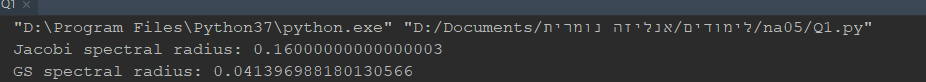
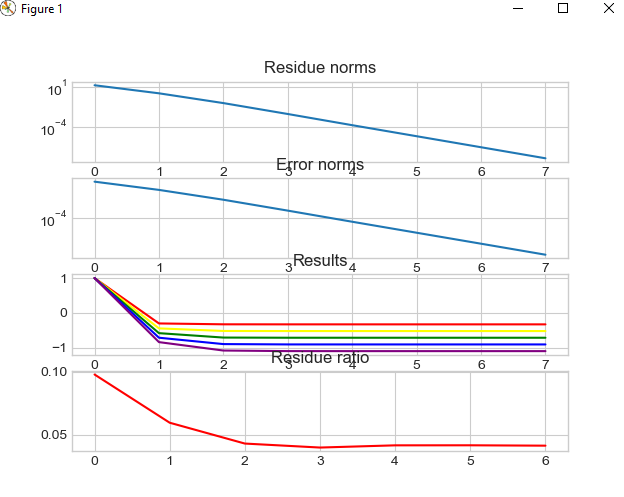
Q1

import numpy as np  
import math  
import matplotlib.pyplot as plt  
plt.style.use('seaborn-whitegrid')  
  
def get\_spectral\_radius(matrix):  
 eigan\_vals = np.linalg.eigvals(matrix)  
 rho = max([math.fabs(val) for val in eigan\_vals])  
  
 return rho  
  
def jacobi\_iteration\_matrix(matrix):  
 d = np.diag(np.diag(matrix))  
 eye = np.identity(len(matrix))  
 da = np.linalg.solve(d, matrix)  
  
 return np.subtract(eye, da)  
  
  
def GS\_iteration\_matrix(matrix):  
 l = np.tril(matrix)  
 eye = np.identity(len(matrix))  
 la = np.linalg.solve(l, matrix)  
  
 return np.subtract(eye, la)  
  
def lud\_factorization(matrix):  
 l = np.zeros(matrix.shape)  
 u = np.zeros(matrix.shape)  
 d = np.zeros(matrix.shape)  
  
 for rowIndex, row in enumerate(matrix):  
 for colIndex, col in enumerate(row):  
 if rowIndex < colIndex:  
 l[rowIndex][colIndex] = matrix[rowIndex][colIndex]  
 elif rowIndex == colIndex:  
 d[rowIndex][colIndex] = matrix[rowIndex][colIndex]  
 else:  
 u[rowIndex][colIndex] = matrix[rowIndex][colIndex]  
  
 return l, u, d  
  
  
def GS\_step (a\_matrix, b\_vector, previous\_step, ld\_inv):  
 ax\_prev = np.dot(a\_matrix, previous\_step)  
 b\_minus\_ax = np.subtract(b\_vector, ax\_prev)  
  
 return np.add(previous\_step, np.dot(ld\_inv, b\_minus\_ax))  
  
  
  
def l2\_norm(vector):  
 return math.sqrt(sum([math.pow(item, 2) for item in vector]))  
  
  
def get\_residue\_norm(a\_matrix, b\_vector, step\_result):  
 ax = np.dot(a\_matrix, step\_result)  
 ax\_minus\_b = np.subtract(ax, b\_vector)  
  
 return l2\_norm(ax\_minus\_b)  
  
  
def GS (a\_matrix, b\_vector, initial\_guess, max\_residue):  
 l, u, d = lud\_factorization(a\_matrix)  
 ld\_inv = np.linalg.inv(np.add(l, d))  
  
 step\_results = [initial\_guess]  
 residues = [get\_residue\_norm(a\_matrix, b\_vector, initial\_guess)]  
  
 while residues[-1] > max\_residue:  
 nextIteration = GS\_step(a\_matrix, b\_vector, step\_results[-1], ld\_inv)  
  
 step\_results.append(nextIteration)  
 residues.append(get\_residue\_norm(a\_matrix, b\_vector, nextIteration))  
  
 return step\_results, residues  
  
  
def get\_error\_norms(a\_matrix, b\_vector, results\_steps):  
 actual\_results = np.linalg.solve(a\_matrix, b\_vector)  
 error\_values = [np.subtract(result\_step, actual\_results)  
 for result\_step in results\_steps]  
  
 return [l2\_norm(error) for error in error\_values]  
  
def residue\_ratio(residues):  
 ratios = []  
 for i, val in enumerate(residues[:-1]):  
 ratios.append(residues[i + 1] / residues[i])  
  
 return ratios  
  
  
  
def plot(step\_results, residues, errors):  
 f, axarr = plt.subplots(4, 1)  
  
 axarr[0].set\_title("Residue norms")  
 axarr[0].semilogy(residues)  
  
 axarr[1].set\_title("Error norms")  
 axarr[1].semilogy(errors)  
  
 axarr[2].set\_title("Results")  
 axarr[2].plot([tuple[0] for tuple in step\_results], color='red', label="x1")  
 axarr[2].plot([tuple[1] for tuple in step\_results], color='yellow', label="x2")  
 axarr[2].plot([tuple[2] for tuple in step\_results], color='green', label="x3")  
 axarr[2].plot([tuple[3] for tuple in step\_results], color='blue', label="x4")  
 axarr[2].plot([tuple[4] for tuple in step\_results], color='purple', label="x5")  
  
 axarr[3].set\_title("Residue ratio")  
 axarr[3].plot(residue\_ratio(residues), color='red')  
  
 plt.show()  
  
  
  
a = np.array([[-5, 0.2, 0.2, 0.2, 0.2],  
 [0.2, -5, 0.2, 0.2, 0.2],  
 [0.2, 0.2, -5, 0.2, 0.2],  
 [0.2, 0.2, 0.2, -5, 0.2],  
 [0.2, 0.2, 0.2, 0.2, -5]  
 ])  
print("Jacobi spectral radius: " + str(get\_spectral\_radius(jacobi\_iteration\_matrix(a))))  
print("GS spectral radius: " + str(get\_spectral\_radius(GS\_iteration\_matrix(a))))  
  
b = np.array([1, 2, 3, 4, 5])  
initial\_guess = np.array([1, 1, 1, 1, 1])  
  
results, residues = GS(a, b, initial\_guess, 0.0000001)  
errors = get\_error\_norms(a, b, results)  
plot(results, residues, errors)





ד. לפי הגרף ניתו לראות כי יחס ה

שארית מתאפס.

Q2

נפתח טורי טיילור בנקודות הנתונות:

נחפש צירוף לינארי שיאפס את כל הנגזרות מלבד הנגזרת השניה (הנתונה בנקודות). נסמן מקדמים:

נחבר:  
נדרוש צירוף לינארי שיאפס (באגף שמאל) את כל הנגזרות מלבד הנגזרת השניה. נקבל את המערכת הבאה:  
נבחר כלשהו, כמו . נפתור את המערכת:  
*נציב בחזרה במשוואה:*

לא ניתן להגיע לדיוק טוב יותר – תוספת מחובר בתור הטיילור היה מוסיף עוד משוואה אותה הצירוף הלינארי היה נדרש לאפס, אך לא מוסיף עוד משתנה מאחר ואין עוד נקודות. לכן היינו מקבלים מערכת של ארבע משוואות עם יותר מארבעה נעלמים, אותה לא נוכל לפתור.