ddumtwvqh

November 29, 2024

1 PROYECTO FINAL

Carolina Daniells Zaldívar

1.0.1 Definir grafos con la matriz estableciendo umbrales de coactivacion de 0.8, 0.9 y 1 y graficar cada grafo. Añadir las coordenadas tridimencionales (incluidas en el archivo mat)

[34]:
```
import scipy.io as sp
import numpy as np
import pandas as pd
import networkx as nx
import matplotlib.pyplot as plt
import seaborn as sns

data = sp.loadmat(r"C:
 ↪\Users\dnlls\OneDrive\Escritorio\PUMA\Neurociencias\Escritorio␣
 ↪Neurociencias\5o semestre\Modelos\PROYECTO FINAL MODELOS␣
 ↪dump\FinalModelosCode\Coactivation_matrix.mat")
coactivation_matrix = data['Coactivation_matrix']
coord = data['Coord']
x, y, z = coord[:, 0], coord[:, 1], coord[:, 2]
m = 0

umbrales = [0.2, 0.8, 0.9, 1]

for u in umbrales:

    filtered_matrix= np.where(coactivation_matrix >= u, 1, 0)
    G = nx.from_numpy_array(filtered_matrix)
    fig = plt.figure(figsize=(10, 10))
    ax = fig.add_subplot(projection='3d')
    ax.scatter(x, y, z, c='blue', s=50, marker='.')
    print(f'Para el grafo {u}, existen las sig coords de conexiones:\n')
    for edge in G.edges():
        node1, node2 = edge
        x_coords = [x[node1], x[node2]]
        y_coords = [y[node1], y[node2]]
```

```
        z_coords = [z[node1], z[node2]]
        ax.plot(x_coords, y_coords, z_coords, c='red', alpha=0.5)
        m+= 1
        print('Conexión: ','x: ', x_coords, 'y: ',y_coords, 'z',z_coords)

    ax.set_title(f'Grafo con un umbral de {u}', fontsize= 30)
    ax.set_xlabel('X')
    ax.set_ylabel('Y')
    ax.set_zlabel('Z')

plt.show()
```

Para el grafo 0.2, existen las siguientes coordenadas de conexiones:

```
Conexión:  x:  [np.float64(7.55725190839695), np.float64(9.115879828326186)] y:
[np.float64(33.8320610687023), np.float64(22.454935622317606)] z
[np.float64(23.511450381679396), np.float64(24.206008583690988)]
Conexión:  x:  [np.float64(-5.214035087719296), np.float64(-5.017094017094024)]
y:  [np.float64(-12.526315789473685), np.float64(-5.7521367521367495)] z
[np.float64(42.11929824561403), np.float64(37.66666666666667)]
Conexión:  x:  [np.float64(-5.198237885462561), np.float64(-5.677419354838705)]
y:  [np.float64(17.823788546255514), np.float64(26.66129032258064)] z
[np.float64(36.889867841409696), np.float64(40.314516129032256)]
Conexión:  x:  [np.float64(-8.016597510373444), np.float64(-6.0297029702970235)]
y:  [np.float64(-38.016597510373444), np.float64(-38.00990099009901)] z
[np.float64(48.73858921161826), np.float64(40.95049504950495)]
Conexión:  x:  [np.float64(-6.114942528735625), np.float64(-4.257142857142853)]
y:  [np.float64(-34.67432950191571), np.float64(-42.409523809523805)] z
[np.float64(34.643678160919535), np.float64(28.799999999999997)]
Conexión:  x:  [np.float64(8.1869918699187), np.float64(8.421940928270047)] y:
[np.float64(19.86178861788619), np.float64(26.405063291139243)] z
[np.float64(33.11382113821138), np.float64(36.42194092827005)]
Conexión:  x:  [np.float64(-8.206489675516224), np.float64(-6.972111553784856)]
y:  [np.float64(-68.33038348082596), np.float64(-63.31474103585657)] z
[np.float64(26.359882005899706), np.float64(27.856573705179287)]
Conexión:  x:  [np.float64(-8.21052631578948), np.float64(-16.328358208955223)]
y:  [np.float64(-90.77732793522267), np.float64(-83.3507462686567)] z
[np.float64(29.141700404858298), np.float64(31.13432835820896)]
Conexión:  x:  [np.float64(16.4320987654321), np.float64(23.755319148936167)] y:
[np.float64(-95.48148148148148), np.float64(-90.56382978723404)] z
[np.float64(10.654320987654316), np.float64(12.031914893617028)]
Conexión:  x:  [np.float64(-51.27088607594936), np.float64(-48.376811594202906)]
y:  [np.float64(12.12658227848101), np.float64(17.775362318840592)] z
[np.float64(24.99746835443038), np.float64(26.76086956521739)]
Conexión:  x:  [np.float64(-51.27088607594936), np.float64(-48.1930501930502)]
y:  [np.float64(12.12658227848101), np.float64(7.0888030888030755)] z
[np.float64(24.99746835443038), np.float64(32.308880308880305)]
```

Conexión: x: [np.float64(51.43726235741445), np.float64(54.24545454545454)] y: [np.float64(14.646387832699617), np.float64(22.681818181818187)] z [np.float64(4.365019011406844), np.float64(3.654545454545456)]
Conexión: x: [np.float64(45.06837606837607), np.float64(44.18410041841004)] y: [np.float64(11.418803418803407), np.float64(16.878661087866107)] z [np.float64(32.5042735042735), np.float64(31.43096234309624)]
Conexión: x: [np.float64(45.06837606837607), np.float64(50.06896551724138)] y: [np.float64(11.418803418803407), np.float64(7.400000000000006)] z [np.float64(32.5042735042735), np.float64(40.28965517241379)]
Conexión: x: [np.float64(52.421487603305785), np.float64(55.407766990291265)] y: [np.float64(13.487603305785115), np.float64(14.0)] z [np.float64(17.305785123966942), np.float64(11.5631067961165)]
Conexión: x: [np.float64(-35.256281407035175), np.float64(-31.714285714285708)] y: [np.float64(37.6281407035176), np.float64(28.73260073260073)] z [np.float64(-12.974874371859293), np.float64(-16.468864468864467)]
Conexión: x: [np.float64(-23.509090909090915), np.float64(-20.66666666666667)] y: [np.float64(-1.9454545454545524), np.float64(-11.02057613168725)] z [np.float64(-18.454545454545453), np.float64(-18.995884773662553)]
Conexión: x: [np.float64(-44.743494423791816), np.float64(-45.769230769230774)] y: [np.float64(32.37918215613382), np.float64(27.01538461538462)] z [np.float64(18.84014869888476), np.float64(24.830769230769235)]
Conexión: x: [np.float64(-44.743494423791816), np.float64(-42.03076923076924)] y: [np.float64(32.37918215613382), np.float64(33.9897435897436)] z [np.float64(18.84014869888476), np.float64(26.266666666666666)]
Conexión: x: [np.float64(-49.66666666666666), np.float64(-45.769230769230774)] y: [np.float64(21.724999999999994), np.float64(27.01538461538462)] z [np.float64(17.950000000000003), np.float64(24.830769230769235)]
Conexión: x: [np.float64(-40.7032967032967), np.float64(-43.36603773584906)] y: [np.float64(40.036630036630044), np.float64(39.418867924528314)] z [np.float64(10.065934065934073), np.float64(1.0339622641509436)]
Conexión: x: [np.float64(47.55905511811024), np.float64(43.905138339920946)] y: [np.float64(31.38582677165354), np.float64(37.49407114624506)] z [np.float64(22.464566929133852), np.float64(19.501976284584984)]
Conexión: x: [np.float64(-5.710743801652896), np.float64(-6.273127753303967)] y: [np.float64(39.3388429752066), np.float64(34.70484581497797)] z [np.float64(-10.454545454545453), np.float64(-5.092511013215855)]
Conexión: x: [np.float64(-4.067510548523202), np.float64(6.428070175438592)] y: [np.float64(54.751054852320664), np.float64(50.7719298245614)] z [np.float64(-10.270042194092824), np.float64(-8.308771929824559)]
Conexión: x: [np.float64(-39.78867924528302), np.float64(-44.77902621722848)] y: [np.float64(15.93962264150943), np.float64(17.303370786516865)] z [np.float64(42.93584905660377), np.float64(37.14606741573034)]
Conexión: x: [np.float64(-25.42168674698796), np.float64(-31.78867924528302)] y: [np.float64(15.799196787148588), np.float64(17.260377358490558)] z [np.float64(54.85140562248995), np.float64(51.13962264150943)]
Conexión: x: [np.float64(-25.42168674698796), np.float64(-18.87285223367698)]

y: [np.float64(15.799196787148588), np.float64(17.573883161512015)] z
[np.float64(54.85140562248995), np.float64(57.71821305841925)]
Conexión: x: [np.float64(-24.02380952380952), np.float64(-20.75)] y:
[np.float64(47.428571428571416), np.float64(55.4375)] z
[np.float64(28.626984126984127), np.float64(24.546875)]
Conexión: x: [np.float64(40.45454545454545), np.float64(38.65853658536585)] y:
[np.float64(50.09090909090909), np.float64(55.20325203252034)] z
[np.float64(-11.780303030303031), np.float64(-6.040650406504071)]
Conexión: x: [np.float64(29.935275080906152), np.float64(23.755932203389833)]
y: [np.float64(28.03236245954693), np.float64(25.349152542372877)] z
[np.float64(43.94822006472492), np.float64(51.01694915254237)]
Conexión: x: [np.float64(44.73529411764706), np.float64(44.63598326359833)] y:
[np.float64(45.25490196078431), np.float64(48.184100418410054)] z
[np.float64(11.607843137254903), np.float64(4.075313807531387)]
Conexión: x: [np.float64(41.559471365638764), np.float64(43.905138339920946)]
y: [np.float64(40.255506607929505), np.float64(37.49407114624506)] z
[np.float64(25.066079295154182), np.float64(19.501976284584984)]
Conexión: x: [np.float64(27.484126984126988), np.float64(35.62595419847328)]
y: [np.float64(46.293650793650784), np.float64(47.969465648854964)] z
[np.float64(29.626984126984127), np.float64(24.961832061068705)]
Conexión: x: [np.float64(38.969162995594715), np.float64(38.86842105263158)]
y: [np.float64(0.4581497797356917), np.float64(-1.4868421052631646)] z
[np.float64(51.048458149779734), np.float64(58.26315789473685)]
Conexión: x: [np.float64(-24.69291338582677), np.float64(-30.414937759336098)]
y: [np.float64(-6.2755905511810965), np.float64(-9.684647302904565)] z
[np.float64(64.43307086614172), np.float64(60.63070539419087)]
Conexión: x: [np.float64(-23.109489051094897),
np.float64(-21.028571428571425)] y: [np.float64(60.277372262773724),
np.float64(61.428571428571416)] z [np.float64(2.9562043795620383),
np.float64(-6.383673469387759)]
Conexión: x: [np.float64(-19.83333333333333), np.float64(-16.784482758620683)]
y: [np.float64(61.93333333333334), np.float64(60.90517241379311)] z
[np.float64(10.583333333333329), np.float64(18.534482758620683)]
Conexión: x: [np.float64(27.05645161290323), np.float64(26.611111111111114)]
y: [np.float64(-0.5725806451612954), np.float64(-11.992063492063494)] z
[np.float64(-18.806451612903224), np.float64(-18.96825396825397)]
Conexión: x: [np.float64(-5.697560975609761), np.float64(-5.00946372239747)]
y: [np.float64(61.053658536585374), np.float64(50.466876971608826)] z
[np.float64(18.643902439024387), np.float64(19.671924290220815)]
Conexión: x: [np.float64(-5.697560975609761), np.float64(-6.871559633027516)]
y: [np.float64(61.053658536585374), np.float64(60.871559633027516)] z
[np.float64(18.643902439024387), np.float64(27.449541284403665)]
Conexión: x: [np.float64(-8.62135922330097), np.float64(-5.096989966555185)]
y: [np.float64(64.95145631067962), np.float64(57.39130434782609)] z
[np.float64(5.313915857605181), np.float64(7.404682274247492)]
Conexión: x: [np.float64(7.17647058823529), np.float64(10.440366972477065)] y:
[np.float64(48.14117647058825), np.float64(53.83486238532109)] z
[np.float64(38.627450980392155), np.float64(36.49541284403669)]

Conexión: x: [np.float64(23.098814229249015), np.float64(22.66666666666667)]
y: [np.float64(62.02371541501975), np.float64(62.300366300366306)] z
[np.float64(-5.003952569169954), np.float64(2.7692307692307736)]
Conexión: x: [np.float64(-29.63404255319149), np.float64(-26.2510460251046)]
y: [np.float64(-27.710638297872336), np.float64(-28.853556485355654)] z
[np.float64(-25.778723404255317), np.float64(-17.89121338912134)]
Conexión: x: [np.float64(-37.445692883895134), np.float64(-46.71090047393366)]
y: [np.float64(-67.76779026217228), np.float64(-67.260663507109)] z
[np.float64(-14.382022471910112), np.float64(-13.203791469194314)]
Conexión: x: [np.float64(32.65470852017937), np.float64(31.871428571428574)]
y: [np.float64(-9.659192825112115), np.float64(0.5214285714285722)] z
[np.float64(-38.28699551569507), np.float64(-39.464285714285715)]
Conexión: x: [np.float64(39.424778761061944), np.float64(29.706122448979592)]
y: [np.float64(-21.26548672566372), np.float64(-19.004081632653055)] z
[np.float64(-26.415929203539825), np.float64(-25.461224489795917)]
Conexión: x: [np.float64(40.620370370370374), np.float64(44.294117647058826)]
y: [np.float64(-62.01851851851852), np.float64(-72.38970588235294)] z
[np.float64(-17.601851851851855), np.float64(-12.455882352941174)]
Conexión: x: [np.float64(-20.66666666666667), np.float64(-30.347169811320754)]
y: [np.float64(-11.02057613168725), np.float64(-14.981132075471692)] z
[np.float64(-18.995884773662553), np.float64(-16.596226415094343)]
Conexión: x: [np.float64(-20.66666666666667), np.float64(26.611111111111114)]
y: [np.float64(-11.02057613168725), np.float64(-11.992063492063494)] z
[np.float64(-18.995884773662553), np.float64(-18.96825396825397)]
Conexión: x: [np.float64(-38.0952380952381), np.float64(-42.803278688524586)]
y: [np.float64(-63.94444444444444), np.float64(-62.90983606557377)] z
[np.float64(42.86507936507937), np.float64(38.803278688524586)]
Conexión: x: [np.float64(-47.54751131221718), np.float64(-44.75471698113208)]
y: [np.float64(-61.809954751131215), np.float64(-66.32075471698113)] z
[np.float64(25.366515837104075), np.float64(31.632075471698116)]
Conexión: x: [np.float64(-42.803278688524586), np.float64(-44.75471698113208)]
y: [np.float64(-62.90983606557377), np.float64(-66.32075471698113)] z
[np.float64(38.803278688524586), np.float64(31.632075471698116)]
Conexión: x: [np.float64(33.298507462686565), np.float64(26.611111111111114)]
y: [np.float64(-13.082089552238813), np.float64(-11.992063492063494)] z
[np.float64(-16.253731343283583), np.float64(-18.96825396825397)]
Conexión: x: [np.float64(26.611111111111114), np.float64(20.10434782608695)]
y: [np.float64(-11.992063492063494), np.float64(-2.486956521739131)] z
[np.float64(-18.96825396825397), np.float64(-23.930434782608693)]
Conexión: x: [np.float64(-31.52380952380952), np.float64(-31.948717948717942)]
y: [np.float64(23.3113553113553), np.float64(14.88644688644689)] z
[np.float64(1.3846153846153868), np.float64(9.318681318681314)]
Conexión: x: [np.float64(-31.52380952380952), np.float64(37.201581027667984)]
y: [np.float64(23.3113553113553), np.float64(23.87351778656125)] z
[np.float64(1.3846153846153868), np.float64(0.19762845849803057)]
Conexión: x: [np.float64(37.201581027667984), np.float64(40.333333333333336)]
y: [np.float64(23.87351778656125), np.float64(13.207407407407402)] z
[np.float64(0.19762845849803057), np.float64(0.2814814814814781)]

Conexión: x: [np.float64(-15.822784810126578), np.float64(-20.30901287553648)]
y: [np.float64(-88.06751054852322), np.float64(-90.29184549356223)] z
[np.float64(-14.962025316455694), np.float64(-13.39055793991416)]
Conexión: x: [np.float64(-20.30901287553648), np.float64(-23.0204081632653)]
y: [np.float64(-90.29184549356223), np.float64(-93.23265306122448)] z
[np.float64(-13.39055793991416), np.float64(-8.78367346938775)]
Conexión: x: [np.float64(17.61338289962825), np.float64(10.36363636363636)] y:
[np.float64(-49.29368029739777), np.float64(-46.96256684491979)] z
[np.float64(-0.3866171003717511), np.float64(-1.1871657754010698)]
Conexión: x: [np.float64(14.772058823529406), np.float64(16.34042553191489)]
y: [np.float64(-81.13970588235294), np.float64(-79.55623100303951)] z
[np.float64(-3.272058823529406), np.float64(3.920972644376903)]
Conexión: x: [np.float64(-46.71090047393366), np.float64(-54.32075471698113)]
y: [np.float64(-67.260663507109), np.float64(-59.57232704402516)] z
[np.float64(-13.203791469194314), np.float64(-10.333333333333336)]
Conexión: x: [np.float64(-23.0204081632653), np.float64(-21.496855345911953)]
y: [np.float64(-93.23265306122448), np.float64(-93.9496855345912)] z
[np.float64(-8.78367346938775), np.float64(-2.289308176100633)]
Conexión: x: [np.float64(-42.68141592920355), np.float64(-35.47490347490347)]
y: [np.float64(-73.84070796460176), np.float64(-81.66795366795367)] z
[np.float64(-6.1061946902654824), np.float64(-8.903474903474901)]
Conexión: x: [np.float64(44.294117647058826), np.float64(41.24528301886792)]
y: [np.float64(-72.38970588235294), np.float64(-81.10188679245283)] z
[np.float64(-12.455882352941174), np.float64(-5.992452830188682)]
Conexión: x: [np.float64(44.294117647058826), np.float64(50.76086956521739)]
y: [np.float64(-72.38970588235294), np.float64(-63.91847826086956)] z
[np.float64(-12.455882352941174), np.float64(-13.570652173913047)]
Conexión: x: [np.float64(29.00456621004566), np.float64(34.86695278969957)] y:
[np.float64(-94.76712328767124), np.float64(-87.2961373390558)] z
[np.float64(-6.045662100456624), np.float64(-11.133047210300433)]
Conexión: x: [np.float64(-33.76699029126213), np.float64(-33.08943089430895)]
y: [np.float64(-86.8252427184466), np.float64(-85.9349593495935)] z
[np.float64(6.728155339805824), np.float64(-1.203252032520325)]
Conexión: x: [np.float64(-43.25552050473186), np.float64(-48.180257510729604)]
y: [np.float64(-73.88643533123027), np.float64(-65.76824034334764)] z
[np.float64(1.0977917981072522), np.float64(4.472103004291839)]
Conexión: x: [np.float64(-27.322683706070293), np.float64(-32.23178807947019)]
y: [np.float64(-78.23003194888179), np.float64(-73.28476821192052)] z
[np.float64(31.463258785942486), np.float64(34.90066225165563)]
Conexión: x: [np.float64(-27.322683706070293), np.float64(-22.41106719367589)]
y: [np.float64(-78.23003194888179), np.float64(-84.95652173913044)] z
[np.float64(31.463258785942486), np.float64(28.743083003952563)]
Conexión: x: [np.float64(38.536), np.float64(40.752066115702476)] y:
[np.float64(-80.904), np.float64(-79.79338842975207)] z
[np.float64(19.647999999999996), np.float64(11.95867768595042)]
Conexión: x: [np.float64(30.230483271375462), np.float64(34.299065420560744)]
y: [np.float64(-91.25650557620818), np.float64(-90.97196261682242)] z
[np.float64(10.342007434944236), np.float64(2.485981308411212)]

Conexión: x: [np.float64(30.230483271375462), np.float64(23.755319148936167)]
y: [np.float64(-91.25650557620818), np.float64(-90.56382978723404)] z
[np.float64(10.342007434944236), np.float64(12.031914893617028)]
Conexión: x: [np.float64(43.25991189427313), np.float64(47.43873517786561)] y:
[np.float64(-77.40088105726872), np.float64(-69.77075098814228)] z
[np.float64(3.709251101321584), np.float64(5.011857707509876)]
Conexión: x: [np.float64(31.592727272727274), np.float64(25.77477477477477)]
y: [np.float64(-82.64), np.float64(-82.8018018018018)] z
[np.float64(25.10545454545455), np.float64(25.810810810810807)]
Conexión: x: [np.float64(-16.328358208955223), np.float64(-22.41106719367589)]
y: [np.float64(-83.3507462686567), np.float64(-84.95652173913044)] z
[np.float64(31.13432835820896), np.float64(28.743083003952563)]
Conexión: x: [np.float64(42.96234309623431), np.float64(48.055335968379445)]
y: [np.float64(-67.42259414225941), np.float64(-63.22529644268775)] z
[np.float64(37.10460251046025), np.float64(32.3794466403162)]
Conexión: x: [np.float64(34.68376068376068), np.float64(38.71774193548387)] y:
[np.float64(-67.84615384615384), np.float64(-62.87903225806451)] z
[np.float64(44.92307692307692), np.float64(46.99193548387096)]
Conexión: x: [np.float64(34.68376068376068), np.float64(28.164556962025316)]
y: [np.float64(-67.84615384615384), np.float64(-74.22151898734177)] z
[np.float64(44.92307692307692), np.float64(42.46835443037975)]
Conexión: x: [np.float64(50.92134831460674), np.float64(53.93869731800766)] y:
[np.float64(-57.54307116104869), np.float64(-51.831417624521066)] z
[np.float64(26.32958801498127), np.float64(31.47892720306514)]
Conexión: x: [np.float64(18.692307692307693), np.float64(23.73553719008264)]
y: [np.float64(-84.91346153846155), np.float64(-82.31404958677686)] z
[np.float64(28.557692307692307), np.float64(35.719008264462815)]
Conexión: x: [np.float64(-18.061433447098977), np.float64(20.921428571428578)]
y: [np.float64(-1.3583617747440258), np.float64(-1.057142857142864)] z
[np.float64(-1.0443686006825885), np.float64(-1.085714285714289)]
Conexión: x: [np.float64(20.921428571428578), np.float64(25.544117647058826)]
y: [np.float64(-1.057142857142864), np.float64(5.330882352941188)] z
[np.float64(-1.085714285714289), np.float64(2.286764705882348)]
Conexión: x: [np.float64(-10.836879432624116), np.float64(-7.314079422382676)]
y: [np.float64(-19.808510638297875), np.float64(-20.115523465703973)] z
[np.float64(75.64539007092199), np.float64(62.88086642599279)]
Conexión: x: [np.float64(-10.836879432624116),
np.float64(-5.5564202334630295)] y: [np.float64(-19.808510638297875),
np.float64(-27.53307392996109)] z [np.float64(75.64539007092199),
np.float64(64.77042801556419)]
Conexión: x: [np.float64(-7.314079422382676), np.float64(-5.5564202334630295)]
y: [np.float64(-20.115523465703973), np.float64(-27.53307392996109)] z
[np.float64(62.88086642599279), np.float64(64.77042801556419)]
Conexión: x: [np.float64(-48.954198473282446),
np.float64(-41.091445427728615)] y: [np.float64(-37.83969465648855),
np.float64(-44.525073746312685)] z [np.float64(46.1145038167939),
np.float64(52.14749262536873)]
Conexión: x: [np.float64(-48.954198473282446), np.float64(-49.5655172413793)]

y: [np.float64(-37.83969465648855), np.float64(-40.599999999999994)] z
[np.float64(46.1145038167939), np.float64(39.62758620689655)]
Conexión: x: [np.float64(-37.10914454277287), np.float64(-30.687074829931973)]
y: [np.float64(-55.687315634218294), np.float64(-61.74829931972789)] z
[np.float64(52.607669616519175), np.float64(53.25170068027211)]
Conexión: x: [np.float64(-50.77564102564102), np.float64(-55.671875)] y:
[np.float64(-27.647435897435898), np.float64(-25.625)] z
[np.float64(41.967948717948715), np.float64(34.265625)]
Conexión: x: [np.float64(-41.091445427728615),
np.float64(-44.539923954372625)] y: [np.float64(-44.525073746312685),
np.float64(-51.28517110266159)] z [np.float64(52.14749262536873),
np.float64(46.70722433460077)]
Conexión: x: [np.float64(-44.539923954372625), np.float64(-45.72575250836121)]
y: [np.float64(-51.28517110266159), np.float64(-50.2675585284281)] z
[np.float64(46.70722433460077), np.float64(39.53177257525084)]
Conexión: x: [np.float64(-4.228915662650607), np.float64(-2.046948356807505)]
y: [np.float64(-88.43373493975903), np.float64(-86.7793427230047)] z
[np.float64(-8.289156626506028), np.float64(-0.10328638497652776)]
Conexión: x: [np.float64(-11.049180327868854),
np.float64(-10.733668341708537)] y: [np.float64(-59.531615925058546),
np.float64(-51.1608040201005)] z [np.float64(8.758782201405154),
np.float64(11.32663316582915)]
Conexión: x: [np.float64(-5.340206185567013), np.float64(-7.822222222222223)]
y: [np.float64(-97.74226804123711), np.float64(-100.41111111111111)] z
[np.float64(-7.948453608247419), np.float64(-1.37777777777778)]
Conexión: x: [np.float64(1.05263157894737), np.float64(-11.8429003021148)] y:
[np.float64(-86.70676691729324), np.float64(-77.31117824773415)] z
[np.float64(9.488721804511272), np.float64(11.299093655589118)]
Conexión: x: [np.float64(49.51931330472103), np.float64(45.80434782608695)] y:
[np.float64(-41.01287553648069), np.float64(-39.413043478260875)] z
[np.float64(47.45922746781116), np.float64(52.89130434782609)]
Conexión: x: [np.float64(49.51931330472103), np.float64(51.723636363636366)]
y: [np.float64(-41.01287553648069), np.float64(-32.54545454545455)] z
[np.float64(47.45922746781116), np.float64(43.272727272727266)]
Conexión: x: [np.float64(47.49612403100775), np.float64(50.95967741935484)] y:
[np.float64(-51.480620155038764), np.float64(-51.2258064516129)] z
[np.float64(46.52713178294573), np.float64(40.55645161290323)]
Conexión: x: [np.float64(-24.7870036101083), np.float64(-30.687074829931973)]
y: [np.float64(-64.05776173285199), np.float64(-61.74829931972789)] z
[np.float64(56.77256317689532), np.float64(53.25170068027211)]
Conexión: x: [np.float64(-19.24242424242425), np.float64(-10.320610687022906)]
y: [np.float64(-59.4469696969697), np.float64(-62.09923664122137)] z
[np.float64(61.7651515151515), np.float64(62.1145038167939)]
Conexión: x: [np.float64(-16.52233676975945), np.float64(-21.245283018867923)]
y: [np.float64(-72.24742268041237), np.float64(-72.15849056603773)] z
[np.float64(51.02405498281787), np.float64(45.41132075471698)]
Conexión: x: [np.float64(-16.52233676975945), np.float64(-10.15789473684211)]
y: [np.float64(-72.24742268041237), np.float64(-69.91228070175438)] z

[np.float64(51.02405498281787), np.float64(53.85087719298245)]
Conexión: x: [np.float64(18.546218487394952), np.float64(18.66129032258064)]
y: [np.float64(-73.25210084033614), np.float64(-69.95967741935485)] z
[np.float64(50.50420168067227), np.float64(58.04032258064515)]
Conexión: x: [np.float64(18.546218487394952), np.float64(10.355769230769226)]
y: [np.float64(-73.25210084033614), np.float64(-70.86538461538461)] z
[np.float64(50.50420168067227), np.float64(54.74038461538461)]
Conexión: x: [np.float64(23.81967213114754), np.float64(15.43529411764706)] y:
[np.float64(-58.040983606557376), np.float64(-59.521568627450975)] z
[np.float64(67.172131147541), np.float64(68.03137254901961)]
Conexión: x: [np.float64(34.20634920634921), np.float64(30.37007874015748)] y:
[np.float64(-54.388888888888886), np.float64(-62.84251968503936)] z
[np.float64(61.992063492063494), np.float64(59.84251968503938)]
Conexión: x: [np.float64(-41.964601769911496), np.float64(-42.53125)] y:
[np.float64(-28.345132743362825), np.float64(-21.8046875)] z
[np.float64(57.38053097345133), np.float64(52.328125)]
Conexión: x: [np.float64(-41.964601769911496), np.float64(-35.41634241245136)]
y: [np.float64(-28.345132743362825), np.float64(-19.64980544747081)] z
[np.float64(57.38053097345133), np.float64(59.29961089494162)]
Conexión: x: [np.float64(-50.57936507936509), np.float64(53.070422535211264)]
y: [np.float64(-9.682539682539684), np.float64(-8.302816901408448)] z
[np.float64(36.89682539682539), np.float64(33.59154929577464)]
Conexión: x: [np.float64(-60.677248677248684), np.float64(-58.40466926070039)]
y: [np.float64(-17.386243386243393), np.float64(-26.210116731517516)] z
[np.float64(20.539682539682545), np.float64(22.264591439688715)]
Conexión: x: [np.float64(-29.89387755102041), np.float64(-31.457364341085267)]
y: [np.float64(-31.35510204081632), np.float64(-25.054263565891475)] z
[np.float64(65.07755102040815), np.float64(63.72868217054264)]
Conexión: x: [np.float64(56.39543726235741), np.float64(54.95940959409594)] y:
[np.float64(-17.368821292775664), np.float64(-15.070110701107012)] z
[np.float64(38.80608365019012), np.float64(33.040590405904055)]
Conexión: x: [np.float64(30.06374501992032), np.float64(36.78260869565217)] y:
[np.float64(-33.60956175298804), np.float64(-36.35573122529644)] z
[np.float64(66.55776892430279), np.float64(62.22924901185772)]
Conexión: x: [np.float64(45.867867867867865), np.float64(51.763688760806915)]
y: [np.float64(-27.207207207207205), np.float64(-21.0835734870317)] z
[np.float64(53.567567567567565), np.float64(48.39193083573487)]
Conexión: x: [np.float64(14.01129943502825), np.float64(8.51136363636364)] y:
[np.float64(-45.89830508474576), np.float64(-48.022727272727266)] z
[np.float64(71.954802259887), np.float64(68.5)]
Conexión: x: [np.float64(53.070422535211264), np.float64(51.239852398523986)]
y: [np.float64(-8.302816901408448), np.float64(-7.4538745387453815)] z
[np.float64(33.59154929577464), np.float64(41.055350553505534)]
Conexión: x: [np.float64(-42.82527881040892), np.float64(-41.230158730158735)]
y: [np.float64(6.118959107806688), np.float64(0.1428571428571388)] z
[np.float64(42.40148698884758), np.float64(48.58730158730158)]
Conexión: x: [np.float64(-21.0390625), np.float64(-24.424489795918362)] y:
[np.float64(-14.4375), np.float64(-21.61632653061224)] z [np.float64(71.53125),

np.float64(69.68979591836734)]
Conexión: x: [np.float64(-41.59055118110237), np.float64(-41.230158730158735)]
y: [np.float64(-7.425196850393704), np.float64(0.1428571428571388)] z
[np.float64(48.01574803149606), np.float64(48.58730158730158)]
Conexión: x: [np.float64(-35.41634241245136), np.float64(-36.863247863247864)]
y: [np.float64(-19.64980544747081), np.float64(-13.452991452991455)] z
[np.float64(59.29961089494162), np.float64(55.4957264957265)]
Conexión: x: [np.float64(-35.41634241245136), np.float64(-31.457364341085267)]
y: [np.float64(-19.64980544747081), np.float64(-25.054263565891475)] z
[np.float64(59.29961089494162), np.float64(63.72868217054264)]
Conexión: x: [np.float64(36.86466165413534), np.float64(40.169117647058826)]
y: [np.float64(-11.353383458646618), np.float64(-11.477941176470594)] z
[np.float64(57.87969924812029), np.float64(54.95588235294119)]
Conexión: x: [np.float64(50.06896551724138), np.float64(48.9025974025974)] y:
[np.float64(7.400000000000006), np.float64(0.006493506493512768)] z
[np.float64(40.28965517241379), np.float64(44.461038961038966)]
Conexión: x: [np.float64(40.169117647058826), np.float64(45.4527027027027)] y:
[np.float64(-11.477941176470594), np.float64(-12.891891891891888)] z
[np.float64(54.95588235294119), np.float64(48.12837837837837)]
Conexión: x: [np.float64(-5.976095617529879), np.float64(-6.972111553784856)]
y: [np.float64(-57.69721115537848), np.float64(-63.31474103585657)] z
[np.float64(31.434262948207177), np.float64(27.856573705179287)]
Conexión: x: [np.float64(14.458181818181814), np.float64(9.978339350180505)]
y: [np.float64(-61.18545454545455), np.float64(-55.277978339350184)] z
[np.float64(24.35636363636364), np.float64(27.53068592057761)]
Conexión: x: [np.float64(9.229508196721312), np.float64(7.095435684647299)] y:
[np.float64(-60.68852459016394), np.float64(-53.82572614107883)] z
[np.float64(40.88524590163935), np.float64(45.468879668049794)]
Conexión: x: [np.float64(13.460396039603964), np.float64(15.899705014749259)]
y: [np.float64(-48.069306930693074), np.float64(-56.90265486725664)] z
[np.float64(11.133663366336634), np.float64(11.722713864306783)]
Conexión: x: [np.float64(15.899705014749259), np.float64(19.481707317073173)]
y: [np.float64(-56.90265486725664), np.float64(-63.11585365853658)] z
[np.float64(11.722713864306783), np.float64(9.810975609756099)]
Conexión: x: [np.float64(11.1958041958042), np.float64(16.34042553191489)] y:
[np.float64(-85.1048951048951), np.float64(-79.55623100303951)] z
[np.float64(10.272727272727266), np.float64(3.920972644376903)]
Conexión: x: [np.float64(30.666666666666664), np.float64(25.544117647058826)]
y: [np.float64(2.86597938144331), np.float64(5.330882352941188)] z
[np.float64(5.319587628865975), np.float64(2.286764705882348)]
Conexión: x: [np.float64(-4.392452830188674), np.float64(-4.782608695652172)]
y: [np.float64(49.16981132075472), np.float64(39.565217391304344)] z
[np.float64(-17.932075471698113), np.float64(-21.762845849802375)]
Conexión: x: [np.float64(-45.52287581699346), np.float64(-48.415430267062305)]
y: [np.float64(-14.169934640522882), np.float64(-15.821958456973292)] z
[np.float64(12.503267973856211), np.float64(14.492581602373889)]
Conexión: x: [np.float64(57.5108359133127), np.float64(57.832699619771866)] y:
[np.float64(-13.708978328173373), np.float64(-20.00760456273764)] z

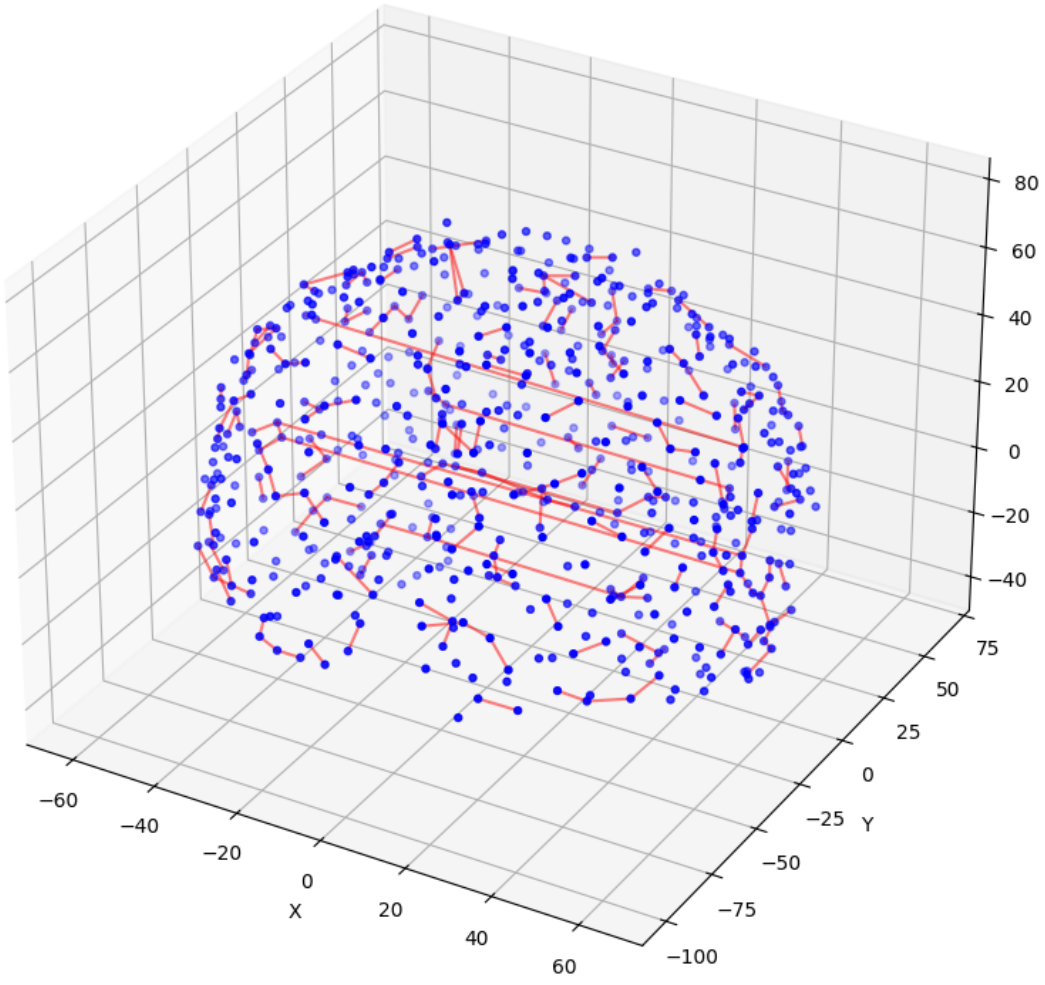
[np.float64(13.87616099071208), np.float64(9.140684410646386)]
Conexión: x: [np.float64(-2.606837606837601), np.float64(-4.55230125523012)]
y: [np.float64(12.0), np.float64(17.054393305439334)] z
[np.float64(55.27350427350427), np.float64(47.84937238493724)]
Conexión: x: [np.float64(-2.606837606837601), np.float64(7.942446043165461)]
y: [np.float64(12.0), np.float64(14.050359712230204)] z
[np.float64(55.27350427350427), np.float64(49.99280575539568)]
Conexión: x: [np.float64(-2.606837606837601), np.float64(6.018018018018012)]
y: [np.float64(12.0), np.float64(9.360360360360374)] z
[np.float64(55.27350427350427), np.float64(59.29729729729729)]
Conexión: x: [np.float64(12.732283464566933), np.float64(7.577889447236174)]
y: [np.float64(14.20472440944883), np.float64(12.050251256281399)] z
[np.float64(64.78740157480314), np.float64(64.0)]
Conexión: x: [np.float64(-54.13654618473896), np.float64(-55.79919678714859)]
y: [np.float64(-46.18473895582329), np.float64(-43.164658634538156)] z
[np.float64(30.955823293172685), np.float64(25.510040160642575)]
Conexión: x: [np.float64(59.506726457399104), np.float64(58.30711610486891)]
y: [np.float64(-38.843049327354265), np.float64(-37.752808988764045)] z
[np.float64(26.896860986547082), np.float64(19.101123595505612)]
Conexión: x: [np.float64(61.65853658536585), np.float64(61.24229074889868)] y:
[np.float64(-24.512195121951223), np.float64(-20.22026431718062)] z
[np.float64(22.910569105691053), np.float64(28.951541850220266)]
Conexión: x: [np.float64(54.384313725490195), np.float64(58.20083682008368)]
y: [np.float64(-28.031372549019608), np.float64(-32.32635983263599)] z
[np.float64(38.831372549019605), np.float64(30.35146443514644)]
Conexión: x: [np.float64(-45.30000000000001), np.float64(-42.39603960396039)]
y: [np.float64(0.39090909090910486), np.float64(5.069306930693074)] z
[np.float64(-38.31818181818182), np.float64(-36.71287128712871)]
Conexión: x: [np.float64(-56.61290322580646), np.float64(-50.9469964664311)]
y: [np.float64(-49.5241935483871), np.float64(-53.90106007067138)] z
[np.float64(-14.975806451612904), np.float64(-19.32862190812721)]
Conexión: x: [np.float64(-54.042735042735046), np.float64(-52.90836653386455)]
y: [np.float64(-10.299145299145295), np.float64(-3.04382470119522)] z
[np.float64(-30.24786324786325), np.float64(-26.677290836653384)]
Conexión: x: [np.float64(59.5), np.float64(62.890625)] y:
[np.float64(-19.75), np.float64(-14.375)] z [np.float64(-25.909090909090907),
np.float64(-19.515625)]
Conexión: x: [np.float64(50.76086956521739), np.float64(54.70918367346939)] y:
[np.float64(-63.91847826086956), np.float64(-57.95408163265306)] z
[np.float64(-13.570652173913047), np.float64(-8.25)]
Conexión: x: [np.float64(-54.6778523489933), np.float64(-50.021052631578954)]
y: [np.float64(-5.429530201342288), np.float64(-1.431578947368422)] z
[np.float64(-15.583892617449663), np.float64(-10.280701754385966)]
Conexión: x: [np.float64(-52.90836653386455), np.float64(-51.60337552742615)]
y: [np.float64(-3.04382470119522), np.float64(1.7974683544303787)] z
[np.float64(-26.677290836653384), np.float64(-24.42194092827004)]
Conexión: x: [np.float64(-59.12863070539419), np.float64(-59.29966329966331)]
y: [np.float64(-18.116182572614107), np.float64(-15.919191919191917)] z

[np.float64(-11.219917012448136), np.float64(-18.96969696969697)]
Conexión: x: [np.float64(-59.5655172413793), np.float64(-53.07499999999999)]
y: [np.float64(-16.868965517241378), np.float64(-18.299999999999997)] z
[np.float64(-3.737931034482756), np.float64(3.191666666666663)]
Conexión: x: [np.float64(-54.33600000000001), np.float64(-52.994818652849744)]
y: [np.float64(-57.592), np.float64(-54.73575129533678)] z
[np.float64(-0.9440000000000026), np.float64(6.186528497409327)]
Conexión: x: [np.float64(-51.60337552742615), np.float64(-48.079681274900395)]
y: [np.float64(1.7974683544303787), np.float64(12.621513944223096)] z
[np.float64(-24.42194092827004), np.float64(-31.322709163346616)]
Conexión: x: [np.float64(62.57959183673469), np.float64(61.048780487804876)]
y: [np.float64(-34.90612244897959), np.float64(-39.08943089430895)] z
[np.float64(-3.395918367346937), np.float64(-9.22764227642277)]
Conexión: x: [np.float64(59.34166666666667), np.float64(59.86065573770492)] y:
[np.float64(-39.18333333333334), np.float64(-32.786885245901644)] z
[np.float64(4.275000000000006), np.float64(7.729508196721312)]
Conexión: x: [np.float64(57.508532423208194), np.float64(56.84137931034483)]
y: [np.float64(0.8873720136518841), np.float64(-3.4758620689655118)] z
[np.float64(-21.351535836177476), np.float64(-25.172413793103445)]
Conexión: x: [np.float64(57.508532423208194), np.float64(57.92825112107624)]
y: [np.float64(0.8873720136518841), np.float64(9.659192825112115)] z
[np.float64(-21.351535836177476), np.float64(-21.98206278026906)]
Conexión: x: [np.float64(51.20987654320987), np.float64(47.654901960784315)]
y: [np.float64(-60.592592592592595), np.float64(-68.11764705882354)] z
[np.float64(18.288065843621396), np.float64(16.572549019607848)]
Conexión: x: [np.float64(54.94308943089431), np.float64(55.2129963898917)] y:
[np.float64(-50.71544715447155), np.float64(-47.068592057761734)] z
[np.float64(14.113821138211378), np.float64(18.12274368231047)]
Conexión: x: [np.float64(64.15899581589957), np.float64(62.82527881040892)] y:
[np.float64(-24.26778242677824), np.float64(-27.992565055762086)] z
[np.float64(-8.31799163179916), np.float64(-11.278810408921935)]
Conexión: x: [np.float64(62.170212765957444), np.float64(56.56677524429968)]
y: [np.float64(-13.821276595744678), np.float64(-5.433224755700323)] z
[np.float64(-12.970212765957449), np.float64(-10.54071661237785)]
Conexión: x: [np.float64(62.890625), np.float64(56.84137931034483)] y:
[np.float64(-14.375), np.float64(-3.4758620689655118)] z
[np.float64(-19.515625), np.float64(-25.172413793103445)]
Conexión: x: [np.float64(45.45528455284553), np.float64(38.18018018018018)] y:
[np.float64(14.21951219512195), np.float64(17.405405405405418)] z
[np.float64(-37.3739837398374), np.float64(-36.972972972972975)]
Conexión: x: [np.float64(55.50597609561753), np.float64(53.11372549019608)] y:
[np.float64(12.239043824701184), np.float64(13.858823529411751)] z
[np.float64(-16.98804780876494), np.float64(-11.788235294117648)]
Conexión: x: [np.float64(-52.10788381742739), np.float64(-55.94238683127571)]
y: [np.float64(-22.43983402489627), np.float64(-27.588477366255148)] z
[np.float64(10.049792531120332), np.float64(8.156378600823047)]
Conexión: x: [np.float64(-52.10788381742739), np.float64(-53.07499999999999)]
y: [np.float64(-22.43983402489627), np.float64(-18.299999999999997)] z

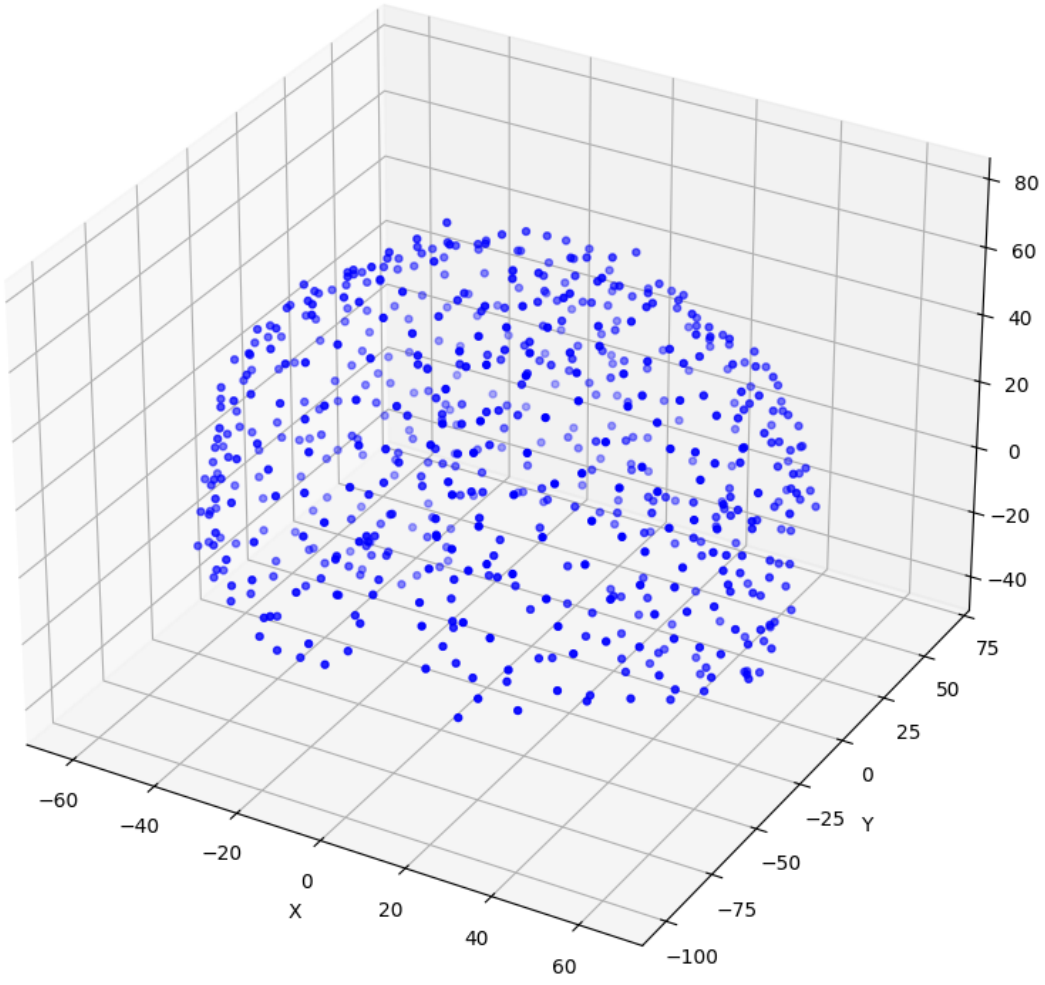
[np.float64(10.049792531120332), np.float64(3.191666666666663)]
Conexión: x: [np.float64(-52.10788381742739), np.float64(57.832699619771866)]
y: [np.float64(-22.43983402489627), np.float64(-20.00760456273764)] z
[np.float64(10.049792531120332), np.float64(9.140684410646386)]
Conexión: x: [np.float64(-53.291338582677156),
np.float64(-50.021052631578954)] y: [np.float64(-3.3622047244094517),
np.float64(-1.431578947368422)] z [np.float64(-4.314960629921259),
np.float64(-10.280701754385966)]
Conexión: x: [np.float64(-53.82517482517483), np.float64(-55.14960629921259)]
y: [np.float64(-44.328671328671334), np.float64(-36.7244094488189)] z
[np.float64(16.27972027972028), np.float64(19.039370078740163)]
Conexión: x: [np.float64(-53.07499999999999), np.float64(56.44843049327354)]
y: [np.float64(-18.299999999999997), np.float64(-17.38116591928251)] z
[np.float64(3.191666666666663), np.float64(2.1883408071748818)]
Conexión: x: [np.float64(-55.14960629921259), np.float64(-55.87179487179486)]
y: [np.float64(-36.7244094488189), np.float64(-33.91452991452991)] z
[np.float64(19.039370078740163), np.float64(12.777777777777771)]
Conexión: x: [np.float64(58.878787878787875), np.float64(57.991379310344826)]
y: [np.float64(-6.287878787878782), np.float64(-4.09482758620689)] z
[np.float64(1.3333333333333286), np.float64(-5.206896551724142)]
Conexión: x: [np.float64(59.83399209486166), np.float64(58.30711610486891)] y:
[np.float64(-35.70750988142292), np.float64(-37.752808988764045)] z
[np.float64(13.296442687747032), np.float64(19.101123595505612)]
Conexión: x: [np.float64(59.86065573770492), np.float64(57.53310104529617)] y:
[np.float64(-32.786885245901644), np.float64(-25.54703832752614)] z
[np.float64(7.729508196721312), np.float64(1.1777003484320545)]
Conexión: x: [np.float64(57.991379310344826), np.float64(56.56677524429968)]
y: [np.float64(-4.09482758620689), np.float64(-5.433224755700323)] z
[np.float64(-5.206896551724142), np.float64(-10.54071661237785)]
Conexión: x: [np.float64(57.832699619771866), np.float64(56.44843049327354)]
y: [np.float64(-20.00760456273764), np.float64(-17.38116591928251)] z
[np.float64(9.140684410646386), np.float64(2.1883408071748818)]
Conexión: x: [np.float64(56.44843049327354), np.float64(58.86852589641434)] y:
[np.float64(-17.38116591928251), np.float64(-15.81673306772909)] z
[np.float64(2.1883408071748818), np.float64(-3.6414342629482093)]
Conexión: x: [np.float64(58.86852589641434), np.float64(56.56677524429968)] y:
[np.float64(-15.81673306772909), np.float64(-5.433224755700323)] z
[np.float64(-3.6414342629482093), np.float64(-10.54071661237785)]
Conexión: x: [np.float64(-15.400000000000006),
np.float64(-14.693140794223822)] y: [np.float64(-25.599999999999994),
np.float64(-15.653429602888082)] z [np.float64(8.48571428571428),
np.float64(7.812274368231044)]
Conexión: x: [np.float64(-14.693140794223822), np.float64(-7.268656716417908)]
y: [np.float64(-15.653429602888082), np.float64(-11.007462686567166)] z
[np.float64(7.812274368231044), np.float64(5.597014925373131)]
Conexión: x: [np.float64(-7.268656716417908), np.float64(8.902255639097746)]
y: [np.float64(-11.007462686567166), np.float64(-11.94736842105263)] z
[np.float64(5.597014925373131), np.float64(3.744360902255636)]

```
Conexión:  x:  [np.float64(11.120300751879697), np.float64(8.902255639097746)]
y:  [np.float64(-20.21052631578948), np.float64(-11.94736842105263)] z
[np.float64(11.954887218045116), np.float64(3.744360902255636)]
Conexión:  x:  [np.float64(11.120300751879697), np.float64(13.646153846153851)]
y:  [np.float64(-20.21052631578948), np.float64(-27.030769230769238)] z
[np.float64(11.954887218045116), np.float64(4.461538461538467)]
Conexión:  x:  [np.float64(8.902255639097746), np.float64(17.184905660377353)]
y:  [np.float64(-11.94736842105263), np.float64(-16.06037735849057)] z
[np.float64(3.744360902255636), np.float64(6.701886792452825)]
Conexión:  x:  [np.float64(-3.418604651162795), np.float64(-4.538745387453872)]
y:  [np.float64(39.92248062015503), np.float64(46.53874538745387)] z
[np.float64(10.170542635658919), np.float64(3.06273062730628)]
Para el grafo 0.8, existen las siguientes coordenadas de conexiones:

Para el grafo 0.9, existen las siguientes coordenadas de conexiones:

Para el grafo 1, existen las siguientes coordenadas de conexiones:
```

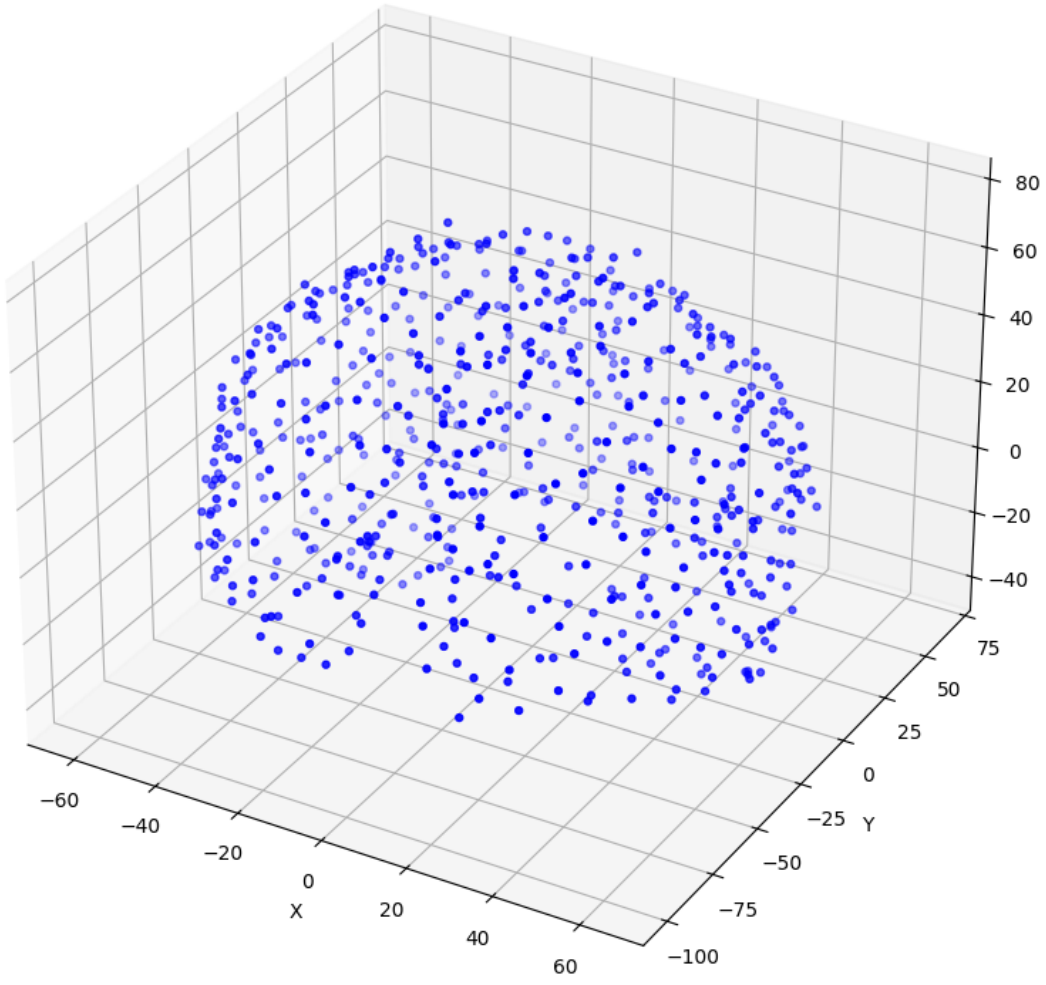
Grafo con un umbral de 0.2

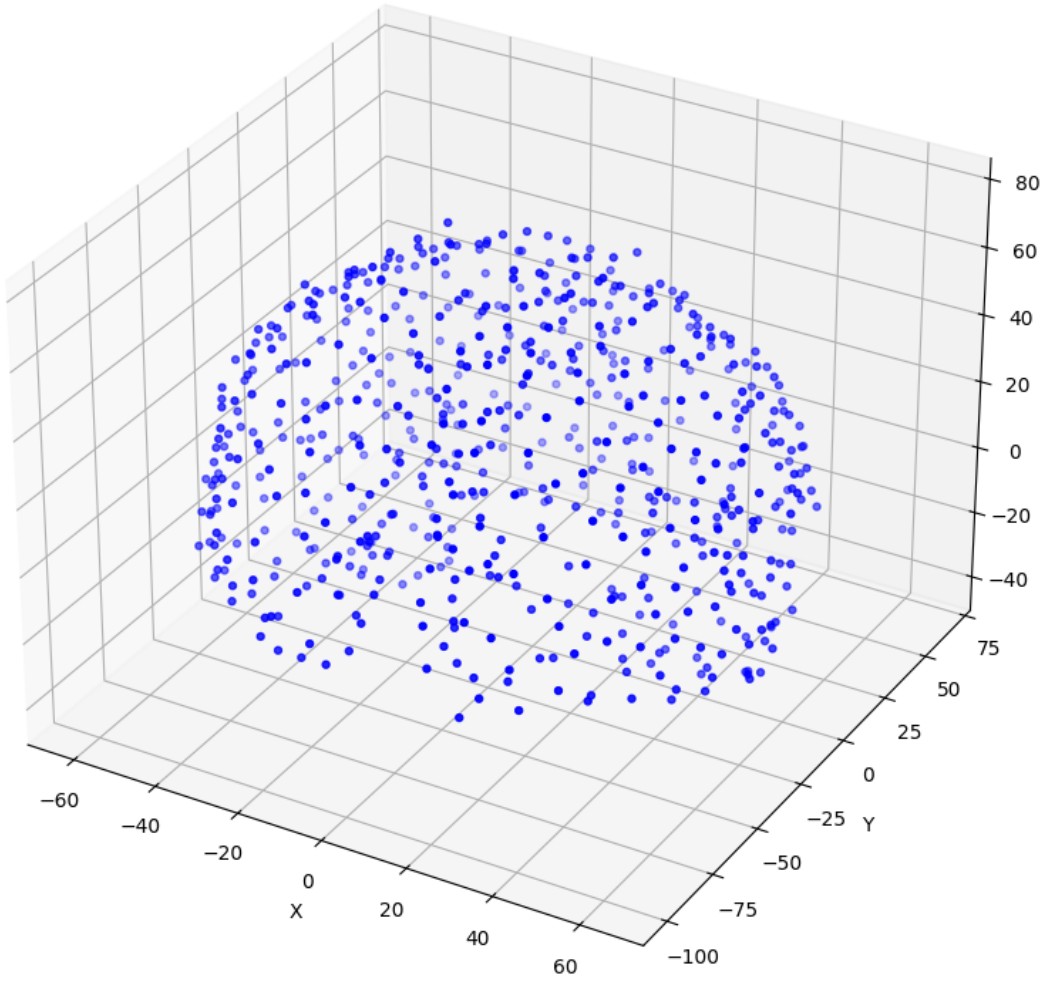


Grafo con un umbral de 0.8



Grafo con un umbral de 0.9



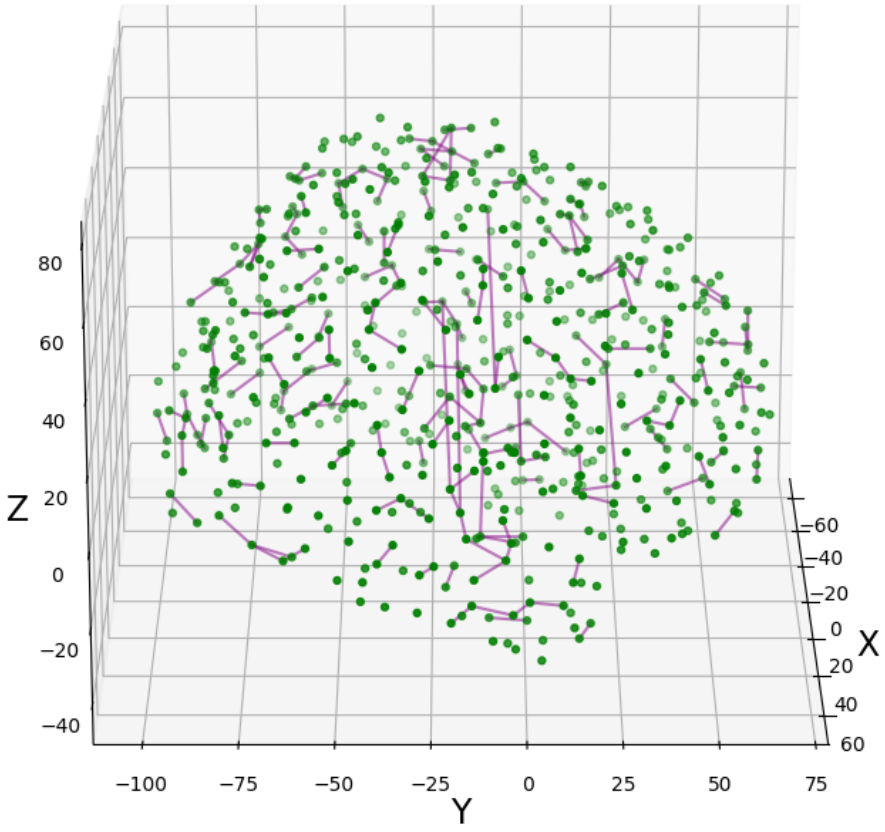


1.0.2 Con uno de los grafos en el punto uno con umbral 0.9, generar una animacion donde se haga girar el grafo del cerebro para visualizar las conexiones establecidas.

Se utilizó un umbral de 0.2 porque con el resto de umbrales no se observaban las conexiones.

[36]:
```
from matplotlib.animation import FuncAnimation, PillowWriter
from IPython.display import Image
import scipy.io as sp
import numpy as np
import pandas as pd
```

```
import networkx as nx
import matplotlib.pyplot as plt
import seaborn as sns

data = sp.loadmat(r"C:
 ↪\Users\dnlls\OneDrive\Escritorio\PUMA\Neurociencias\Escritorio␣
 ↪Neurociencias\5o semestre\Modelos\PROYECTO FINAL MODELOS␣
 ↪dump\FinalModelosCode\Coactivation_matrix.mat")
coactivation_matrix = data['Coactivation_matrix']
coord = data['Coord']
x, y, z = coord[:, 0], coord[:, 1], coord[:, 2]
filtered_matrix= np.where(coactivation_matrix >= 0.2, coactivation_matrix, 0)
G= nx.from_numpy_array(filtered_matrix)
fig = plt.figure(figsize=(10, 10))
ax = fig.add_subplot(projection='3d')
ax.scatter(x, y, z, c='green', s=50, marker='.')

for edge in G.edges():
    node1, node2 = edge
    x_coords = [x[node1], x[node2]]
    y_coords = [y[node1], y[node2]]
    z_coords = [z[node1], z[node2]]
    ax.plot(x_coords, y_coords, z_coords, c='purple', alpha=0.5)

ax.set_title(f'Grafo con un umbral de 0.2', fontsize= 30)
ax.set_xlabel('X', fontsize= 17)
ax.set_ylabel('Y', fontsize= 17)
ax.set_zlabel('Z', fontsize= 17)


def init():
    ax.view_init(elev=20, azim=0)
    return fig,

def animate(frame):
    ax.view_init(elev=20, azim=frame)
    return fig,

animacion = FuncAnimation(fig, animate, init_func=init, frames=360,␣
 ↪interval=50, blit=False)
```

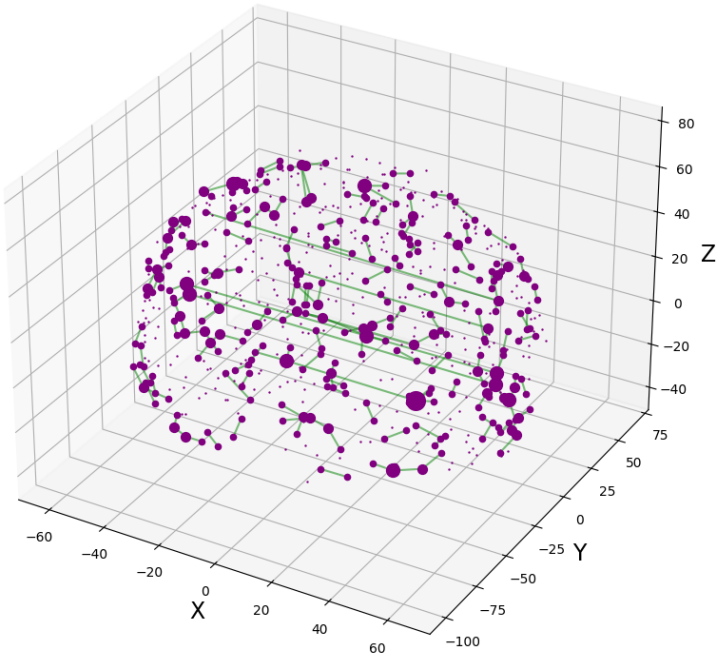


1.0.3 Encontrar los hubs del grafo, y establecer el tamaño del nodo proporcional al valor del grado

[37]:
```
grados = dict(G.degree())
lista_grados = list(grados.values())
grado_maximo = max(lista_grados)
grado_minimo = min(lista_grados)

tam_max = 1
tam_min = 100
```

```
grado_tamaño = {0:0.50, 1:20, 2:50, 3:100, 4:200}

c_m_02 = data['Coactivation_matrix']
filtered_matrix_02 = np.where(c_m_02 >= 0.2, c_m_02, 0)
G02= nx.from_numpy_array(filtered_matrix)
G01= nx.Graph()
x, y, z = coord[:, 0], coord[:, 1], coord[:, 2]

figura = plt.figure(figsize= (10, 10))
Ax = figura.add_subplot(projection= '3d')

for nodo in grados.keys():
    valor= grados[nodo]
    tamaño= grado_tamaño[valor]
    Ax.scatter(x[nodo], y[nodo], z[nodo], color= 'purple', s= tamaño)

for edge in G02.edges():
    node1, node2 = edge
    x_coords = [x[node1], x[node2]]
    y_coords = [y[node1], y[node2]]
    z_coords = [z[node1], z[node2]]
    Ax.plot(x_coords, y_coords, z_coords, c='green', alpha=0.5)

Ax.set_title(f'Grafo 3D con los nodos presentados de distinto tamaño',␣
 ↪fontsize= 30)
Ax.set_xlabel('X', fontsize= 17)
Ax.set_ylabel('Y', fontsize= 17)
Ax.set_zlabel('Z', fontsize= 17)
```

[37]: Text(0.5, 0, 'Z')



[38]:
```
Umbral = np.percentile(list(grados.values()), 99)
Hubs = [node for node, grado in grados.items() if grado >= Umbral]
print(f"Existen {len(Hubs)} Hubs. Los Hubs son los nodos: {Hubs}", f'\nEl␣
 ↪umbral considerado fue de {Umbral}')
```

```
Existen 11 Hubs. Los Hubs son los nodos: [217, 226, 263, 404, 482, 599, 603,
615, 616, 618, 629]
El umbral considerado fue de 3.0
```

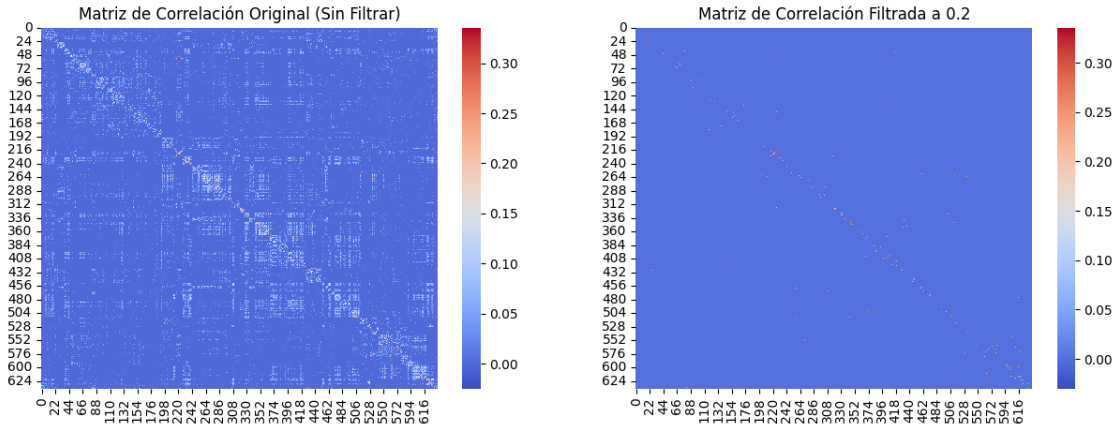
1.0.4 En funcion de la matriz de emparejamiento(correlacion de la matriz de adyacencia), establecer una particion de los nodos en modulos. Escoger el numero de modulos que creas conveniente y justificar por que escogiste ese numero.

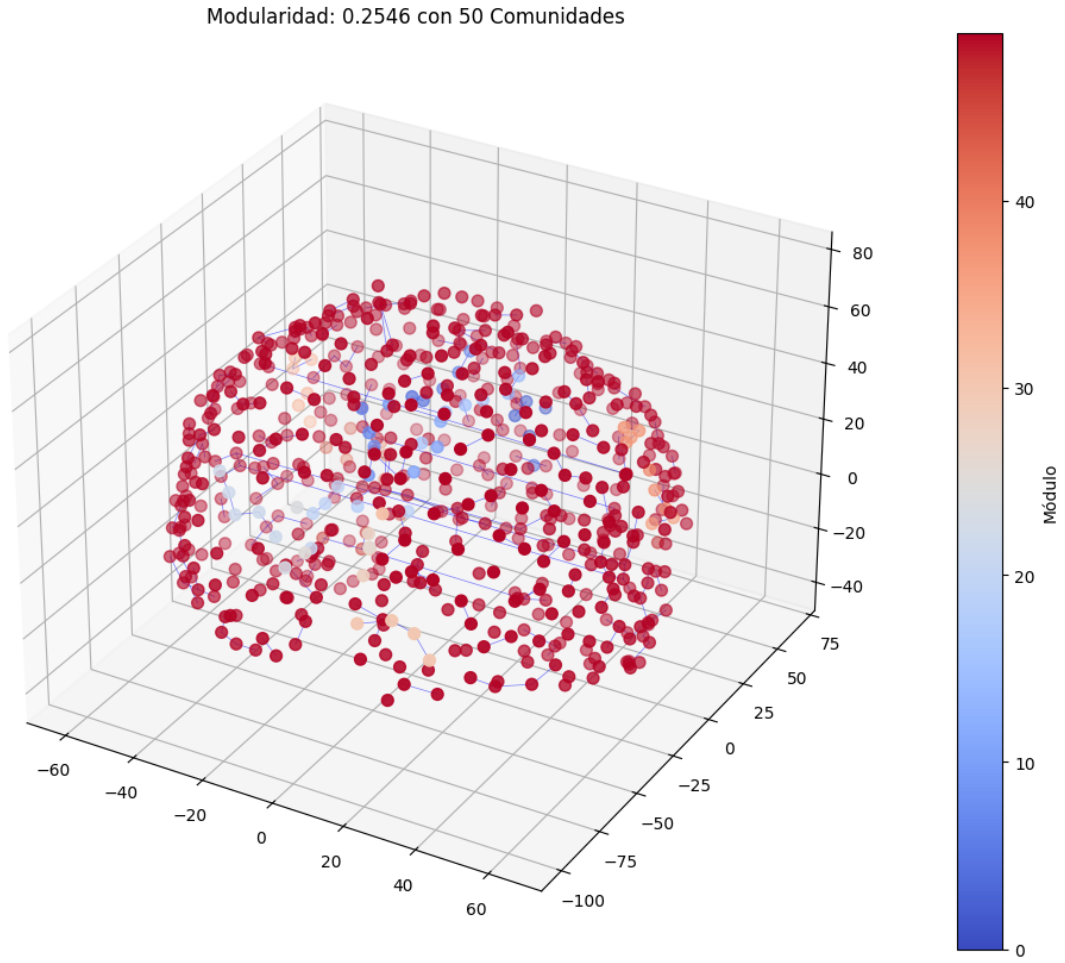
[39]:
```
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import networkx as nx
from networkx.algorithms.community import label_propagation_communities

c_m_03 = data['Coactivation_matrix']
coord = data['Coord']
x, y, z = coord[:, 0], coord[:, 1], coord[:, 2]
```

```
filtered_matrix_03 = np.where(c_m_03 >= 0.2, c_m_03, 0)
corrmat_sf = np.corrcoef(c_m_03)
corrmat = np.corrcoef(filtered_matrix_03)

plt.figure(figsize=(15, 5))
plt.subplot(1, 2, 1)
sns.heatmap(c_m_03, cmap='coolwarm', vmin=-0.025)
plt.title("Matriz de Correlación Original (Sin Filtrar)")

plt.subplot(1, 2, 2)
sns.heatmap(filtered_matrix_03, cmap='coolwarm', vmin=-0.03)
plt.title("Matriz de Correlación Filtrada a 0.2")
plt.show()

G = nx.from_numpy_array(filtered_matrix_03)

def calcular_modularidad(G, num_comunidades):
    comunidades_encontradas = list(label_propagation_communities(G))

    while len(comunidades_encontradas) > num_comunidades:
        comunidades_encontradas[-2].update(comunidades_encontradas[-1])
        comunidades_encontradas = comunidades_encontradas[:-1]

    while len(comunidades_encontradas) < num_comunidades:
        comunidades_encontradas[0].add(min(G.nodes()))

    particion_nodos = {nodo: i for i, comunidad in␣
↪enumerate(comunidades_encontradas) for nodo in comunidad}

    modularidad = nx.algorithms.community.quality.modularity(G,␣
↪comunidades_encontradas)
    colores_nodos = [particion_nodos[nodo] for nodo in G.nodes()]

    figura = plt.figure(figsize=(18, 10))
    ejes = figura.add_subplot(111, projection='3d')
    posiciones_nodos = {i: (x, y, z) for i, (x, y, z) in enumerate(coord)}
    x_nodos, y_nodos, z_nodos = np.array(list(posiciones_nodos.values())).T

    dispersión = ejes.scatter(x_nodos, y_nodos, z_nodos, c=colores_nodos,␣
↪cmap="coolwarm", s=50, label='Nodos')

    for arista in G.edges():
        x_arista, y_arista, z_arista = zip(posiciones_nodos[arista[0]],␣
↪posiciones_nodos[arista[1]])
        ejes.plot(x_arista, y_arista, z_arista, c='b', alpha=0.5, linewidth=0.5)
```

```
    ejes.set_title(f"Modularidad: {modularidad:.4f} con {num_comunidades}␣
 ↪Comunidades")
    plt.colorbar(dispersión, ax=ejes, label="Módulo")
    plt.show()

    return modularidad

num_comunidades = 50

modularidad = calcular_modularidad(G, num_comunidades)

print(f"Modularidad con {num_comunidades} comunidades: {modularidad:.4f}")
```

```
C:\Users\dnlls\AppData\Roaming\Python\Python39\site-
packages\numpy\lib\_function_base_impl.py:2922: RuntimeWarning: invalid value
encountered in divide
  c /= stddev[:, None]
C:\Users\dnlls\AppData\Roaming\Python\Python39\site-
packages\numpy\lib\_function_base_impl.py:2923: RuntimeWarning: invalid value
encountered in divide
  c /= stddev[None, :]
```






```
Modularidad con 50 comunidades: 0.2546
```

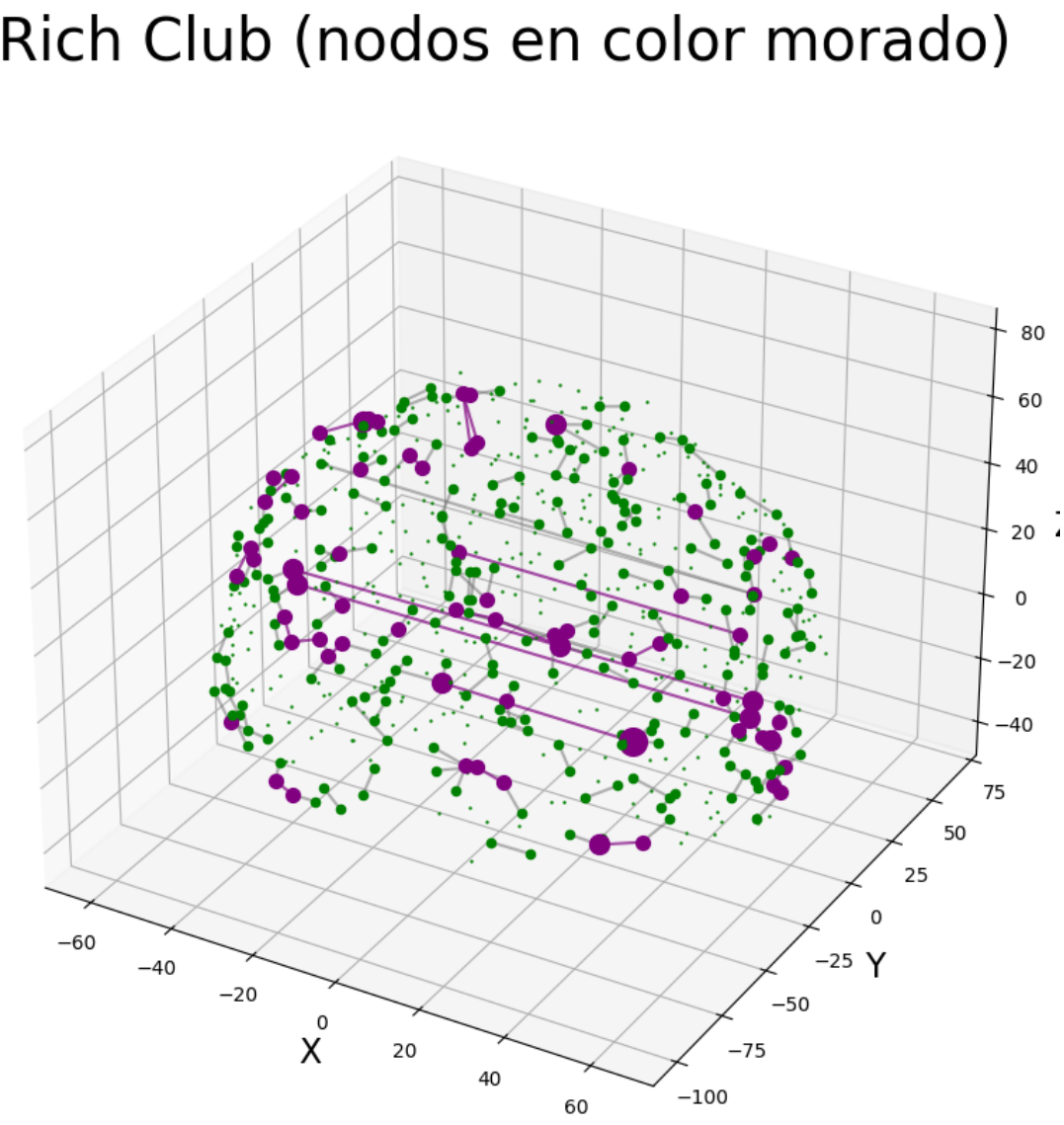
Elegí 50 módulos porque con este número obtuve una alta modularidad, lo que indica una partición clara y bien estructurada de los nodos en comunidades. El valor de modularidad sugiere que las relaciones dentro de las comunidades son más fuertes que entre ellas, lo que mejora la calidad de la partición.

1.0.5 Determinar el conjunto de Rich Club y discutir las implicaciones anatomicas y funcionales de este grupo de nodos (minimo 100 palabras)

[41]:
```
grades = dict(G02.degree())

threshold = np.percentile(list(grades.values()), 90)
rich_club = [nodo for nodo, grado in grades.items() if grado >= threshold]

print(f"Nodos Rich Club (grado  {threshold}): {rich_club}")

tamaño_nodo = {0: 0.50, 1: 20, 2: 50, 3: 100, 4: 200}
```

```
figura = plt.figure(figsize=(10, 10))
ejes = figura.add_subplot(projection='3d')

for nodo, valor in grades.items():
    tamaño = tamaño_nodo.get(valor, 10)
    color = 'purple' if nodo in rich_club else 'green'
    ejes.scatter(x[nodo], y[nodo], z[nodo], color=color, s=tamaño)

for nodo1, nodo2 in G02.edges():
    coordenadas_x = [x[nodo1], x[nodo2]]
    coordenadas_y = [y[nodo1], y[nodo2]]
    coordenadas_z = [z[nodo1], z[nodo2]]

    color_arista = 'purple' if nodo1 in rich_club and nodo2 in rich_club else␣
 ↪'dimgray'
    alpha_arista = 0.7 if nodo1 in rich_club and nodo2 in rich_club else 0.5
    ejes.plot(coordenadas_x, coordenadas_y, coordenadas_z, c=color_arista,␣
 ↪alpha=alpha_arista, linewidth=1.5)

ejes.set_title('Rich Club (nodos en color morado)', fontsize=30)
ejes.set_xlabel('X', fontsize=17)
ejes.set_ylabel('Y', fontsize=17)
ejes.set_zlabel('Z', fontsize=17)

plt.show()
```

```
Nodos Rich Club (grado  2.0): [38, 42, 62, 69, 92, 124, 150, 217, 221, 223,
226, 230, 235, 245, 259, 260, 263, 275, 281, 290, 292, 296, 304, 309, 319, 320,
321, 327, 330, 331, 344, 352, 356, 358, 369, 393, 404, 407, 408, 416, 419, 428,
456, 460, 465, 482, 532, 547, 561, 565, 574, 579, 599, 603, 604, 605, 611, 613,
614, 615, 616, 617, 618, 622, 623, 628, 629]
```



Según el grafo 3D generado, los hubs cerebrales se concentran en el área parieto-occipito-temporal y cerca del giro temporal superior debido a su rol clave en la integración sensorial y cognitiva. Estas regiones son esenciales para procesar información de múltiples modalidades sensoriales, como la visión, la audición y el tacto, lo que permite una percepción coherente del entorno. Además, el giro temporal superior está involucrado en funciones cognitivas complejas como la memoria, el lenguaje, la atención y la toma de decisiones, que requieren una alta conectividad entre diferentes áreas del cerebro.

Estas áreas actúan como puntos de convergencia, conectando redes cerebrales especializadas y facilitando la comunicación eficiente entre áreas distantes. La alta conectividad de los hubs en estas zonas optimiza el procesamiento de información en tareas complejas y en la formación de recuerdos. Además, el giro temporal superior facilita la coordinación interhemisférica, permitiendo

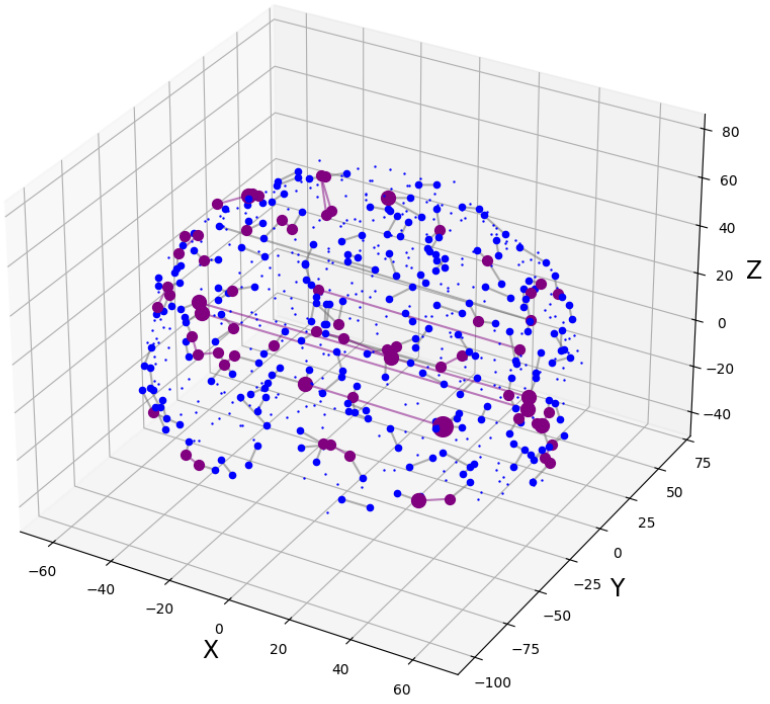
la integración de funciones cognitivas entre los hemisferios derecho e izquierdo.

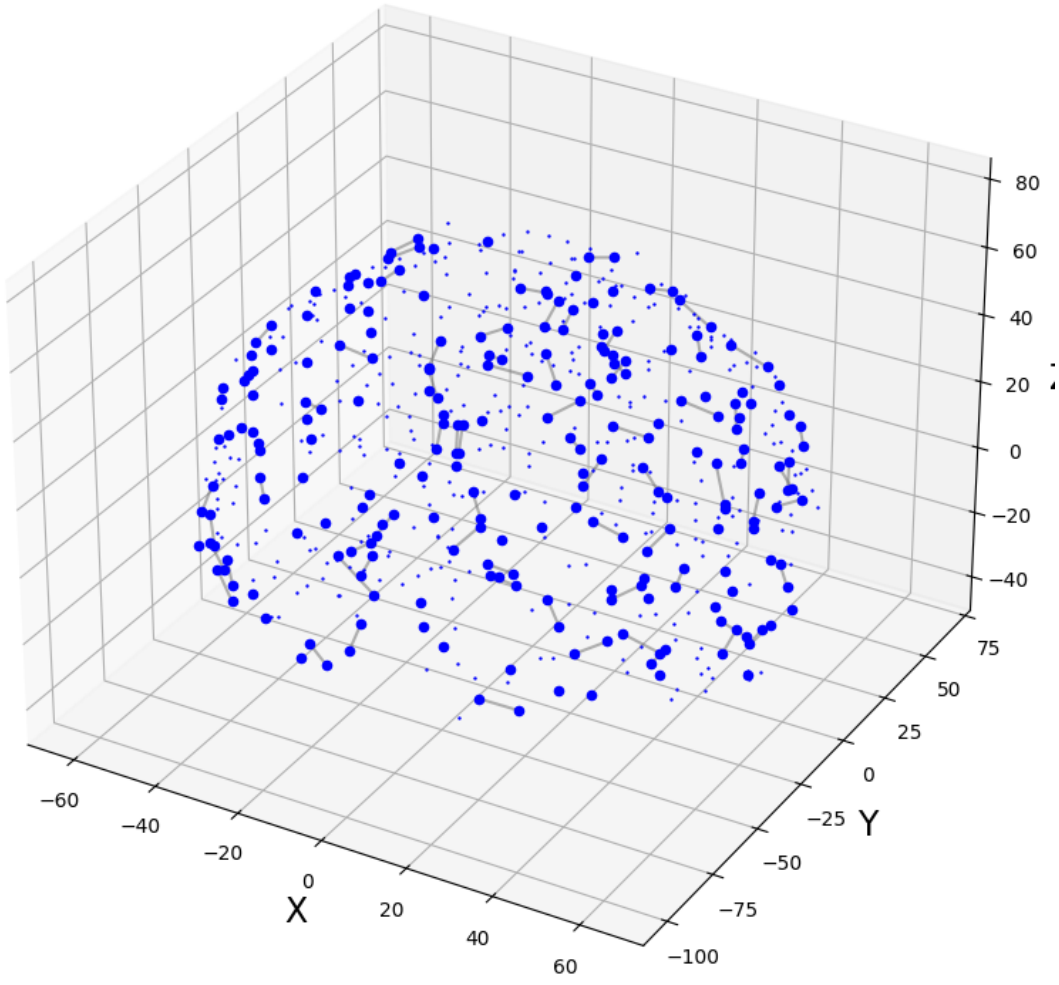
1.0.6 Supongamos que eliminamos los nodos del Rich Club, describir como cambian las propiedades topologicas del grafo, hacer comparativas del grado, coeficiente de cluster, coeficiente de mundo pequeño y las medidas de centralidad (cercania, intermediacion)

[42]:
```
grades = dict(G02.degree())

threshold = np.percentile(list(grades.values()), 90)

rich_club = [nodo for nodo, grado in grades.items() if grado >= threshold]

tamaño_nodo = {0: 0.50, 1: 20, 2: 50, 3: 100, 4: 200}

figura = plt.figure(figsize=(10, 10))
eje = figura.add_subplot(projection='3d')

for nodo in grades:
    color = 'purple' if nodo in rich_club else 'blue'
    tamaño = tamaño_nodo.get(grades[nodo], 10)
    eje.scatter(x[nodo], y[nodo], z[nodo], color=color, s=tamaño)

for nodo1, nodo2 in G02.edges():
    x_arista = [x[nodo1], x[nodo2]]
    y_arista = [y[nodo1], y[nodo2]]
    z_arista = [z[nodo1], z[nodo2]]
    color_arista = 'purple' if nodo1 in rich_club and nodo2 in rich_club else␣
 ↪'dimgray'
    eje.plot(x_arista, y_arista, z_arista, c=color_arista, alpha=0.5,␣
 ↪linewidth=1.5)

eje.set_title('Grafo 3D con nodos Rich Club resaltados en morado', fontsize=30)
eje.set_xlabel('X', fontsize=17)
eje.set_ylabel('Y', fontsize=17)
eje.set_zlabel('Z', fontsize=17)

plt.show()

G_sin_rich_club = G02.copy()
G_sin_rich_club.remove_nodes_from(rich_club)

figura = plt.figure(figsize=(10, 10))
eje = figura.add_subplot(projection='3d')

for nodo in G_sin_rich_club.nodes():
    tamaño = tamaño_nodo.get(grades.get(nodo, 0), 10)
```

```
    eje.scatter(x[nodo], y[nodo], z[nodo], color='blue', s=tamaño)

for nodo1, nodo2 in G_sin_rich_club.edges():
    x_arista = [x[nodo1], x[nodo2]]
    y_arista = [y[nodo1], y[nodo2]]
    z_arista = [z[nodo1], z[nodo2]]
    eje.plot(x_arista, y_arista, z_arista, c='dimgray', alpha=0.5)

eje.set_title('Grafo 3D sin nodos Rich Club', fontsize=30)
eje.set_xlabel('X', fontsize=17)
eje.set_ylabel('Y', fontsize=17)
eje.set_zlabel('Z', fontsize=17)

plt.show()
```

Grafo 3D con nodos Rich Club resaltados en morado





[43]:
```
grados_nodos = dict(G02.degree())

umbral_grado = np.percentile(list(grados_nodos.values()), 90)

nodos_rich_club = [nodo for nodo, grado in grados_nodos.items() if grado >=␣
↪umbral_grado]

grafo_sin_rich_club = G02.copy()
grafo_sin_rich_club.remove_nodes_from(nodos_rich_club)

def obtener_propiedades(grafo, nombre_grafo):
```

```
    propiedades = {
        "Número de nodos": grafo.number_of_nodes(),
        "Número de aristas": grafo.number_of_edges(),
        "Grado promedio": sum(dict(grafo.degree()).values()) / grafo.
 ↪number_of_nodes(),
        "Coeficiente de clustering promedio": nx.average_clustering(grafo),
        "Centralidad de cercanía (promedio)": np.mean(list(nx.
 ↪closeness_centrality(grafo).values())),
        "Centralidad de intermediación (promedio)": np.mean(list(nx.
 ↪betweenness_centrality(grafo).values()))
    }

    print(f"Propiedades del grafo {nombre_grafo}:")
    for propiedad, valor in propiedades.items():
        print(f"  {propiedad}: {valor}")
    print()

    return propiedades

propiedades_grafo_original = obtener_propiedades(G02, 'original')

propiedades_grafo_sin_rich_club = obtener_propiedades(grafo_sin_rich_club, 'sin␣
 ↪nodos Rich Club')
```

```
Propiedades del grafo original:
  Número de nodos: 638
  Número de aristas: 188
  Grado promedio: 0.5893416927899686
  Coeficiente de clustering promedio: 0.004702194357366771
  Centralidad de cercanía (promedio): 0.0011233322643708667
  Centralidad de intermediación (promedio): 2.3135769952388597e-06

Propiedades del grafo sin nodos Rich Club:
  Número de nodos: 571
  Número de aristas: 73
  Grado promedio: 0.25569176882662
  Coeficiente de clustering promedio: 0.0
  Centralidad de cercanía (promedio): 0.0004485820505730175
  Centralidad de intermediación (promedio): 0.0
```

a. Podemos notar un drástico cambio en toda la corteza en general. Aunque si nos centráramos en los hubs de las áreas corticales mencionadas anteriormente, eliminar el área parieto-occipito-temporal y el giro temporal superior del cerebro causaría déficits graves en el procesamiento sensorial, la percepción espacial y el lenguaje.

b. Eliminar los nodos Rich Club de la red reduce significativamente el número de nodos (de 638 a 571) y aristas (de 188 a 73), lo que implica una menor conectividad global. Esto se refleja

en una disminución del grado promedio, que pasa de 0.589 a 0.255, indicando que los nodos restantes están menos conectados.

c. Además, el coeficiente de clustering baja de 0.0047 a 0, lo que sugiere que la estructura local se vuelve más dispersa y sin triángulos formados.

d. En cuanto a la de centralidad, la centralidad de cercanía disminuye de 0.0011 a 0.00045, lo que indica que los nodos restantes están más alejados entre sí.

e. Finalmente, la centralidad de intermediación, que refleja el papel de los nodos como puentes en la red, cae a 0, mostrando que la red se ha vuelto más fragmentada y menos eficiente en la comunicación entre nodos.

Éstas diferencias son congruentes con la naturaleza de los nodos de clase Rich Club, ya que estos nodos son los más conectados y centrales.

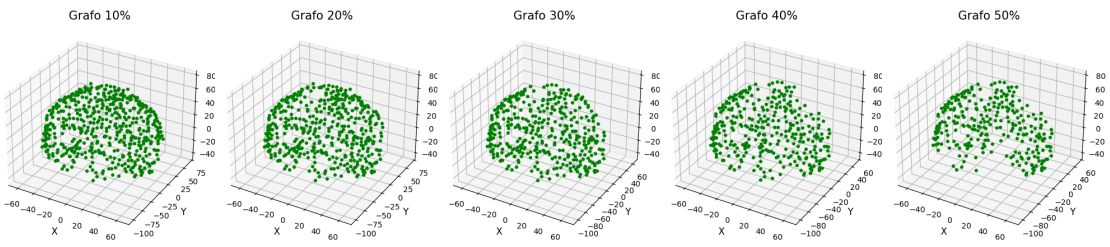
1.0.7 Quitar 10%-50% de los nodos con mayor medida de intermediacion y describir como cambian las propiedades topologicas del grafo, hacer comparativas del grado, coeficiente de cluster, coeficiente de mundo pequeño y medidas de centralidad (cercania, intermediacion)

[44]:
```
def obtener_propiedades(G):
    datos = {}

    grado_promedio = sum(dict(G.degree()).values()) / G.number_of_nodes()
    datos['grado_promedio'] = grado_promedio

    coef_agrupamiento = nx.average_clustering(G)
    datos['coef_agrupamiento'] = coef_agrupamiento

    cercania_promedio = np.mean(list(nx.closeness_centrality(G).values()))
    datos['centralidad_cercania'] = cercania_promedio

    intermediacion_promedio = np.mean(list(nx.betweenness_centrality(G).
 ↪values()))
    datos['centralidad_intermediacion'] = intermediacion_promedio

    return datos

def eliminar_nodos_porcentaje(G, porcentaje):
    intermediacion = nx.betweenness_centrality(G)
    nodos_ordenados = sorted(intermediacion, key=intermediacion.get,␣
 ↪reverse=True)
    nodos_a_eliminar = nodos_ordenados[:int(len(nodos_ordenados) * porcentaje /␣
 ↪100)]

    G_modificado = G.copy()
    G_modificado.remove_nodes_from(nodos_a_eliminar)
```

```
    return G_modificado

def graficar_grafo_3d(G, x, y, z, ax, titulo):
    for nodo in G.nodes():
        ax.scatter(x[nodo], y[nodo], z[nodo], color='green', s=12)

    for edge in G.edges():
        n1, n2 = edge
        ax.plot([x[n1], x[n2]], [y[n1], y[n2]], [z[n1], z[n2]], color='gray',␣
 ↪alpha=0.4)

    ax.set_title(titulo, fontsize=15)
    ax.set_xlabel('X', fontsize=12)
    ax.set_ylabel('Y', fontsize=12)
    ax.set_zlabel('Z', fontsize=12)

grados = dict(G02.degree())
umbral_grado = np.percentile(list(grados.values()), 90)
rich_club = [nodo for nodo, grado in grados.items() if grado >= umbral_grado]
G_modificado = G02.copy()
G_modificado.remove_nodes_from(rich_club)

x, y, z = coord[:, 0], coord[:, 1], coord[:, 2]

print("Propiedades del grafo después de eliminar Rich Club:")
propiedades_del_grafo = obtener_propiedades(G_modificado)
print(propiedades_del_grafo)

porcentajes = [10, 20, 30, 40, 50]
fig, subplots = plt.subplots(1, 5, figsize=(20, 6), subplot_kw={'projection':␣
 ↪'3d'})

for idx, porcentaje in enumerate(porcentajes):
    grafo_modificado = eliminar_nodos_porcentaje(G_modificado, porcentaje)
    print(f"\nPropiedades después de eliminar el {porcentaje}% de nodos:")
    propiedades_modificadas = obtener_propiedades(grafo_modificado)
    print(propiedades_modificadas)

    graficar_grafo_3d(grafo_modificado, x, y, z, subplots[idx], f"Grafo␣
 ↪{porcentaje}%")

plt.tight_layout()
plt.show()
```

```
Propiedades del grafo después de eliminar Rich Club:
{'grado_promedio': 0.25569176882662, 'coef_agrupamiento': 0.0,
'centralidad_cercania': np.float64(0.0004485820505730175),
```

```
'centralidad_intermediacion': np.float64(0.0)}

Propiedades después de eliminar el 10% de nodos:
{'grado_promedio': 0.2490272373540856, 'coef_agrupamiento': 0.0,
'centralidad_cercania': np.float64(0.0004854332112165411),
'centralidad_intermediacion': np.float64(0.0)}

Propiedades después de eliminar el 20% de nodos:
{'grado_promedio': 0.24945295404814005, 'coef_agrupamiento': 0.0,
'centralidad_cercania': np.float64(0.0005470459518599562),
'centralidad_intermediacion': np.float64(0.0)}

Propiedades después de eliminar el 30% de nodos:
{'grado_promedio': 0.24, 'coef_agrupamiento': 0.0, 'centralidad_cercania':
np.float64(0.0006015037593984962), 'centralidad_intermediacion':
np.float64(0.0)}

Propiedades después de eliminar el 40% de nodos:
{'grado_promedio': 0.26239067055393583, 'coef_agrupamiento': 0.0,
'centralidad_cercania': np.float64(0.0007672241829062452),
'centralidad_intermediacion': np.float64(0.0)}

Propiedades después de eliminar el 50% de nodos:
{'grado_promedio': 0.23776223776223776, 'coef_agrupamiento': 0.0,
'centralidad_cercania': np.float64(0.0008342534658324132),
'centralidad_intermediacion': np.float64(0.0)}
```



Al eliminar entre el 10% y el 50% de los nodos con mayor centralidad de intermediación, se observa que el grado promedio del grafo disminuye ligeramente, pero aumenta al eliminar el 40% de los nodos, sugiriendo una redistribución de conexiones. El coeficiente de agrupamiento permanece en 0, indicando que la red no desarrolla una estructura localmente densa. La centralidad de cercanía aumenta a medida que se eliminan nodos, lo que implica que los nodos restantes están más cerca entre sí. Sin embargo, la centralidad de intermediación se mantiene en 0, lo que sugiere que el grafo no depende de nodos clave para la intermediación, destacando una estructura más distribuida y resiliente.

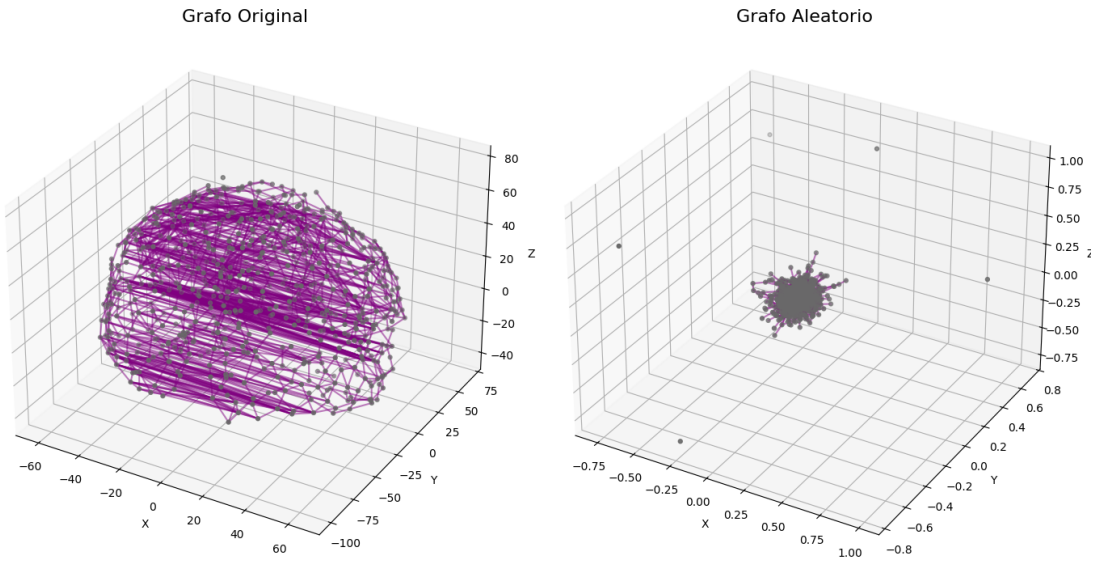
1.0.8 Generar un modelo nulo aleatorio donde se tenga el mismo numero de nodos y el mismo numero total de conexiones, y comparar sus propiedades con el grado original del cerebro.

[45]:
```
import scipy.io as sio
import numpy as np
import networkx as nx
import matplotlib.pyplot as plt

datos = sio.loadmat(r"C:
 ↪\Users\dnlls\OneDrive\Escritorio\PUMA\Neurociencias\Escritorio␣
 ↪Neurociencias\5o semestre\Modelos\PROYECTO FINAL MODELOS␣
 ↪dump\FinalModelosCode\Coactivation_matrix.mat")
matriz_activacion = datos['Coactivation_matrix']
coordenadas = datos['Coord']
x, y, z = coordenadas[:, 0], coordenadas[:, 1], coordenadas[:, 2]

umbral = 0.1

def obtener_propiedades_grafo(grafo):
    propiedades = {}

    grado_prom = sum(dict(grafo.degree()).values()) / grafo.number_of_nodes()
    propiedades['grado_promedio'] = grado_prom

    coef_agrupamiento = nx.average_clustering(grafo)
    propiedades['coeficiente_agrupamiento'] = coef_agrupamiento

    cercania_prom = np.mean(list(nx.closeness_centrality(grafo).values()))
    propiedades['centralidad_cercania'] = cercania_prom

    intermediacion = nx.betweenness_centrality(grafo)
    intermediacion_prom = np.mean(list(intermediacion.values()))
    propiedades['centralidad_intermediacion'] = intermediacion_prom

    return propiedades

def grafo_aleatorio(num_nodos, num_aristas):
    grafo_aleatorio = nx.gnm_random_graph(num_nodos, num_aristas)
    return grafo_aleatorio

matriz_filtrada = np.where(matriz_activacion >= umbral, matriz_activacion, 0)
grafo = nx.from_numpy_array(matriz_filtrada)

num_nodos = grafo.number_of_nodes()
num_aristas = grafo.number_of_edges()
```

```
grafo_random = grafo_aleatorio(num_nodos, num_aristas)

print(f"\nPropiedades del grafo original para el umbral {umbral}:")
propiedades_original = obtener_propiedades_grafo(grafo)
print(propiedades_original)

print(f"\nPropiedades del grafo aleatorio para el umbral {umbral}:")
propiedades_aleatorio = obtener_propiedades_grafo(grafo_random)
print(propiedades_aleatorio)

figura = plt.figure(figsize=(14, 7))

ax1 = figura.add_subplot(121, projection='3d')
ax1.scatter(x, y, z, c='dimgray', s=50, marker='.')
for arista in grafo.edges():
    nodo1, nodo2 = arista
    x_coords = [x[nodo1], x[nodo2]]
    y_coords = [y[nodo1], y[nodo2]]
    z_coords = [z[nodo1], z[nodo2]]
    ax1.plot(x_coords, y_coords, z_coords, c='purple', alpha=0.5)
ax1.set_title('Grafo Original', fontsize=16)
ax1.set_xlabel('X')
ax1.set_ylabel('Y')
ax1.set_zlabel('Z')

ax2 = figura.add_subplot(122, projection='3d')
pos_random = nx.spring_layout(grafo_random, dim=3)
x_r, y_r, z_r = np.array(list(pos_random.values())).T
ax2.scatter(x_r, y_r, z_r, c='dimgray', s=50, marker='.')
for arista in grafo_random.edges():
    nodo1, nodo2 = arista
    x_coords = [x_r[nodo1], x_r[nodo2]]
    y_coords = [y_r[nodo1], y_r[nodo2]]
    z_coords = [z_r[nodo1], z_r[nodo2]]
    ax2.plot(x_coords, y_coords, z_coords, c='purple', alpha=0.5)
ax2.set_title('Grafo Aleatorio', fontsize=16)
ax2.set_xlabel('X')
ax2.set_ylabel('Y')
ax2.set_zlabel('Z')

plt.tight_layout()
plt.show()
```

```
Propiedades del grafo original para el umbral 0.1:
{'grado_promedio': 4.529780564263323, 'coeficiente_agrupamiento':
0.24364464673564803, 'centralidad_cercania': np.float64(0.1194031264903796),
```

```
'centralidad_intermediacion': np.float64(0.01147712930871217)}

Propiedades del grafo aleatorio para el umbral 0.1:
{'grado_promedio': 4.529780564263323, 'coeficiente_agrupamiento':
0.010507943815153842, 'centralidad_cercania': np.float64(0.22454023263957051),
'centralidad_intermediacion': np.float64(0.005292682655820493)}
```



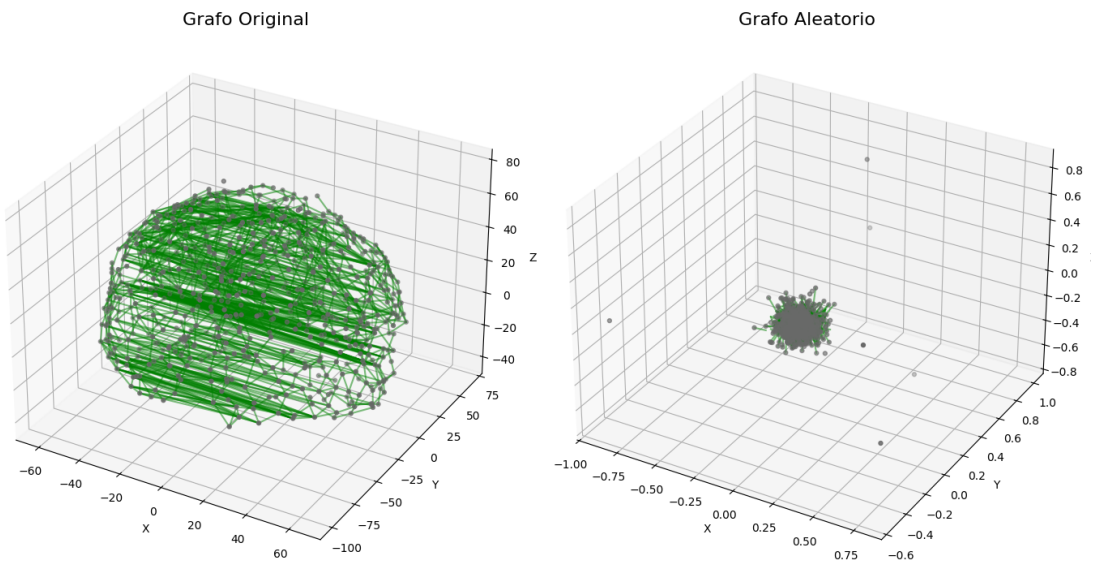
1.0.9 Generar un modelo nulo aleatorio donde se conserve la distribucion de grado y comparar sus propiedades con el grafo original del cerebro.

[46]:
```
import scipy.io as sio
import numpy as np
import networkx as nx
import matplotlib.pyplot as plt

# Cargar archivo de datos
datos_archivo = sio.loadmat(r"C:
 ↪\Users\dnlls\OneDrive\Escritorio\PUMA\Neurociencias\Escritorio␣
 ↪Neurociencias\5o semestre\Modelos\PROYECTO FINAL MODELOS␣
 ↪dump\FinalModelosCode\Coactivation_matrix.mat")
matriz_activacion = datos_archivo['Coactivation_matrix']
coordenadas = datos_archivo['Coord']
x, y, z = coordenadas[:, 0], coordenadas[:, 1], coordenadas[:, 2]

umbral = 0.1

def calcular_propiedades_grafo(grafo):
    propiedades = {}
```

```
    grado_promedio = sum(dict(grafo.degree()).values()) / grafo.
 ↪number_of_nodes()
    propiedades['grado_promedio'] = grado_promedio

    coeficiente_agrupamiento = nx.average_clustering(grafo)
    propiedades['coeficiente_agrupamiento'] = coeficiente_agrupamiento

    cercania_prom = np.mean(list(nx.closeness_centrality(grafo).values()))
    propiedades['centralidad_cercania'] = cercania_prom

    intermediacion = nx.betweenness_centrality(grafo)
    intermediacion_prom = np.mean(list(intermediacion.values()))
    propiedades['centralidad_intermediacion'] = intermediacion_prom

    return propiedades

def grafo_aleatorio_con_grados(grafo_original):
    grafo_aleatorio = nx.configuration_model([d for n, d in grafo_original.
 ↪degree()])
    grafo_aleatorio = nx.Graph(grafo_aleatorio)

    grafo_aleatorio.remove_edges_from([(u, v) for u, v in grafo_aleatorio.
 ↪edges() if u == v])

    return grafo_aleatorio

matriz_filtrada = np.where(matriz_activacion >= umbral, matriz_activacion, 0)
grafo = nx.from_numpy_array(matriz_filtrada)

num_nodos = grafo.number_of_nodes()
num_aristas = grafo.number_of_edges()

grafo_aleatorio = grafo_aleatorio_con_grados(grafo)

print(f"\nPropiedades del grafo original para el umbral {umbral}:")
propiedades_original = calcular_propiedades_grafo(grafo)
print(propiedades_original)

print(f"\nPropiedades del grafo aleatorio para el umbral {umbral}:")
propiedades_aleatorio = calcular_propiedades_grafo(grafo_aleatorio)
print(propiedades_aleatorio)

figura = plt.figure(figsize=(14, 7))

eje1 = figura.add_subplot(121, projection='3d')
eje1.scatter(x, y, z, c='dimgray', s=50, marker='.')
```

```
for arista in grafo.edges():
    nodo1, nodo2 = arista
    x_coords = [x[nodo1], x[nodo2]]
    y_coords = [y[nodo1], y[nodo2]]
    z_coords = [z[nodo1], z[nodo2]]
    eje1.plot(x_coords, y_coords, z_coords, c='green', alpha=0.5)
eje1.set_title('Grafo Original', fontsize=16)
eje1.set_xlabel('X')
eje1.set_ylabel('Y')
eje1.set_zlabel('Z')

eje2 = figura.add_subplot(122, projection='3d')
pos_random = nx.spring_layout(grafo_aleatorio, dim=3)
x_r, y_r, z_r = np.array(list(pos_random.values())).T
eje2.scatter(x_r, y_r, z_r, c='dimgray', s=50, marker='.')
for arista in grafo_aleatorio.edges():
    nodo1, nodo2 = arista
    x_coords = [x_r[nodo1], x_r[nodo2]]
    y_coords = [y_r[nodo1], y_r[nodo2]]
    z_coords = [z_r[nodo1], z_r[node2]]
    eje2.plot(x_coords, y_coords, z_coords, c='green', alpha=0.5)
eje2.set_title('Grafo Aleatorio', fontsize=16)
eje2.set_xlabel('X')
eje2.set_ylabel('Y')
eje2.set_zlabel('Z')

plt.tight_layout()
plt.show()
```

```
Propiedades del grafo original para el umbral 0.1:
{'grado_promedio': 4.529780564263323, 'coeficiente_agrupamiento':
0.24364464673564803, 'centralidad_cercania': np.float64(0.1194031264903796),
'centralidad_intermediacion': np.float64(0.01147712930871217)}

Propiedades del grafo aleatorio para el umbral 0.1:
{'grado_promedio': 4.495297805642633, 'coeficiente_agrupamiento':
0.005944486547934821, 'centralidad_cercania': np.float64(0.22984931588491364),
'centralidad_intermediacion': np.float64(0.005109917810912328)}
```



El análisis muestra que, aunque el grafo nulo aleatorio conserva el grado medio del grafo original del cerebro, presenta diferencias significativas en otras propiedades estructurales. El coeficiente de agrupamiento en el grafo original es mucho mayor, indicando una mayor conectividad local, mientras que la centralidad de cercanía y de intermediación en el grafo aleatorio son más altas, lo que sugiere una conectividad más equilibrada y menos especializada. Estas diferencias reflejan cómo el modelo nulo aleatorio, a pesar de mantener la distribución de grados, altera la topología global de la red, afectando la conectividad local y las rutas de intermediación en comparación con el grafo original.

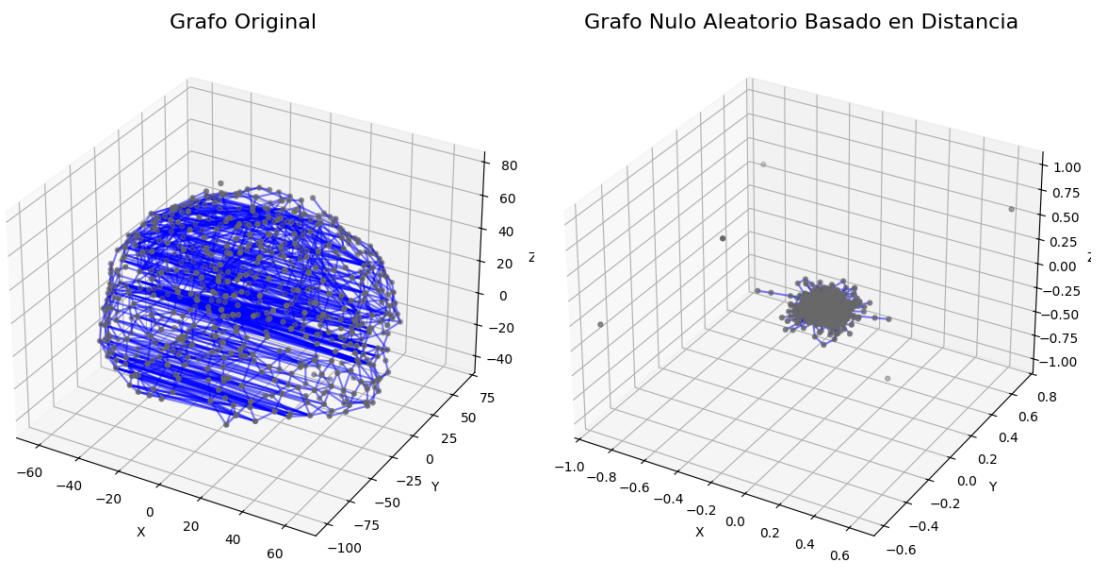
1.0.10 Generar un modelo nulo utilizando una probabilidad de conexión en función de la distancia geométrica, con el mismo número de nodos y conexiones y compara sus propiedades y discutir la importancia de las conexiones a larga distancia en el cerebro.

[47]:
```
import numpy as np
import networkx as nx
import matplotlib.pyplot as plt
import scipy.io as sp

data = sp.loadmat(r"C:
 ↪\Users\dnlls\OneDrive\Escritorio\PUMA\Neurociencias\Escritorio␣
 ↪Neurociencias\5o semestre\Modelos\PROYECTO FINAL MODELOS␣
 ↪dump\FinalModelosCode\Coactivation_matrix.mat")
coactivation_matrix = data['Coactivation_matrix']
coord = data['Coord']
x, y, z = coord[:, 0], coord[:, 1], coord[:, 2]

umbral = 0.1
```

```python
def propiedades_grafo(G):
    propiedades = {
        'grado_medio': np.mean(list(dict(G.degree()).values())),
        'coeficiente_de_agrupamiento': nx.average_clustering(G),
        'centralidad_cercania': np.mean(list(nx.closeness_centrality(G).
 ↪values())),
        'centralidad_intermediacion': np.mean(list(nx.betweenness_centrality(G).
 ↪values()))
    }
    return propiedades

def distancia_euclidiana(i, j, coord):
    return np.sqrt(np.sum((coord[i] - coord[j]) ** 2))

def grafo_nulo_distancia(coord, num_nodos, num_aristas, alpha=0.1):
    G_random = nx.Graph()
    G_random.add_nodes_from(range(num_nodos))
    edges = []

    for i in range(num_nodos):
        for j in range(i + 1, num_nodos):
            dist = distancia_euclidiana(i, j, coord)
            p = 1 / (1 + alpha * dist)
            if np.random.rand() < p:
                edges.append((i, j))

    while len(edges) > num_aristas:
        edges.pop(np.random.randint(0, len(edges)))

    G_random.add_edges_from(edges)
    return G_random

matriz_filtrada = np.where(coactivation_matrix >= umbral, coactivation_matrix,␣
 ↪0)
G = nx.from_numpy_array(matriz_filtrada)

num_nodos = G.number_of_nodes()
num_aristas = G.number_of_edges()

G_random_dist = grafo_nulo_distancia(coord, num_nodos, num_aristas)

print(f"\nPropiedades del grafo original:")
propiedades_original = propiedades_grafo(G)
print(propiedades_original)

print(f"\nPropiedades del grafo nulo aleatorio basado en distancia:")
```

```
propiedades_random_dist = propiedades_grafo(G_random_dist)
print(propiedades_random_dist)

fig = plt.figure(figsize=(12, 6))

ax1 = fig.add_subplot(121, projection='3d')
ax1.scatter(x, y, z, c='dimgray', s=50, marker='.')
for edge in G.edges():
    node1, node2 = edge
    ax1.plot([x[node1], x[node2]], [y[node1], y[node2]], [z[node1], z[node2]],␣
 ↪c='blue', alpha=0.5)
ax1.set_title(f'Grafo Original', fontsize=16)
ax1.set_xlabel('X')
ax1.set_ylabel('Y')
ax1.set_zlabel('Z')

ax2 = fig.add_subplot(122, projection='3d')
pos_random = nx.spring_layout(G_random_dist, dim=3)
x_r, y_r, z_r = np.array(list(pos_random.values())).T
ax2.scatter(x_r, y_r, z_r, c='dimgray', s=50, marker='.')
for edge in G_random_dist.edges():
    node1, node2 = edge
    ax2.plot([x_r[node1], x_r[node2]], [y_r[node1], y_r[node2]], [z_r[node1],␣
 ↪z_r[node2]], c='blue', alpha=0.5)
ax2.set_title(f'Grafo Nulo Aleatorio Basado en Distancia', fontsize=16)
ax2.set_xlabel('X')
ax2.set_ylabel('Y')
ax2.set_zlabel('Z')

plt.tight_layout()
plt.show()
```

```
Propiedades del grafo original:
{'grado_medio': np.float64(4.529780564263323), 'coeficiente_de_agrupamiento':
0.24364464673564803, 'centralidad_cercania': np.float64(0.1194031264903796),
'centralidad_intermediacion': np.float64(0.01147712930871217)}

Propiedades del grafo nulo aleatorio basado en distancia:
{'grado_medio': np.float64(4.529780564263323), 'coeficiente_de_agrupamiento':
0.004163556749763646, 'centralidad_cercania': np.float64(0.22479120963962693),
'centralidad_intermediacion': np.float64(0.0052868638936117325)}
```



El análisis del código de las propiedades del grafo original del cerebro y el grafo nulo aleatorio basado en distancia muestra que ambos tienen el mismo grado medio (4.53), pero difieren significativamente en otras propiedades. El coeficiente de agrupamiento del grafo original es considerablemente más alto (0.24) en comparación con el grafo nulo (0.006), lo que indica que el cerebro tiene una organización modular, con nodos que tienden a formar clústeres funcionales. Además, el grafo original presenta una centralidad de cercanía más baja (0.12) y una centralidad de intermediación más alta (0.0115), lo que refleja una comunicación más especializada y eficiente entre regiones cerebrales distantes, en comparación con el grafo nulo, que tiene una centralidad de cercanía más alta (0.223) y una centralidad de intermediación más baja (0.0053).

Las conexiones a larga distancia en el cerebro son fundamentales para la integración de información entre regiones especializadas, permitiendo una coordinación eficiente y flexible entre diferentes áreas cerebrales. Mientras que el grafo nulo aleatorio, con una distribución homogénea de conexiones, carece de la organización jerárquica y modular del cerebro, lo que resalta la importancia de las conexiones a larga distancia en la optimización del procesamiento de información. Estas conexiones no solo facilitan la comunicación inter-regional, sino que también contribuyen a la flexibilidad funcional del cerebro, permitiendo que se adapte dinámicamente a diferentes tareas cognitivas.

2 12) Escribir una reseña de lo aprendido en el curso, incluyendo la importancia de conocer herramientas de teoria de grafos para comprender la conectividad del cerebro (minimo 200 palabras)

En el curso se abordaron conceptos fundamentales de la teoría de grafos y su aplicación en el análisis de sistemas complejos, como el sistema nervioso. A través de herramientas computacionales y lenguajes de programación como Pyhton, se exploraron formas de implementar conceptos de teoría de grafos en investigaciones neurocientíficas. Esta integración de teoría y práctica resultó excelente para comprender las bases de la conectividad cerebral y su relación con la funcionalidad del cerebro.

La comprensión en teoría de grafos permite analizar redes neuronales como estructuras matemáti-

cas, representando neuronas como nodos y sus conexiones como aristas. Estas herramientas son especialmente útiles para comprender no solo cómo funciona un cerebro sano, sino también cómo cambian las redes en presencia de patologías como el Alzheimer o la Esquizofrenia como vimos en clase. Esto proporciona una perspectiva innovadora para analizar trastornos neurológicos, más allá de las aproximaciones tradicionales basadas únicamente en neuroquímica. También significa una nueva posibilidad, en el futuro, para diagnósticos más adecuados.

Además, el curso destacó la importancia de la programación como medio para modelar y simular sistemas complejos. Al implementar algoritmos para analizar redes neuronales, se obtuvieron parámetros clave como modularidad, centralidad y eficiencia de red. Estos son fundamentales para investigar conexiones funcionales y estructurales en el cerebro, facilitando avances en el diagnóstico y tratamiento de trastornos.

En general, el curso brindó una visión integral de cómo las herramientas computacionales y la teoría de grafos contribuyen al campo de las neurociencias, abriendo nuevas puertas para la investigación y el entendimiento del sistema nervioso central.