts);
                else
                    nonRefMask = contacts ~= 4;
                end
            else
                % If no contact column, assume all are valid
                nonRefMask = true(sum(patientMask), 1);
            end
        end
    end

```

```

    end

    % Combine masks: patient AND non-reference
    combinedMask = false(size(patientMask));
    combinedMask(patientMask) = nonRefMask;
    % Extract coordinates
    x = electrodeData.x_MNI_coord(combinedMask);
    y = electrodeData.y_MNI_coord(combinedMask);
    z = electrodeData.z_MNI_coord(combinedMask);

    % Extract electrode names
    if ismember('channel', electrodeData.Properties.VariableNames)
        channelData = electrodeData.channel(combinedMask);
    elseif ismember('elec_name', electrodeData.Properties.VariableNames)
        channelData = electrodeData.elec_name(combinedMask);
    else
        % Generate default names
        channelData = cell(sum(combinedMask), 1);
        for i = 1:sum(combinedMask)
            channelData{i} = sprintf('E%d', i);
        end
    end
    % Ensure names are in cell array format
    if iscell(channelData)
        app.ElectrodeInfo.Names = channelData;
    else
        app.ElectrodeInfo.Names = cellstr(string(channelData));
    end

    % Extract networks
    if ismember('Network', electrodeData.Properties.VariableNames)
        networkData = electrodeData.Network(combinedMask);
    else
        % Generate default networks
        networkData = cell(sum(combinedMask), 1);
        for i = 1:sum(combinedMask)
            networkData{i} = 'Unknown';
        end
    end
    % Ensure networks are in cell array format
    if iscell(networkData)
        app.ElectrodeInfo.Networks = networkData;
    else
        app.ElectrodeInfo.Networks = cellstr(string(networkData));
    end

    % Store coordinates
    app.ElectrodeCoords = [x, y, z];
    app.NumElectrodes = size(app.ElectrodeCoords, 1);

    % Initialize electrode strongest connection structure
    app.ElectrodeInfo.StrongestConnections = struct(...,
        'PLVScore', num2cell(zeros(app.NumElectrodes, 1)), ...
        'PairIdx', num2cell(zeros(app.NumElectrodes, 1)), ...
        'PairName', cell(app.NumElectrodes, 1), ...
        'PairNetwork', cell(app.NumElectrodes, 1));
    % Assign colors to electrodes based on their networks
    app.ElectrodeInfo.NetworkColors = zeros(app.NumElectrodes, 3);
    app.createNetworkColorMap();
    for i = 1:app.NumElectrodes
        networkName = app.ElectrodeInfo.Networks(i);
        if isKey(app.NetworkColorMap, networkName)
            app.ElectrodeInfo.NetworkColors(i, :) =
        app.NetworkColorMap(networkName);
        else
            app.ElectrodeInfo.NetworkColors(i, :) = [0.5, 0.5, 0.5]; % Gray
        end
    end

    catch ME
        fprintf('Error in createElectrodeInfo: %s\n', ME.message);
        rethrow(ME);
    end
end

```

```

% Prepare the current test condition data
function updateCurrentConditionData(app)
    if isempty(app.PatientData)
        return;
    end
    currentCondition = app.ConditionDropDown.Value;
    dataField = '';
    % Determine how many samples to trim (from Lab's instructions)
    switch currentCondition
        case 'Rest'
            dataField = 'all_rest';
            trimStartSamples = app.SamplingFrequency * 1; % 1 second from start
            trimEndSamples = app.SamplingFrequency * 1; % 1 second from end
        case 'Easy Task (CountF)'
            dataField = 'all_countF';
            trimStartSamples = app.SamplingFrequency * 1; % 1 second from start
            trimEndSamples = app.SamplingFrequency * 1; % 1 second from end
        case 'Hard Task (Alt)'
            dataField = 'all_alt';
            trimStartSamples = app.SamplingFrequency * 3; % 3 seconds from start
            trimEndSamples = app.SamplingFrequency * 1; % 1 second from end
    end

    if isfield(app.PatientData.data_all, dataField)
        trialsData = app.PatientData.data_all.(dataField);

        % Combine trials into one 'long' trial
        % Assuming each trial is n_electrodes x time:
        %
        %           EL1 [ trial           trial ]
        % combinedData = ... [ 1   + ... +   n   ]
        %           ELn [ data           data ]
        %
        combinedData = [];

        % Check data type
        if iscell(trialsData)
            % Original cell array approach
            for i = 1:numel(trialsData)
                trial = trialsData(i);
                % Only trim if trial is long enough
                if size(trial, 2) > (trimStartSamples + trimEndSamples)
                    trial = trial(:, trimStartSamples+1:end-trimEndSamples);
                end
                combinedData = [combinedData, trial];
            end
        elseif isnumeric(trialsData)
            % Direct numeric array
            combinedData = trialsData;
        else
            error('Unexpected data structure type: %s', class(trialsData));
        end
        % Filter data to include only valid electrodes based on ElectrodesData
        % Extract channel information to map raw data indices
        try
            electrodeData = app.ElectrodesData.elecs_n79;

            % Create patient mask for current patient
            if iscell(electrodeData.subject)
                patientMask = strcmp(electrodeData.subject, app.PatientName);
            else
                patientMask = string(electrodeData.subject) ==
string(app.PatientName);
            end

            % Get contact information (position within stripe, 1-4)
            if ismember('contact', electrodeData.Properties.VariableNames)
                allContacts = electrodeData.contact(patientMask);
            else
                % If no contact info, assume all are valid (not reference)
                allContacts = ones(sum(patientMask), 1);
            end

            % Get channel indices for mapping to raw data
            if ismember('channel', electrodeData.Properties.VariableNames)

```

```

        allChannels = electrodeData.channel(patientMask);
    else
        % Fallback: use sequential indices
        allChannels = (1:sum(patientMask))';
    end

    % Filter out reference channels (contact == 4)
    validMask = true(size(allContacts));
    for i = 1:length(allContacts)
        contact = allContacts(i);
        if iscell(allContacts)
            contact = contact{1};
        end
        if contact == 4
            validMask(i) = false;
        end
    end

    % Get valid channel indices for data extraction
    validChannels = allChannels(validMask);
    if iscell(validChannels)
        validChannels = cell2mat(validChannels);
    end

    % Extract only valid (non-reference) channels from raw data
    if ~isempty(validChannels) && max(validChannels) <= size(combinedData,
1)
        app.CurrentConditionData = combinedData(validChannels, :);
    else
        app.CurrentConditionData = combinedData;
        warning('Could not map electrode channels, using all data');
    end
    catch ME
        warning(ME.identifier, 'Error filtering electrode data: %s. Using all
channels.', ME.message);
        app.CurrentConditionData = combinedData;
    end
    else
        app.CurrentConditionData = [];
        uialert(app.UIFigure, ['No data found for ' currentCondition '.'],
'Warning', 'Icon', 'warning');
    end
end
% Start processing and update display
function processAndDisplayData(app)
    % Step 1: Process data (frequency filtering, PLV computation)
    app.processECOGData(app.CurrentConditionData, ...
        app.Freq1DropDown.Value, ...
        app.Freq2DropDown.Value);

    % Step 2: Update the graphs
    app.updateBrainModel();
    app.updateCircularGraph();

    % Step 3: Update info panel
    app.updateInfoPanel(0); % (0) = Show general patient info

end
% ECoG Data Processing and PLV Calculation
function processECOGData(app, ecogData, freqBand1Str, freqBand2Str)
    % Get sampling frequency from app properties
    Fs = app.SamplingFrequency;

    numElectrodes = app.NumElectrodes;
    % Initiate plvMatrix
    plvMatrix = zeros(numElectrodes, numElectrodes);
    % Convert string frequency band names to numerical ranges [low_freq,
high_freq]
    [band1_low, band1_high] = app.getFrequencyBandLimits(freqBand1Str);
    [band2_low, band2_high] = app.getFrequencyBandLimits(freqBand2Str);
    % If Freq2 is lower than Freq1, automatically swap them
    if band2_low < band1_low
        % Change the labels order in left menu
        temp = app.Freq1DropDown.Value;
        app.Freq1DropDown.Value = app.Freq2DropDown.Value;
        app.Freq2DropDown.Value = temp;
    end
end

```

```

app.Freq2DropDown.Value = temp;
% Switch values
temp = band1_low;
band1_low = band2_low;
band2_low = temp;
temp = band1_high;
band1_high = band2_high;
band2_high = temp;
% Notify the user
uialert(app.UIFigure, 'Frequencies automatically swapped to maintain
lower-higher order', 'Info', 'Icon', 'info');
end
% Loop through all electrode pairs to compute connectivity strength (PLV)
for electrode1 = 1:numElectrodes
    for electrode2 = 1:numElectrodes
        if electrode1 ~= electrode2
            % Extract signals for current electrode pair
            phase_signal = ecogData(electrode1, :); % Electrode 1 Phase
frequency
            amp_signal = ecogData(electrode2, :); % Electrode 2 Amplitude
envelope
            % 1. Phase frequency - Filter signal for freqBand1 (LOWER
frequency band)
            try
                % Apply bandpass filter
                filtered_signal_band1_phase = bandpass(phase_signal,
[band1_low band1_high], Fs);
                catch ME
                    try
                        warning('ECOGAnalysisApp:FilterErrorBand1', 'Error
filtering signal for band 1 (%s): %s. Using butter() and filtfilt().', freqBand1Str,
ME.message);
                        [b, a] = butter(4, [band1_low band1_high]/(Fs/2),
'bandpass'); % Order 4: -24 dB per octave
                        filtered_signal_band1_phase = filtfilt(b, a,
phase_signal);
                    catch ME
                        warning('ECOGAnalysisApp:FilterErrorBand1', 'Error
filtering signal for band 1 (%s): %s.', freqBand1Str, ME.message);
                        uialert(app.UIFigure, 'Signal Processing Toolbox is
missing. Please install and re-run the program.', 'Warning', 'Icon', 'warning');
                        return;
                    end
                end
                % 2. Amplitude envelope - Filter signal for freqBand2 (HIGHER
frequency band)
                try
                    filtered_signal_band2_amp = bandpass(amp_signal, [band2_low
band2_high], Fs);
                    catch ME
                        try
                            warning('ECOGAnalysisApp:FilterErrorBand2', 'Error
filtering signal for band 2 (%s): %s. Using butter() and filtfilt().', freqBand2Str,
ME.message);
                            [b, a] = butter(4, [band2_low band2_high]/(Fs/2),
'bandpass');
                            filtered_signal_band2_amp = filtfilt(b, a, amp_signal);
                        catch ME
                            warning('ECOGAnalysisApp:FilterErrorBand2', 'Error
filtering signal for band 2 (%s): %s.', freqBand2Str, ME.message);
                            uialert(app.UIFigure, 'Signal Processing Toolbox is
missing. Please install and re-run the program.', 'Warning', 'Icon', 'warning');
                            return;
                        end
                    end
                    % Ensure signals are not empty after filtering
                    if isempty(filtered_signal_band1_phase) ||
isempty(filtered_signal_band2_amp)
                        plvMatrix(electrode1, electrode2) = NaN;
                        continue;
                    end
                    % Compute Phase of the signal from freqBand1
                    % Phase of low frequency signal
                    try
                        hilbert_signal_band1 = hilbert(filtered_signal_band1_phase);

```

```

        catch
            % Manual hilbert function if 'Signal Processing Toolbox' is
not installed
            hilbert_signal_band1 = hilbert_alt(app,
filtered_signal_band1_phase);
            end
            low_phase = angle(hilbert_signal_band1);
            % Compute Amplitude Envelope of the signal from freqBand2
            % Amplitude of high frequency signal
            try
                hilbert_signal_band2_complex =
hilbert(filtered_signal_band2_amp);
                catch
                    hilbert_signal_band2_complex = hilbert_alt(app,
filtered_signal_band2_amp);
                    end
                    high_amp = abs(hilbert_signal_band2_complex); % Amplitude envelope
                    % Compute Phase of the high frequency Amplitude Envelope
                    % This involves applying the Hilbert transform to the amplitude
envelope itself.
                    try
                        hilbert_high_amp_envelope = hilbert(high_amp);
                        catch
                            hilbert_high_amp_envelope = hilbert_alt(app, high_amp);
                        end
                        high_phase_from_amp_envelope = angle(hilbert_high_amp_envelope);
                        % Phase Difference: between phase of band1 and phase of band2's
amplitude envelope
                        phaseDiff = high_phase_from_amp_envelope - low_phase;
                        % Calculate the PLV (Phase Locking Value)
                        PLV_result = abs(mean(exp(1i * phaseDiff), 'omitnan')); % Use
'omitnan' to ignore NaNs if any

                        % Save result in matrix
                        plvMatrix(electrode1, electrode2) = PLV_result;
                    else
                        plvMatrix(electrode1, electrode2) = NaN; % PLV with self is
undefined or 0
                    end
                end
            end

            % Store PLV Matrix in global variable
            app.PLVMATRIX = plvMatrix;

            % The PLV Matrix structure:
            %
            % p = Phase frequency (LOWER frequency band)
            % a = Amplitude envelope (HIGHER frequency band)
            %
            %      E1      E2      E3      E4
            % E1  x      1p-2a  1p-3a  1p-4a
            % E2  2p-1a  x      2p-3a  2p-4a
            % E3  3p-1a  3p-2a  x      3p-4a
            % E4  4p-1a  4p-2a  4p-3a  x
            %

            % Get maximum PLV for consistent scaling
            maxPLV = max(app.PLVMATRIX(:), [], 'omitnan');
            if ~isnan(maxPLV) && maxPLV > 0
                app.ColorbarMax = maxPLV * 1.2;
            else
                app.ColorbarMax = 1; % Default value
            end
            % Calculate strongest connections for each electrode (done after PLV
calculation)
            app.calculateStrongestConnections();

        end
    % Manual hilbert function if 'Signal Processing Toolbox' is not installed
    function y = hilbert_alt(app, x)
        N = length(x);
        X = fft(x);
        h = zeros(1, N);
        if mod(N,2) == 0

```

```

        h([1 N/2+1]) = 1;
        h(2:N/2) = 2;
    else
        h(1) = 1;
        h(2:(N+1)/2) = 2;
    end
    y = ifft(X .* h);
end
% Convert frequency band limits from string to numeric
function [low_freq, high_freq] = getFrequencyBandLimits(app, bandName)
switch bandName
    case 'Delta (0.5-4 Hz)'
        low_freq = 0.5;
        high_freq = 4;
    case 'Theta (4-8 Hz)'
        low_freq = 4;
        high_freq = 8;
    case 'Alpha (8-12 Hz)'
        low_freq = 8;
        high_freq = 12;
    case 'Beta (12-30 Hz)'
        low_freq = 12;
        high_freq = 30;
    case 'Gamma (30-70 Hz)'
        low_freq = 30;
        high_freq = 70;
    case 'High Gamma (70-250 Hz)'
        low_freq = 70;
        high_freq = 250;
    end
end
% Calculate strongest connections for each electrode
function calculateStrongestConnections(app)
if isempty(app.PLVMatrix)
    return;
end

% Find strongest connection for each electrode
for i = 1:app.NumElectrodes
    % Get PLV scores for current electrode (exclude self-connection)
    plvScores = app.PLVMatrix(i, :);
    plvScores(i) = NaN; % Exclude self

    % Find maximum PLV
    [maxPLV, maxIdx] = max(plvScores, [], 'omitnan');

    % Store result
    if ~isnan(maxPLV)
        app.ElectrodeInfo.StrongestConnections(i).PLVScore = maxPLV;
        app.ElectrodeInfo.StrongestConnections(i).PairIdx = maxIdx;
        app.ElectrodeInfo.StrongestConnections(i).PairName =
app.ElectrodeInfo.Names{maxIdx};
        app.ElectrodeInfo.StrongestConnections(i).PairNetwork =
app.ElectrodeInfo.Networks{maxIdx};
    end
end
%%%%%%% Info Panel %%%%%%
% Create information panel
function createInfoPanel(app)
    % Create a container panel for row 2
    infoPanelContainer = uipanel(app.GridLayout4, "BackgroundColor", [0.25 0.25
0.25]);
    infoPanelContainer.Layout.Row = 2;
    infoPanelContainer.Layout.Column = 1;
    infoPanelContainer.BorderType = 'none';

    % Create grid inside the container with padding column
    infoSubGrid = uigridlayout(infoPanelContainer);
    infoSubGrid.BackgroundColor = [0.25 0.25 0.25];
    infoSubGrid.ColumnWidth = {15, '1x'}; % 20px left padding + content
    infoSubGrid.RowHeight = {'1x'};
    infoSubGrid.Padding = [0 0 0 0]; % No extra padding from grid

    % Create a uilabel for text display

```

```

app.InfoPanel = uilabel(infoSubGrid, ...
    'Text', '', ...
    'FontSize', 16, ...
    'FontColor', 'white', ...
    'BackgroundColor', 'none', ... % No background
    'HorizontalAlignment', 'left', ...
    'VerticalAlignment', 'top', ...
    'WordWrap', 'on', ...
    'Interpreter', 'html', ...      % Enable HTML interpretation
    'Visible', 'off');

% Position it in the info panel's sub grid
app.InfoPanel.Layout.Row = 1;
app.InfoPanel.Layout.Column = 2; % Use content column
% Initialize visibility flag
app.InfoPanelVisible = false;
end

% Update information panel
function updateInfoPanel(app, selectedElectrodeIdx)
    if isempty(app.ElectrodeInfo)
        return;
    end

    % Generate info text content
    if app.ElectrodeRole == 0
        elec1Freq = "phase";
        elec2Freq = "amp";
    else
        elec1Freq = "amp";
        elec2Freq = "phase";
    end
    if nargin < 2 || selectedElectrodeIdx == 0
        % Show general patient information
        [globalMaxPLV, globalMaxIdx] = max(app.PLVMatrix(:), [], 'omitnan');
        [row, col] = ind2sub(size(app.PLVMatrix), globalMaxIdx);

        if ~isnan(globalMaxPLV)
            elec1Name = app.ElectrodeInfo.Names{row};
            elec2Name = app.ElectrodeInfo.Names{col};
            elec1Network = app.ElectrodeInfo.Networks{row};
            elec2Network = app.ElectrodeInfo.Networks{col};

            % Get dynamic colors
            plvColor = app.getPLVColor(globalMaxPLV);
            network1Color = app.getNetworkColor(elec1Network);
            network2Color = app.getNetworkColor(elec2Network);

            infoText = sprintf(['<font color="white"><b>PATIENT
INFO</b></font><br>' ...
                '<font
color="white">_____</font><br>' ...
                '<font color="cyan">%s</font><br>' ...
                '<font color="cyan">%d</font><br><br>' ...
                '<font color="white">Patient: </font><font
color="white">_____</font><br>' ...
                '<font color="white">Electrodes: </font><font
color="white">_____</font><br>' ...
                '<font color="white"><b>STRONGEST
CONNECTION</b></font><br>' ...
                '<font
color="white">_____</font><br>' ...
                '<font color="white">PLV: </font><font
color="%s">%.4f</font><br>' ...
                '<font color="white">%s (%s) ↔ %s (%s)</font><br>' ...
                '<font color="%s">%s</font><font color="white"> ↔
</font><font color="%s">%s</font>', ...
                app.PatientName, app.NumElectrodes, ...
                plvColor, globalMaxPLV, elec1Name, elec1Freq,
                network1Color, elec1Network, network2Color,
                elec2Name, elec2Freq, ...
                elec2Network);
            else
                infoText = sprintf(['<font color="white"><b>PATIENT
INFO</b></font><br>' ...
                    '<font
color="white">_____</font><br>' ...

```

```

        '<font color="white">Patient: </font><font
color="cyan">%s</font><br>' ...
        '<font color="white">Electrodes: </font><font
color="cyan">%d</font><br><br>' ...
        '<font color="red">No PLV data available</font>'],
...
        app.PatientName, app.NumElectrodes);
    end
else
    % Show selected electrode information
    coords = app.ElectrodeCoords(selectedElectrodeIdx, :);

    % Dynamically find strongest connection based on current role
    if app.ElectrodeRole == 0
        plvScores = app.PLVMATRIX(selectedElectrodeIdx, :); % Row
    else
        plvScores = app.PLVMATRIX(:, selectedElectrodeIdx)'; % Column
    end
    plvScores(selectedElectrodeIdx) = NaN; % Exclude self

    % Find maximum PLV for this electrode with current role
    [maxPLV, maxIdx] = max(plvScores, [], 'omitnan');

    % Get target electrode info
    if ~isnan(maxPLV)
        targetName = app.ElectrodeInfo.Names{maxIdx};
        targetNetwork = app.ElectrodeInfo.Networks{maxIdx};
    else
        targetName = 'None';
        targetNetwork = 'None';
        maxPLV = 0;
    end

    % Get dynamic colors
    selectedNetwork = app.ElectrodeInfo.Networks{selectedElectrodeIdx};
    plvColor = app.getPLVColor(maxPLV);
    selectedNetworkColor = app.getNetworkColor(selectedNetwork);
    targetNetworkColor = app.getNetworkColor(targetNetwork);

    infoText = sprintf(['<font color="white"><b>SELECTED
ELECTRODE</b></font><br>' ...
                    '<font color="white">_____</font><br>'
...
                    '<font color="yellow">%s</font><br>' ...
                    '<font color="white">Name: </font><font
color="%s">%s</font><br>' ...
                    '<font color="white">Network: </font><font
color="white">%.1f</font><br>' ...
                    '<font color="white">Coords:</font><br>' ...
                    '<font color="white"> X: </font><font
color="white">%.1f</font><br>' ...
                    '<font color="white"> Y: </font><font
color="white">%.1f</font><br>' ...
                    '<font color="white"> Z: </font><font
color="white">%.1f</font><br>' ...
                    '<font color="white"><b>STRONGEST
CONNECTION</b></font><br>' ...
                    '<font color="white">_____</font><br>'
...
                    '<font color="white">PLV: </font><font
color="%s">%.4f</font><br>' ...
                    '<font color="white">To: </font><font color="white">%s
(%s)</font><br>' ...
                    '<font color="white">Network: </font><font
color="%s">%s</font>'], ...
        app.ElectrodeInfo.Names{selectedElectrodeIdx}, ...
        selectedNetworkColor, selectedNetwork, ...
        coords(1), coords(2), coords(3), ...
        plvColor, maxPLV, targetName, elec2Freq, ...
        targetNetworkColor, targetNetwork);
end

% Update text content directly on the label
if ~isempty(app.InfoPanel) && isValid(app.InfoPanel)
    app.InfoPanel.Text = infoText;

```

```

    end
end
% Method to show/hide the floating info panel
function setInfoPanelVisible(app, visible)
    app.InfoPanelVisible = visible;

    if ~isempty(app.InfoPanel) && isvalid(app.InfoPanel)
        if visible
            app.InfoPanel.Visible = 'on';
        else
            app.InfoPanel.Visible = 'off';
        end
    end
end
%%%%%%%%%%%%% Main Graphs %%%%%%
% Update 3D Brain Model visualization
function updateBrainModel(app)

    % Complete reset
    cla(app.BrainAxes, 'reset');
    % Set background to black to prevent white axes flash
    app.BrainAxes.Color = [0 0 0]; % Black background
    rotate3d(app.BrainAxes,'on'); % Enable free mouse rotating
    %app.BrainAxes.Toolbar.Visible = 'off'; % Hide toolbar

    hold(app.BrainAxes, 'on');

    % Load and plot brain model
    try
        % Try multiple possible locations for the brain model
        meshFile = [];

        % Option 1: Try relative to the app file
        try
            meshFile = fullfile(fileparts(mfilename('fullpath')), 'head3d.mat');
        catch
        end

        % Option 2: Try in the resources folder for packaged apps
        if isempty(meshFile) || ~exist(meshFile, 'file')
            try
                meshFile = fullfile(ctfroot, 'head3d.mat');
            catch
            end
        end

        % Option 3: Try to find it anywhere on the path
        if isempty(meshFile) || ~exist(meshFile, 'file')
            meshFile = which('head3d.mat');
        end
        % Load the brain model if found
        if ~isempty(meshFile) && exist(meshFile, 'file')
            data3d = load('-mat', meshFile);
            cortexMesh = data3d.head3d.cortex.mesh;

            % Save brain handle
            app.BrainModelHandle = patch(app.BrainAxes, ...
                'Vertices', cortexMesh.vertices, ...
                'Faces', cortexMesh.faces, ...
                'FaceColor', [0.9 0.9 0.9], ...
                'EdgeColor', 'none', ...
                'FaceLighting', 'gouraud', ...
                'AmbientStrength', 0.3, ...
                'FaceAlpha', 0.1);
        else
            % Plot a 3D sphere if the brain model is unavailable
            [x,y,z] = sphere(20);
            app.BrainModelHandle = surf(app.BrainAxes, 50*x, 50*y, 50*z, ...
                'FaceColor', [0.9 0.9 0.9], 'EdgeColor', 'none', ...
                'FaceLighting', 'gouraud', 'AmbientStrength', 0.3, 'FaceAlpha',
0.1);
        end
    catch
        [x,y,z] = sphere(20);
        app.BrainModelHandle = surf(app.BrainAxes, 50*x, 50*y, 50*z, ...

```

```

    'FaceColor', [0.9 0.9 0.9], 'EdgeColor', 'none', ...
    'FaceLighting', 'gouraud', 'AmbientStrength', 0.3, 'FaceAlpha', 0.1);
end

% Axes settings
axis(app.BrainAxes, 'equal');
axis(app.BrainAxes, 'off');
view(app.BrainAxes, 3);
camlight(app.BrainAxes, 'right');

% Only add electrodes if data is loaded
if app.NumElectrodes > 0 && ~isempty(app.ElectrodeCoords)

    % Use network-based coloring
    if ~isempty(app.ElectrodeInfo)
        app.ElectrodeColors = app.ElectrodeInfo.NetworkColors;
    else % Blue
        app.ElectrodeColors = repmat([0 0 1], app.NumElectrodes, 1);
    end

    % Plot connectivity lines
    if ~isempty(app.PLVMatrix)
        if app.SelectedElectrodeIdx == 0
            % No electrode selected: draw background lines + highlight
strongest
            app.drawBackgroundLines3D();
            app.drawStrongestConnection3D();
        else
            % Electrode selected: draw background lines + selected electrode
lines
            app.drawBackgroundLines3D();
            app.drawSelectedElectrodeLines3D();
        end
    end

    % Plot electrodes with click functionality
app.BrainElectrodeHandles = gobjects(app.NumElectrodes,1);
app.BrainElectrodeLabels = gobjects(app.NumElectrodes,1);

    % Electrodes and Labels
for i = 1:app.NumElectrodes
    if i == app.SelectedElectrodeIdx
        markerSize = 300;
        edgeColor = 'yellow'; % Highlight selected
        lineWidth = 3;
    else
        markerSize = 200;
        edgeColor = 'black';
        lineWidth = 1;
    end

    % Create electrodes
    app.BrainElectrodeHandles(i) = scatter3(app.BrainAxes, ...
        app.ElectrodeCoords(i,1), app.ElectrodeCoords(i,2),
app.ElectrodeCoords(i,3), ...
        markerSize, app.ElectrodeColors(i,:), 'filled', ...
        'MarkerEdgeColor', edgeColor, 'LineWidth', lineWidth, ...
        'DisplayName', ['Electrode ' num2str(i)], ...
        'ButtonDownFcn', @(src, event) app.onElectrodeClicked(src, event),
...
        'PickableParts', 'all', ... % Ensure clickable
        'HitTest', 'on', 'UserData', i);

    % Create electrode labels
    app.BrainElectrodeLabels(i) = text(app.BrainAxes, ...
        app.ElectrodeCoords(i,1)*1.15, app.ElectrodeCoords(i,2)*1.15,
app.ElectrodeCoords(i,3)*1.15, ...
        app.ElectrodeInfo.Names{i}, 'FontSize', 14, 'FontWeight',
'normal', ...
        'Color', 'white', 'HorizontalAlignment', 'right',
'VerticalAlignment', 'bottom',...
        'Visible', app.ShowElectrodesLabelsSwitch.Value);
    end
end

```

```

hold(app.BrainAxes, 'off');

% Refresh info panel overlay if it should be visible
if app.InfoPanelVisible & ~isempty(app.ElectrodeInfo)
    app.updateInfoPanel(app.SelectedElectrodeIdx);
end

end
% Update 2D Circular Network visualization
function updateCircularGraph(app)

% Complete reset to remove axes
cla(app.CircleAxes, 'reset');
hold(app.CircleAxes, 'on');
% Only proceed if we have electrode data
if app.NumElectrodes == 0 || isempty(app.ElectrodeCoords)
    return;
end

% Places electrodes evenly around a circle
angles = linspace(0, 2*pi, app.NumElectrodes + 1);
angles(end) = [];

% Compute coordinates
radius = 0.7;
x = radius * cos(angles);
y = radius * sin(angles);
app.CircleElectrodeCoords = [x(:), y(:)];
% Plot connectivity lines FIRST (so they appear behind the electrodes)
if ~isempty(app.PLVMatrix)
    app.plotConnectivityLines2D();
end

% Plot electrodes
if ~isempty(app.ElectrodeInfo)
    % Use network colors
    electrodeColors = app.ElectrodeInfo.NetworkColors;

    app.CircleElectrodeHandles = gobjects(app.NumElectrodes, 1);
    app.CircleElectrodeLabels = gobjects(app.NumElectrodes, 1);

    for i = 1:app.NumElectrodes
        % Determine marker size based on selection
        if i == app.SelectedElectrodeIdx
            markerSize = 18;
            edgeColor = 'yellow';
            lineWidth = 3;
            labelColor = 'yellow';
        else
            markerSize = 14;
            edgeColor = 'black';
            lineWidth = 1;
            labelColor = 'white';
        end

        % Create electrodes
        app.CircleElectrodeHandles(i) = plot(app.CircleAxes, ...
            app.CircleElectrodeCoords(i,1), app.CircleElectrodeCoords(i,2),
            'o', ...
            'MarkerSize', markerSize, 'MarkerFaceColor', electrodeColors(i,:),
            ...
            'MarkerEdgeColor', edgeColor, 'LineWidth', lineWidth, ...
            'ButtonDownFcn', @(src, event) app.onElectrodeClicked(src, event),
            ...
            'PickableParts', 'visible', ...
            'HitTest', 'on', 'UserData', i);

        % Create electrode labels
        app.CircleElectrodeLabels(i) = text(app.CircleAxes, ...
            app.CircleElectrodeCoords(i,1)*1.2,
            app.CircleElectrodeCoords(i,2)*1.2, ...
            app.ElectrodeInfo.Names{i}, 'FontSize', 14, 'FontWeight',
            'normal', ...
            'Color', labelColor, 'HorizontalAlignment', 'center', ...
            'Visible', app.ShowElectrodesLabelsSwitch.Value);
    end
end

```

```

        end
    end

    % Completely hide axes
    axis(app.CircleAxes, 'equal');
    axis(app.CircleAxes, 'off');
    xlim(app.CircleAxes, [-1.2, 1.2]); % Add padding around the circle
    ylim(app.CircleAxes, [-1.2, 1.2]);
    hold(app.CircleAxes, 'off');
    % Plot colorbar
    colormap(app.CircleAxes, app.cmap);
    c = colorbar(app.CircleAxes, 'southoutside', 'Color', 'white');
    c.Label.String = 'PLV Score';
    c.Label.Color = 'white';
    % Set colorbar range
    try
        clim(app.CircleAxes, [0, app.ColorbarMax]); % MATLAB 2024b
    catch
        caxis(app.CircleAxes, [0, app.ColorbarMax]); % MATLAB 2022b
    end

    % Refresh info panel overlay if it should be visible
    if app.InfoPanelVisible && ~isempty(app.ElectrodeInfo)
        app.updateInfoPanel(app.SelectedElectrodeIdx);
    end

end
%%%%%%%%%%%%% Electrodes & Lines %%%%%%%%%%%%%%
% Draw all connections in grey for 3D
% (called once when no electrode is selected)
function drawBackgroundLines3D(app)
    % Clear any existing background lines
    if ~isempty(app.BrainBackgroundLines)
        for i = 1:length(app.BrainBackgroundLines)
            if isvalid(app.BrainBackgroundLines(i))
                delete(app.BrainBackgroundLines(i));
            end
        end
    end

    if isempty(app.PLVMATRIX)
        app.BrainBackgroundLines = gobjects(0);
        return;
    end

    % Ensure we're working with the correct axes
    hold(app.BrainAxes, 'on');

    % Initialize parameters
    connMatrix = app.PLVMATRIX;
    numElectrodes = app.NumElectrodes;
    lineHandles = gobjects(numElectrodes^2, 1);
    idx = 1;

    % Draw all connections in grey with arched lines
    for i = 1:numElectrodes
        for j = i+1:numElectrodes
            val = connMatrix(i,j);

            if ~isnan(val) && val > 0
                % Create 3D arched path
                point1 = app.ElectrodeCoords(i,:);
                point2 = app.ElectrodeCoords(j,:);

                % Use different arch heights based on distance for variety
                distance = norm(point2 - point1);
                archHeight = 0.2 + 0.1 * (distance / 100); % Scale arch with
distance

                [xArch, yArch, zArch] = app.create3DArch(point1, point2, 30,
archHeight);

                lineHandles(idx) = plot3(app.BrainAxes, xArch, yArch, zArch, ...
                    'Color', [0.5, 0.5, 0.5], 'LineWidth', 0.2, 'LineSmoothing',
'on');
            end
        end
    end
end

```

```

        idx = idx + 1;
    end
end

% Store handles
app.BrainBackgroundLines = lineHandles(1:idx-1);
end

% Draw strongest connection highlight for 3D
function drawStrongestConnection3D(app)
    % Clear previous strongest line
    if ~isempty(app.BrainStrongestLine) && isvalid(app.BrainStrongestLine)
        delete(app.BrainStrongestLine);
    end

    if isempty(app.PLVMATRIX)
        app.BrainStrongestLine = gobjects(0);
        return;
    end

    % Ensure we're working with the correct axes
    hold(app.BrainAxes, 'on');

    % Find global strongest connection
    [globalMaxPLV, globalMaxIdx] = max(app.PLVMATRIX(:, [], 'omitnan'));
    [strongestRow, strongestCol] = ind2sub(size(app.PLVMATRIX), globalMaxIdx);

    if ~isnan(globalMaxPLV)
        minWidth = 0.2;
        maxWidth = 3;

        % Dynamic color and width for strongest connection
        normVal = globalMaxPLV / app.ColorbarMax;
        normVal = max(0, min(1, normVal));
        colorIdx = max(1, round(normVal * (size(app.cmap, 1) - 1)) + 1);
        colorIdx = min(colorIdx, size(app.cmap, 1));
        colorLine = app.cmap(colorIdx, :);
        maxPLV = max(app.PLVMATRIX(:, [], 'omitnan'));
        lineWidthNorm = globalMaxPLV / maxPLV;
        lineWidth = minWidth + lineWidthNorm * (maxWidth - minWidth);

        % Create 3D arched path for strongest connection
        point1 = app.ElectrodeCoords(strongestRow,:);
        point2 = app.ElectrodeCoords(strongestCol,:);

        % Make strongest connection more prominent with higher arch
        [xArch, yArch, zArch] = app.create3DArch(point1, point2, 50, 0.5);

        app.BrainStrongestLine = plot3(app.BrainAxes, xArch, yArch, zArch, ...
            'Color', colorLine, 'LineWidth', lineWidth, 'LineSmoothing', 'on');
    end
end

% Draw only selected electrode connections for 3D
function drawSelectedElectrodeLines3D(app)
    % Clear previous selected lines
    if ~isempty(app.BrainSelectedLines)
        for i = 1:length(app.BrainSelectedLines)
            if isvalid(app.BrainSelectedLines(i))
                delete(app.BrainSelectedLines(i));
            end
        end
    end

    if isempty(app.PLVMATRIX) || app.SelectedElectrodeIdx == 0
        app.BrainSelectedLines = gobjects(0);
        return;
    end

    % Ensure we're working with the correct axes
    hold(app.BrainAxes, 'on');

    % Initialize parameters
    connMatrix = app.PLVMATRIX;

```

```

numElectrodes = app.NumElectrodes;
selectedElec = app.SelectedElectrodeIdx;
minWidth = 0.2;
maxWidth = 3;

lineHandles = gobjects(numElectrodes, 1);
idx = 1;

% Draw selected electrode connections with arched lines
for j = 1:numElectrodes
    if j ~= selectedElec
        % Get PLV value based on selected electrode role
        if app.ElectrodeRole == 0
            val = connMatrix(selectedElec, j);
        else
            val = connMatrix(j, selectedElec);
        end

        if ~isnan(val) && val > 0
            % Dynamic color and width
            normVal = val / app.ColorbarMax;
            normVal = max(0, min(1, normVal));
            colorIdx = max(1, round(normVal * (size(app.cmap, 1) - 1)) + 1);
            colorIdx = min(colorIdx, size(app.cmap, 1));
            colorLine = app.cmap(colorIdx, :);
            maxPLV = max(connMatrix(:, [], 'omitnan'));
            lineWidthNorm = val / maxPLV;
            lineWidth = minWidth + lineWidthNorm * (maxWidth - minWidth);

            % Create 3D arched path
            point1 = app.ElectrodeCoords(selectedElec,:);
            point2 = app.ElectrodeCoords(j,:);

            % Vary arch height based on PLV strength for visual hierarchy
            archHeight = 0.2 + 0.3 * lineWidthNorm; % Higher PLV = higher arch
            [xArch, yArch, zArch] = app.create3DArch(point1, point2, 40,
archHeight);

            lineHandles(idx) = plot3(app.BrainAxes, xArch, yArch, zArch, ...
                'Color', colorLine, 'LineWidth', lineWidth, 'LineSmoothing',
'on');
            idx = idx + 1;
        end
    end
end

% Store handles
app.BrainSelectedLines = lineHandles(1:idx-1);
end

% Separate function for 2D connectivity lines
function plotConnectivityLines2D(app)
    % Clear previous connectivity lines
    if ~isempty(app.CircleLinesHandles)
        for i = 1:length(app.CircleLinesHandles)
            if isvalid(app.CircleLinesHandles(i))
                delete(app.CircleLinesHandles(i));
            end
        end
    end

    if isempty(app.PLVMatrix)
        app.CircleLinesHandles = gobjects(0);
        return;
    end

    connMatrix = app.PLVMatrix;
    numElectrodes = app.NumElectrodes;

    % Get maximum PLV for scaling
    maxPLV = max(connMatrix(:, [], 'omitnan'));
    if isnan(maxPLV) || maxPLV == 0
        app.CircleLinesHandles = gobjects(0);
        return;
    end

```

```

minWidth = 0.2;
maxWidth = 4;

lineHandles = gobjects(numElectrodes^2, 1);
idx = 1;

if app.SelectedElectrodeIdx == 0
    % NO ELECTRODE SELECTED: Show all connections in grey + global strongest
connection highlighted

    % Find global strongest connection (ALWAYS from original matrix, ignore
selected electrode role)
    [globalMaxPLV, globalMaxIdx] = max(app.PLVMMatrix(:, [], 'omitnan');
[strongestRow, strongestCol] = ind2sub(size(app.PLVMMatrix), globalMaxIdx);

    % Plot all connections
    for i = 1:numElectrodes
        for j = i+1:numElectrodes
            % For display purposes, choose connection value based on selected
electrode role
            if app.ElectrodeRole == 0
                val = connMatrix(i,j);
            else
                val = connMatrix(j,i);
            end

            if ~isnan(val) && val > 0
                % Check if this is the strongest connection (based on original
matrix position)
                isStrongest = (i == strongestRow && j == strongestCol) || ...
(i == strongestCol && j == strongestRow);

                if isStrongest
                    % Highlight strongest connection with its ORIGINAL PLV
                    originalPLV = app.PLVMMatrix(strongestRow, strongestCol);
                    normVal = originalPLV / app.ColorbarMax;
                    normVal = max(0, min(1, normVal));
                    colorIdx = max(1, round(normVal * (size(app.cmap, 1) - 1)) +
1);

                    colorIdx = min(colorIdx, size(app.cmap, 1));
                    colorLine = app.cmap(colorIdx, :);
                    lineWidthNorm = originalPLV / maxPLV;
                    lineWidth = minWidth + lineWidthNorm * (maxWidth -
minWidth);
                else
                    % All other connections in grey with min width
                    colorLine = [0.5, 0.5, 0.5]; % Grey
                    lineWidth = minWidth;
                end

                % Get 2D coordinates and plot
                x = [app.CircleElectrodeCoords(i,1),
app.CircleElectrodeCoords(j,1)];
                y = [app.CircleElectrodeCoords(i,2),
app.CircleElectrodeCoords(j,2)];

                lineHandles(idx) = plot(app.CircleAxes, x, y, ...
                    'Color', colorLine, 'LineWidth', lineWidth, 'LineStyle',
'-' );
                idx = idx + 1;
            end
        end
    end

else
    % ELECTRODE SELECTED: Show selected electrode connections + background
connections
    selectedElec = app.SelectedElectrodeIdx;

    % First, plot all background connections between unselected electrodes in
grey
    for i = 1:numElectrodes
        for j = i+1:numElectrodes

```

```

        % Skip connections involving the selected electrode
        if i ~= selectedElec && j ~= selectedElec
            % Use original matrix values for background connections
            % (ignore selected electrode role)
            val = connMatrix(i,j);

            if ~isnan(val) && val > 0
                % Grey background connections with minimum width
                colorLine = [0.5, 0.5, 0.5];
                lineWidth = minWidth;

                % Get 2D coordinates and plot
                x = [app.CircleElectrodeCoords(i,1),
                app.CircleElectrodeCoords(j,1)];
                y = [app.CircleElectrodeCoords(i,2),
                app.CircleElectrodeCoords(j,2)];

                lineHandles(idx) = plot(app.CircleAxes, x, y, ...
                    'Color', colorLine, 'LineWidth', lineWidth,
                    'LineStyle', '-');
                idx = idx + 1;
            end
        end
    end

    % Then, plot selected electrode connections with dynamic colors/widths
    for j = 1:numElectrodes
        if j ~= selectedElec
            % Get PLV value based on selected electrode role
            if app.ElectrodeRole == 0
                val = connMatrix(selectedElec, j); % Row selectedElec
            else
                val = connMatrix(j, selectedElec); % Column selectedElec
            end

            if ~isnan(val) && val > 0
                % Dynamic color and width based on PLV value
                normVal = val / app.ColorbarMax;
                normVal = max(0, min(1, normVal));
                colorIdx = max(1, round(normVal * (size(app.cmap, 1) - 1)) +
                1);
                colorIdx = min(colorIdx, size(app.cmap, 1));
                colorLine = app.cmap(colorIdx, :);

                lineWidthNorm = val / maxPLV;
                lineWidth = minWidth + lineWidthNorm * (maxWidth - minWidth);

                % Get 2D coordinates and plot
                x = [app.CircleElectrodeCoords(selectedElec,1),
                app.CircleElectrodeCoords(j,1)];
                y = [app.CircleElectrodeCoords(selectedElec,2),
                app.CircleElectrodeCoords(j,2)];

                lineHandles(idx) = plot(app.CircleAxes, x, y, ...
                    'Color', colorLine, 'LineWidth', lineWidth, 'LineStyle',
                    '-');
                idx = idx + 1;
            end
        end
    end

    app.CircleLinesHandles = lineHandles(1:idx-1);
end
% Create 3D arched path between two points
function [xArch, yArch, zArch] = create3DArch(app, point1, point2, numPoints,
archHeight)
    % Create a smooth 3D arch between two electrode points
    %
    % Inputs:
    %   point1, point2: [x, y, z] coordinates of start and end points
    %   numPoints: number of points along the arch (default: 50)
    %   archHeight: relative height of the arch (default: 0.3)

```

```

if nargin < 4
    numPoints = 50;
end
if nargin < 5
    archHeight = 0.3; % 30% of distance between points
end

% Calculate the midpoint and distance
midpoint = (point1 + point2) / 2;
distance = norm(point2 - point1);

% Create arch in local coordinate system
t = linspace(0, 1, numPoints);

% Find a vector perpendicular to the line (pointing outward from brain center)
brainCenter = [0, 0, 0]; % Assuming brain is centered at origin
toCenter = midpoint - brainCenter;
if norm(toCenter) > 0
    outwardDir = toCenter / norm(toCenter);
else
    % Fallback: use z-direction
    outwardDir = [0, 0, 1];
end

% Create arch points
xArch = zeros(1, numPoints);
yArch = zeros(1, numPoints);
zArch = zeros(1, numPoints);

for i = 1:numPoints
    % Linear interpolation along the direct path
    basePoint = point1 + t(i) * (point2 - point1);

    % Add parabolic offset in outward direction
    archOffset = 4 * t(i) * (1 - t(i)) * archHeight * distance * outwardDir;

    archPoint = basePoint + archOffset;
    xArch(i) = archPoint(1);
    yArch(i) = archPoint(2);
    zArch(i) = archPoint(3);
end

% Handle electrode selection
function onElectrodeClicked(app, src, ~)

    % Find which electrode was clicked
    clickedElectrodeIdx = 0;
    % First check for UserData
    if isprop(src, 'UserData') && ~isempty(src.UserData)
        clickedElectrodeIdx = src.UserData;
    else
        % Search through 3D electrodes
        for i = 1:length(app.BrainElectrodeHandles)
            if isvalid(app.BrainElectrodeHandles(i)) && src ==
app.BrainElectrodeHandles(i)
                clickedElectrodeIdx = i;
                break;
            end
        end

        % If not found in 3D, search through 2D electrodes
        if clickedElectrodeIdx == 0
            for i = 1:length(app.CircleElectrodeHandles)
                if isvalid(app.CircleElectrodeHandles(i)) && src ==
app.CircleElectrodeHandles(i)
                    clickedElectrodeIdx = i;
                    break;
                end
            end
        end
    end

    if clickedElectrodeIdx > 0
        % Toggle selection
        if app.SelectedElectrodeIdx == clickedElectrodeIdx

```

```

        app.SelectedElectrodeIdx = 0;
        app.updateInfoPanel(0);
    else
        app.SelectedElectrodeIdx = clickedElectrodeIdx;
        app.updateInfoPanel(clickedElectrodeIdx);
    end

    % Update electrode appearance for both 3D and 2D views
    app.updateElectrodeAppearance();

    % Update 3D connectivity lines efficiently with proper axes state
    hold(app.BrainAxes, 'on'); % Ensure we don't clear existing graphics

    if app.SelectedElectrodeIdx == 0
        app.drawBackgroundLines3D();
        app.drawStrongestConnection3D();
        if ~isempty(app.BrainSelectedLines)
            for i = 1:length(app.BrainSelectedLines)
                if isValid(app.BrainSelectedLines(i))
                    delete(app.BrainSelectedLines(i));
                end
            end
            app.BrainSelectedLines = gobjects(0);
        end
    else
        if isempty(app.BrainBackgroundLines)
            app.drawBackgroundLines3D();
        end
        if ~isempty(app.BrainStrongestLine) && isValid(app.BrainStrongestLine)
            delete(app.BrainStrongestLine);
            app.BrainStrongestLine = gobjects(0);
        end
        app.drawSelectedElectrodeLines3D();
    end

    hold(app.BrainAxes, 'off'); % Reset hold state
    % For 2D: complete redraw
    app.updateCircularGraph();
end
end

% Update electrode appearance without full graph redraw
function updateElectrodeAppearance(app)
    % Update 3D electrode appearance
    if ~isempty(app.BrainElectrodeHandles)
        for i = 1:length(app.BrainElectrodeHandles)
            if isValid(app.BrainElectrodeHandles(i))
                if i == app.SelectedElectrodeIdx
                    % Highlight selected electrode
                    app.BrainElectrodeHandles(i).SizeData = 300;
                    app.BrainElectrodeHandles(i).MarkerEdgeColor = 'yellow';
                    app.BrainElectrodeHandles(i).LineWidth = 3;
                    if ~isempty(app.BrainElectrodeLabels) &&
isValid(app.BrainElectrodeLabels(i))
                        app.BrainElectrodeLabels(i).Color = 'yellow';
                    end
                else
                    % Normal appearance
                    app.BrainElectrodeHandles(i).SizeData = 200;
                    app.BrainElectrodeHandles(i).MarkerEdgeColor = 'black';
                    app.BrainElectrodeHandles(i).LineWidth = 1;
                    if ~isempty(app.BrainElectrodeLabels) &&
isValid(app.BrainElectrodeLabels(i))
                        app.BrainElectrodeLabels(i).Color = 'white';
                    end
                end
            end
        end
    end

    % Update 2D electrode appearance
    if ~isempty(app.CircleElectrodeHandles)
        for i = 1:length(app.CircleElectrodeHandles)
            if isValid(app.CircleElectrodeHandles(i))
                if i == app.SelectedElectrodeIdx
                    % Highlight selected electrode

```

```

        app.CircleElectrodeHandles(i).MarkerSize = 18;
        app.CircleElectrodeHandles(i).MarkerEdgeColor = 'yellow';
        app.CircleElectrodeHandles(i).LineWidth = 3;
        if ~isempty(app.CircleElectrodeLabels) &&
isValid(app.CircleElectrodeLabels(i))
            app.CircleElectrodeLabels(i).Color = 'yellow';
        end
    else
        % Normal appearance
        app.CircleElectrodeHandles(i).MarkerSize = 14;
        app.CircleElectrodeHandles(i).MarkerEdgeColor = 'black';
        app.CircleElectrodeHandles(i).LineWidth = 1;
        if ~isempty(app.CircleElectrodeLabels) &&
isValid(app.CircleElectrodeLabels(i))
            app.CircleElectrodeLabels(i).Color = 'white';
        end
    end
end
end

% Callbacks that handle component events
methods (Access = private)
    % Code that executes after component creation
    function startupFcn(app)
        % Set default values
        app.ProcessingLabel.Text = 'No data';
        app.ConditionDropDown.Value = 'Rest';
        app.Freq1DropDown.Value = 'Beta (12-30 Hz)';
        app.Freq2DropDown.Value = 'High Gamma (70-250 Hz)';
        app.ElectrodeRole = 0;
        app.ShowElectrodesLabelsSwitch.Value = 'On';
        app.NumElectrodes = 0;
        app.ElectrodeRole = 0; % Phase vs Amplitude
        app.SelectedElectrodeIdx = 0; % No selected electrode
        app.SamplingFrequency = 2000;
        % Initialize loading animation properties
        app.IsDataLoaded = false;
        app.NeedsReprocessing = false;
        app.LoadingTimer = [];
        app.IsProcessing = false;
        % Hide right panel on startup
        % OVERRIDE the SizeChangedFcn to control layout behavior
        app.UIFigure.SizeChangedFcn = @(src,evt) app.customLayoutControl(src,evt);

        % Initialize graphics handle arrays
        app.CircleLinesHandles = gobjects(0);
        app.BrainLinesHandles = gobjects(0);
        app.BrainElectrodeLabels = gobjects(0);
        app.CircleElectrodeLabels = gobjects(0);

        % Initialize custom colormap
        app.customColormap();

        % Initialize brain model
        app.updateBrainModel();
        rotate3d(app.BrainAxes,'on'); % Enable free mouse rotating

        % Create the info panel
        app.createInfoPanel();
        app.InfoPanelVisible = false; % Show info panel only after processing
    end
    % Button pushed function: UploadMATFileButton
    function UploadMATFileButtonPushed(app, event)
        % Hide the main window
        app.UIFigure.Visible = 'off';

        % Open file dialog to select .mat file
        [file, path] = uigetfile('*.*mat', 'Select ECoG Data .mat File');

        % Restore main window
        app.UIFigure.Visible = 'on';
        drawnow;
    end
end

```

```

figure(app.UIFigure); % Gives it focus again
if isequal(file, 0) % If user cancelled
    return;
end

filePath = fullfile(path, file);
try
    % Setup loading animation
    app.setupLoadingAnimation();

    % Start the processing with background animation
    app.processDataWithAnimation(filePath, file);

catch ME
    % Stop loading animation on error
    app.stopLoadingAnimation();

    % Error indicators
    uialert(app.UIFigure, ['Error loading file: ' ME.message], 'Error');
    app.DataLoadLamp.Color = 'red';
    app.ProcessingLabel.FontColor = 'red';
    app.ProcessingLabel.Text = " Error";

    if ~isempty(app.InfoPanel)
        app.setInfoPanelVisible(false); % Hide info panel
    end
end
end

% Value changed function: ShowElectrodesLabelsSwitch
function ShowElectrodesLabelsSwitchValueChanged(app, event)
    % Control 3D labels
    if ~isempty(app.BrainElectrodeLabels)
        for i = 1:numel(app.BrainElectrodeLabels)
            if isValid(app.BrainElectrodeLabels(i))
                app.BrainElectrodeLabels(i).Visible =
app.ShowElectrodesLabelsSwitch.Value;
            end
        end
    end

    % Control 2D labels
    if ~isempty(app.CircleElectrodeLabels)
        for i = 1:numel(app.CircleElectrodeLabels)
            if isValid(app.CircleElectrodeLabels(i))
                app.CircleElectrodeLabels(i).Visible =
app.ShowElectrodesLabelsSwitch.Value;
            end
        end
    end
end

% Value changed function: ConditionDropDown
function ConditionDropDownValueChanged(app, event)
    % Only process if data is already loaded
    if isempty(app.PatientData)
        return;
    end

    % Set lamp to yellow during processing
    app.DataLoadLamp.Color = 'yellow';
    app.ProcessingLabel.FontColor = 'yellow';
    app.ProcessingLabel.Text = " Processing...";
    drawnow; % Force UI update to show yellow immediately

    try
        % Update current condition data based on drop-down selection
        app.updateCurrentConditionData();

        % Proceed to process and update visualization
        app.processAndDisplayData();

        % Set lamp back to green when done
        app.DataLoadLamp.Color = 'green';
        app.ProcessingLabel.Text = "";
    catch ME

```

```

    % Set lamp to red on error
    app.DataLoadLamp.Color = 'red';
    app.ProcessingLabel.FontColor = 'red';
    app.ProcessingLabel.Text = "    Error";
    uialert(app.UIFigure, ['Error processing condition change: ' ME.message],
'Error');
end
end
% Value changed function: Freq1DropDown
function Freq1DropDownValueChanged(app, event)
% Only process if data is already loaded
if isempty(app.PatientData)
    return;
end

% Set lamp to yellow during processing
app.DataLoadLamp.Color = 'yellow';
app.ProcessingLabel.FontColor = 'yellow';
app.ProcessingLabel.Text = "    Processing...";
drawnow;

try
    % Process and update visualization
    app.processAndDisplayData();

    % Set lamp back to green when done
    app.DataLoadLamp.Color = 'green';
    app.ProcessingLabel.Text = "";

catch ME
    app.DataLoadLamp.Color = 'red';
    app.ProcessingLabel.FontColor = 'red';
    app.ProcessingLabel.Text = "    Error";
    uialert(app.UIFigure, ['Error processing frequency change: ' ME.message],
'Error');
end
end
% Value changed function: Freq2DropDown
function Freq2DropDownValueChanged(app, event)
% Only process if data is already loaded
if isempty(app.PatientData)
    return;
end

% Set lamp to yellow during processing
app.DataLoadLamp.Color = 'yellow';
app.ProcessingLabel.FontColor = 'yellow';
app.ProcessingLabel.Text = "    Processing...";
drawnow;

try
    % Process and update visualization
    app.processAndDisplayData();

    % Set lamp back to green when done
    app.DataLoadLamp.Color = 'green';
    app.ProcessingLabel.Text = "";

catch ME
    app.DataLoadLamp.Color = 'red';
    app.ProcessingLabel.FontColor = 'red';
    app.ProcessingLabel.Text = "    Error";
    uialert(app.UIFigure, ['Error processing frequency change: ' ME.message],
'Error');
end
end
% Selection changed function: SelectedElectrodeRole
function ElectrodeRoleChanged(app, event)
selectedButton = app.SelectedElectrodeRole.SelectedObject;

if selectedButton == app.PhaseButton
    app.ElectrodeRole = 0;
else % selectedButton == app.AmpButton
    app.ElectrodeRole = 1;
end

```

```

    % Update visualization for current view mode
    if isempty(app.CurrentConditionData)
        uialert(app.UIFigure, 'No ECoG data available for processing. Please load
data first.', 'Warning', 'Icon', 'warning');
        return;
    end
    % Set lamp to yellow during processing
    app.DataLoadLamp.Color = 'yellow';
    app.ProcessingLabel.FontColor = 'yellow';
    app.ProcessingLabel.Text = " Processing...";
    drawnow;

    try
        % Update the graphs
        app.updateBrainModel();
        app.updateCircularGraph();

        % Update info panel only if an electrode is selected (global info
shouldn't change)
        if app.InfoPanelVisible && ~isempty(app.ElectrodeInfo) &&
app.SelectedElectrodeIdx > 0
            app.updateInfoPanel(app.SelectedElectrodeIdx);
        end

        % Set lamp back to green when done
        app.DataLoadLamp.Color = 'green';
        app.ProcessingLabel.Text = "";

    catch ME
        app.DataLoadLamp.Color = 'red';
        app.ProcessingLabel.FontColor = 'red';
        app.ProcessingLabel.Text = " Error";
        uialert(app.UIFigure, ['Error changing selected electrode role: ' +
ME.message], 'Error');
    end
end
% Changes arrangement of the app based on UIFigure width
function updateAppLayout(app, event)
    currentFigureWidth = app.UIFigure.Position(3);
    if(currentFigureWidth <= app.onePanelWidth)
        % Change to a 3x1 grid
        app.GridLayout.RowHeight = {587, 587, 587};
        app.GridLayout.ColumnWidth = {'1x'};
        app.CenterPanel.Layout.Row = 1;
        app.CenterPanel.Layout.Column = 1;
        app.LeftPanel.Layout.Row = 2;
        app.LeftPanel.Layout.Column = 1;
        app.RightPanel.Layout.Row = 3;
        app.RightPanel.Layout.Column = 1;
    elseif (currentFigureWidth > app.onePanelWidth && currentFigureWidth <=
app.twoPanelWidth)
        % Change to a 2x2 grid
        app.GridLayout.RowHeight = {587, 587};
        app.GridLayout.ColumnWidth = {'1x', '1x'};
        app.CenterPanel.Layout.Row = 1;
        app.CenterPanel.Layout.Column = [1,2];
        app.LeftPanel.Layout.Row = 2;
        app.LeftPanel.Layout.Column = 1;
        app.RightPanel.Layout.Row = 2;
        app.RightPanel.Layout.Column = 2;
    else
        % Change to a 1x3 grid
        app.GridLayout.RowHeight = {'1x'};
        app.GridLayout.ColumnWidth = {222, '1x', 324};
        app.LeftPanel.Layout.Row = 1;
        app.LeftPanel.Layout.Column = 1;
        app.CenterPanel.Layout.Row = 1;
        app.CenterPanel.Layout.Column = 2;
        app.RightPanel.Layout.Row = 1;
        app.RightPanel.Layout.Column = 3;
    end
end
% Component initialization

```

```

methods (Access = private)
    % Create UIFigure and components
    function createComponents(app)
        % Get the file path for locating images
        pathToMLAPP = fileparts(fullfile('fullpath'));
        % Create UIFigure and hide until all components are created
        app.UIFigure = uifigure('Visible', 'off');
        app.UIFigure.AutoScaleChildren = 'off';
        app.UIFigure.Position = [100 100 1146 587];
        app.UIFigure.Name = 'MATLAB App';
        app.UIFigure.Icon = fullfile(pathToMLAPP, 'PLView_icon.png');
        app.UIFigure.SizeChangedFcn = createCallbackFcn(app, @updateAppLayout, true);
        app.UIFigure.WindowState = 'maximized';
        % Create GridLayout
        app.GridLayout = uigridlayout(app.UIFigure);
        app.GridLayout.ColumnWidth = {222, '1x', 324};
        app.GridLayout.RowHeight = {'1x'};
        app.GridLayout.ColumnSpacing = 0;
        app.GridLayout.RowSpacing = 0;
        app.GridLayout.Padding = [0 0 0 0];
        app.GridLayout.Scrollable = 'on';
        % Create LeftPanel
        app.LeftPanel = uipanel(app.GridLayout);
        app.LeftPanel.Layout.Row = 1;
        app.LeftPanel.Layout.Column = 1;
        % Create GridLayout2
        app.GridLayout2 = uigridlayout(app.LeftPanel);
        app.GridLayout2.ColumnWidth = {'0.15x'};
        app.GridLayout2.RowHeight = {30, 100, '1x', 30, 30, 30, 30, 30, 65, 30, 30,
        '1x', 30};
        % Create UploadMATFileButton
        app.UploadMATFileButton = uibutton(app.GridLayout2, 'push');
        app.UploadMATFileButton.ButtonPushedFcn = createCallbackFcn(app,
        @UploadMATFileButtonPushed, true);
        app.UploadMATFileButton.Icon = fullfile(pathToMLAPP, 'upload.png');
        app.UploadMATFileButton.Layout.Row = 13;
        app.UploadMATFileButton.Layout.Column = 1;
        app.UploadMATFileButton.Text = 'Upload (.mat file)';
        % Create ShowElectrodesLabelsSwitch
        app.ShowElectrodesLabelsSwitch = uiswitch(app.GridLayout2, 'slider');
        app.ShowElectrodesLabelsSwitch.ValueChangedFcn = createCallbackFcn(app,
        @ShowElectrodeLabelsSwitchValueChanged, true);
        app.ShowElectrodesLabelsSwitch.Layout.Row = 11;
        app.ShowElectrodesLabelsSwitch.Layout.Column = 1;
        % Create SelectedElectrodeRole
        app.SelectedElectrodeRole = uibuttongroup(app.GridLayout2);
        app.SelectedElectrodeRole.SelectionChangedFcn = createCallbackFcn(app,
        @ElectrodeRoleChanged, true);
        app.SelectedElectrodeRole.BorderType = 'none';
        app.SelectedElectrodeRole.Title = 'Selected Electrode Role:';
        app.SelectedElectrodeRole.Layout.Row = 9;
        app.SelectedElectrodeRole.Layout.Column = 1;
        % Create PhaseButton
        app.PhaseButton = uiradiobutton(app.SelectedElectrodeRole);
        app.PhaseButton.Text = 'Phase      (Low frequency)';
        app.PhaseButton.Position = [11 19 169 22];
        app.PhaseButton.Value = true;
        % Create AmpButton
        app.AmpButton = uiradiobutton(app.SelectedElectrodeRole);
        app.AmpButton.Text = 'Amplitude (High frequency)';
        app.AmpButton.Position = [11 -3 171 22];
        % Create Freq2DropDown
        app.Freq2DropDown = uidropdown(app.GridLayout2);
        app.Freq2DropDown.Items = {'Delta (0.5-4 Hz)', 'Theta (4-8 Hz)', 'Alpha (8-12
        Hz)', 'Beta (12-30 Hz)', 'Gamma (30-70 Hz)', 'High Gamma (70-250 Hz)'};
        app.Freq2DropDown.ValueChangedFcn = createCallbackFcn(app,
        @Freq2DropDownValueChanged, true);
        app.Freq2DropDown.Layout.Row = 8;
        app.Freq2DropDown.Layout.Column = 1;
        app.Freq2DropDown.Value = 'High Gamma (70-250 Hz)';
        % Create Freq1DropDown
        app.Freq1DropDown = uidropdown(app.GridLayout2);
        app.Freq1DropDown.Items = {'Delta (0.5-4 Hz)', 'Theta (4-8 Hz)', 'Alpha (8-12
        Hz)', 'Beta (12-30 Hz)', 'Gamma (30-70 Hz)', 'High Gamma (70-250 Hz)'};
        app.Freq1DropDown.ValueChangedFcn = createCallbackFcn(app,

```

```

@Freq1DropDownValueChanged, true);
    app.Freq1DropDown.Layout.Row = 7;
    app.Freq1DropDown.Layout.Column = 1;
    app.Freq1DropDown.Value = 'Beta (12-30 Hz)';
    % Create ConditionDropDown
    app.ConditionDropDown = uidropdown(app.GridLayout2);
    app.ConditionDropDown.Items = {'Rest', 'Easy Task (CountF)', 'Hard Task
(Alt)'};
    app.ConditionDropDown.ValueChangedFcn = createCallbackFcn(app,
@ConditionDropDownValueChanged, true);
    app.ConditionDropDown.Layout.Row = 5;
    app.ConditionDropDown.Layout.Column = 1;
    app.ConditionDropDown.Value = 'Rest';
    % Create Logo
    app.Logo = uiimage(app.GridLayout2);
    app.Logo.Layout.Row = 2;
    app.Logo.Layout.Column = 1;
    app.Logo.ImageSource = fullfile(pathToMLAPP, 'PLView_logo.png');
    % Create ElectrodesLabelsSwitchLabel
    app.ElectrodesLabelsSwitchLabel = uilabel(app.GridLayout2);
    app.ElectrodesLabelsSwitchLabel.VerticalAlignment = 'bottom';
    app.ElectrodesLabelsSwitchLabel.Layout.Row = 10;
    app.ElectrodesLabelsSwitchLabel.Layout.Column = 1;
    app.ElectrodesLabelsSwitchLabel.Text = 'Electrodes Labels:';
    % Create SelectedElectrodeFrequencyLabel
    app.SelectedElectrodeFrequencyLabel = uilabel(app.GridLayout2);
    app.SelectedElectrodeFrequencyLabel.VerticalAlignment = 'bottom';
    app.SelectedElectrodeFrequencyLabel.Layout.Row = 6;
    app.SelectedElectrodeFrequencyLabel.Layout.Column = 1;
    app.SelectedElectrodeFrequencyLabel.Text = 'Phase and Amp Frequencies:';
    % Create TestConditionDropDownLabel
    app.TestConditionDropDownLabel = uilabel(app.GridLayout2);
    app.TestConditionDropDownLabel.VerticalAlignment = 'bottom';
    app.TestConditionDropDownLabel.Layout.Row = 4;
    app.TestConditionDropDownLabel.Layout.Column = 1;
    app.TestConditionDropDownLabel.Text = 'Test Condition:';
    % Create CenterPanel
    app.CenterPanel = uipanel(app.GridLayout);
    app.CenterPanel.BackgroundColor = [0 0 0];
    app.CenterPanel.Layout.Row = 1;
    app.CenterPanel.Layout.Column = 2;
    % Create GridLayout3
    app.GridLayout3 = uigridlayout(app.CenterPanel);
    app.GridLayout3.ColumnWidth = {15, '1x', 15};
    app.GridLayout3.RowHeight = {15, '3x', 15};
    app.GridLayout3.ColumnSpacing = 0;
    app.GridLayout3.RowSpacing = 0;
    app.GridLayout3.BackgroundColor = [0 0 0];
    % Create BrainAxes
    app.BrainAxes = uiaxes(app.GridLayout3);
    title(app.BrainAxes, 'Title')
    xlabel(app.BrainAxes, 'X')
    ylabel(app.BrainAxes, 'Y')
    zlabel(app.BrainAxes, 'Z')
    app.BrainAxes.GridAlpha = 0;
    app.BrainAxes.Layout.Row = 2;
    app.BrainAxes.Layout.Column = 2;
    % Create DataLoadLamp
    app.DataLoadLamp = uilamp(app.GridLayout3);
    app.DataLoadLamp.Layout.Row = 3;
    app.DataLoadLamp.Layout.Column = 1;
    app.DataLoadLamp.Color = [0.902 0.902 0.902];
    % Create ProcessingLabel
    app.ProcessingLabel = uilabel(app.GridLayout3);
    app.ProcessingLabel.FontColor = [1 1 1];
    app.ProcessingLabel.Layout.Row = 3;
    app.ProcessingLabel.Layout.Column = 2;
    % Create RightPanel
    app.RightPanel = uipanel(app.GridLayout);
    app.RightPanel.Layout.Row = 1;
    app.RightPanel.Layout.Column = 3;
    % Create GridLayout4
    app.GridLayout4 = uigridlayout(app.RightPanel);
    app.GridLayout4.ColumnWidth = {'0.5x', 15};
    app.GridLayout4.RowHeight = {'2x', '1x'};

```

```

    app.GridLayout4.ColumnSpacing = 0;
    app.GridLayout4.RowSpacing = 0;
    app.GridLayout4.HandleVisibility = 'off';
    app.GridLayout4.BackgroundColor = [0.251 0.251 0.251];
    % Create CircleAxes
    app.CircleAxes = uiaxes(app.GridLayout4);
    title(app.CircleAxes, 'Title')
    xlabel(app.CircleAxes, 'X')
    ylabel(app.CircleAxes, 'Y')
    zlabel(app.CircleAxes, 'Z')
    app.CircleAxes.Toolbar.Visible = 'off';
    app.CircleAxes.Layout.Row = 1;
    app.CircleAxes.Layout.Column = 1;
    % Show the figure after all components are created
    app.UIFigure.Visible = 'on';
end
% App creation and deletion
methods (Access = public)
    % Construct app
    function app = PLView
        % Create UIFigure and components
        createComponents(app)
        % Register the app with App Designer
        registerApp(app, app.UIFigure)
        % Execute the startup function
        runStartupFcn(app, @startupFcn)
        if nargout == 0
            clear app
        end
    end
    % Code that executes before app deletion
    function delete(app)
        % Delete UIFigure when app is deleted
        delete(app.UIFigure)
    end
end
end

```