שם: מוחמד דגש
ת"ז: 314811290

שאלה 1: קטע קוד:

```
from scipy.special import factorial
#Question 1a, auxiliary function
def f(n):
    return n**n/factorial(n)
#Question 1a
def question1a():
    k = 1001
    ns = list(range(1, k+1))
    for n in ns:
        print('(', n, ',', f(n), ')')
#Question 1b, auxiliary function
def newf(n):
    newf_n = 1
    for k in reversed(range(1, n)):
        newf_n *= (n/k)
    return newf_n
#Question 1b
def question1b():
    k = 1001
    ns = list(range(1, k+1))
    for n in ns:
        print('(', n, ',', newf(n), ')')
#Question 1b, auxiliary for log2 computations
def log_factorial(n):
    import math
    log_sum = 0
    for i in range(1, n+1):
        log_sum += math.log2(i)
    return log_sum
if __name__ == '__main__':
    part_a = 0
    if part_a:
        question1a()
    else:
        question1b()
```

סעיף א':

n^n מסוג int , הפונקציה שלנו f מחזירה float .

במהלך הדפסה השגיאה מתקבלת ב-144

"Overflow Error : int too large to convert to float"

הסיבה לשגיאה: בעת החישוב נדרש לחלק int n^n שהוא ענק בשלב מסוים , ב- n! float .

לצורך כך נדרש להמיר את n^n ל- float .

כשממירים int ל- float , נדרש לייצג את המספר:

$(1+f)\ 2^{c-1023}\ (-1)^s$, c < 2047 , ולכן אם ה- int גדול מדי ההמרה לא מתאפשרת ומקבלים שגיאה.

אם נדייק: $144 \log_2(144) = 1032 + \varepsilon_1$

$$log_2(144^{144}) = 144 \log_2(144) = 1032 + \varepsilon_1$$

$0 < \varepsilon_1 < 1$

$$log_2(143^{143}) = 143 \log_2(143) = 1024 + \varepsilon_2$$

$0 < \varepsilon_2 < 1$

ולכן ברור שההמרה הראשונה שלא מתאפשרת היא עבור *n=144* , כי המעריך של 2 נדרש להיות בין 1023- ל 1024 .

סעיף ב':

אם f מקבלת *int* , היא תחזיר *float* , במהלך ההדפסה הפעם אין שגיאות, והערכים מודפסים עד 713 החל מ- 714 הערך שמודפס הוא *int* , וזה קורה שכן:

$$log_2\left(\frac{714^{714}}{714!}\right) = 714\log_2(714) - \log_2(714!)$$

$$714\log2(714) = 6768 + \varepsilon_1$$

$0 < \varepsilon_1 < 1$

$$\log_2(714!) = \sum_{k=0}^{714}\log_2(k) = [\text{צירפתי פונקציה}] = 5744 + \varepsilon_2$$

$0 < \varepsilon_2 < \varepsilon_1 < 1$

לכן:

$$log_2\left(\frac{714^{714}}{714!}\right) = 6768 - 5744 = 1024 + \varepsilon_3$$

$0 < \varepsilon_3 < 1$

באופן דומה נחשב :

$$log_2\left(\frac{713^{713}}{713!}\right) = 1022 + \varepsilon_4$$

$0 < \varepsilon_4 < 1$

ולכן כשמציגים מספר כזה כי $(-1)^s\, 2^{c-1023}\, (1+f)$ מקבלים שעבור 713 אין בעיה כי המעריך של 2 בטווח 1023- עד 1024 , עבור 714 אנחנו חורגים.

סעיף ג':

ההבדל הוא ממתי מתבצעת ההמרה מ-int ל- float . (ההמרה נדרשת כשמחלקים int ב- float).

בשיטה הראשונה ההמרה מתבצעת על כל המספר $(\frac{n^n}{n!})$ שהחל מ-n מסוים נעשה ענק(כי $\lim_{n\to\infty}(\frac{n^n}{n!}) = inf$) .

ולכן עצם ההמרה היא בעייתית החל משלב מסוים והקוד מפסיק לרוץ.

בשיטה השנייה ההמרה מתבצעת בכל שלב על מספרים קטנים k,n

$(1 \le k \le n)$ ומשם כבר הפעולות מבוצעות ב-float-ים.

לכן אין כאן שגיאת המרה.

מה שכן קורה הוא שבאיזשהו שלב המספר $(\frac{n^n}{n!})$ חוצה את הטווח המותר לאחסון בשיטת הדיוק הכפול $(-1)^s\ 2^{c-1023}\ (1+f)$ ולכן מאותו שלב ואילך מוגדר כ-inf .

שאלה 2 - קטע קוד:

```
import random
import numpy as np
from fixed_pt import fixed_pt

def bisection(f, a, b, epsilon, it):

    c = (a + b) / 2
    f_c = f(c)
    err = np.abs(a - b) / 2

    if err < epsilon or f_c == 0:
        return c, it, err
    it += 1
    if f_c > 0:
        return bisection(f, c, b, epsilon, it)
    else:
        return bisection(f, a, c, epsilon, it)

def quest_2c(f, a, b):

    epsilon = 0.00001

    x, n_it, err = bisection(f, a, b, epsilon, 1)

    return x, n_it

def find_root_using_fxd_pt(f, x_0, epsilon):

    g = lambda x: x + f(x)
    x, n_it, err = fixed_pt(g, x_0, epsilon, 1)

    return x, n_it, err

def quest_2e(f, a, b):

    epsilon = 0.00001

    x_0 = random.uniform(a, b)
    x, n_it, err = find_root_using_fxd_pt(f, x_0, epsilon)

    return n_it, x

if __name__ == '__main__':
```

```
f = lambda x: np.exp(-2 * x) + np.sin(x)

a = np.pi
b = np.pi * (3 / 2)

n_it_bisection, x_bisection = quest_2c(f, a, b)
n_it_fixed, x_fxd_pt = quest_2e(f, a, b)
```

```
import numpy as np

def fixed_pt(f, x_curr, epsilon, it, sclr=1):

    x_next = f(x_curr)
    if sclr:
        err = np.abs(x_next - x_curr)
    else:
        err = np.linalg.norm(x_next - x_curr)
    if err < epsilon:
        return x_curr, it, err
    it += 1
    return fixed_pt(f, x_next, epsilon, it, sclr=sclr)
```

2) א.

$$f(x) = e^{-2x} + \sin x$$

מתקיים: $e^{-2x} < 1$ לכל $x>0$, $\sin x \geq -1$ לכל x , $e^{-2x} > 1$ לכל $x<0$, $e^{-2x} > 0$ לכל x

לכן:
$$\begin{cases} f(x) > 1-1 = 0, & x<0 \text{ לכל} \\ f(\pi) = e^{-2\pi} + 0 > 0 \\ f\left(\frac{3\pi}{2}\right) = e^{-3\pi} - 1 < 0 \end{cases}$$

לכן בהכרח השורש הקטן ביותר של f נמצא בקטע $\left(\pi, \frac{3\pi}{2}\right)$ לפי משפט ערך הביניים.

בנוסף,
$$f'(x) = \cos x - 2e^{-2x}$$

בקטע $x \in \left(\pi, \frac{3\pi}{2}\right)$:
$$-1 = \cos\pi < \cos x < \cos\left(\frac{3\pi}{2}\right) = 0$$
$$2e^{-2x} > 0$$

ולכן $f' < 0$ בקטע הנ"ל ולכן השורש שקיים בקטע <u>יחיד</u> ממשפט רול.

לכן איתחול טוב לשיטת החצייה יהיה $a_0 = \pi, \quad b_0 = \frac{3\pi}{2}$.

ב. (2)

$$h(x) = f(x) + \varepsilon x, \quad |\varepsilon| = O(10^{-2})$$

נשים לב שמתקיים

$$f(\pi) = e^{-2\pi} < e^{-2\cdot 3} < 2.5^{-6} = \frac{1}{6.25^3} < \frac{1}{6^3} < \frac{1}{200}$$

נסמן ב- a_k את הקצוות לאורך ריצת האלגו' עבורם $f(a_k)>0$

לכן

$$0 < f(a_k) < f(\pi) \leq 0.5 \times 10^{-2} = O(10^{-2})$$

f יורדת

$$\varepsilon \cdot a_k > \pi \cdot \varepsilon_0 = O(10^{-2})$$

כלומר ערכי הפונקציה הם מסדר גודל דומה לערכי הרעש, ומהר מאוד ערכי הרעש יהיו גדולים בהרבה מערכי הפונקציה (ככל ש- $f(a_k)$ תקטן)

(חישוב דומה גם עבור $f(b_k)$)

במילים אחרות, לא ניתן יהיה לסמוך על ערכי $h(a_k)$, $h(b_k)$

ובפרט לא על הסימן שלהם מה שמשבש את ריצת האלגו' ויפר את ההתכנסות לשורש של f ב- $(\pi, \frac{3\pi}{2})$

למשל במצב שבו נקבל ש- $f(a_k), f(b_k)$ שווי סימן אז האלגוריתם נתקע.

(2)ג. התוכנה מצורפת. זה לקח $n=18$ איטרציות.

התאוריה אומרת ש:

$$n \geq \log_2 \frac{b-a}{\varepsilon} = \log_2\left(\frac{\pi/2}{10^{-5}}\right) = \log_2\left(\frac{\pi}{2}\cdot 10^{5}\right)$$

$$= \underbrace{\log_2\left(\frac{\pi}{2}\right)}_{\text{מס' בין 0 ל-1}} + 5\log_2(10) \approx 17.26$$

כלומר זה מסתדר עם התאוריה.

השורש המקורב שהתקבל הוא $x \approx 3.1434562$

3. (2)

$$g(x) = x+f(x) = x+e^{-2x}+\sin x$$

נרצה להראות שניתן להשתמש בשיטת נק' השבת בקטע $I=[\pi,\frac{3\pi}{2}]$. לשם כך מספיק שנראה:

(1) $g(I)\subseteq I$ (2) $|g'(x)|\le k<1$ ב-I.

לפי המשפט זה יספיק על מנת שמכל ניחוש $x_0\in I$ התחלתי תהיה התכנסות לנק' השבת.

נראה את (1). לשם כך מספיק להראות:

(*) $g(x)\ge\pi$ לכל $x\ge\pi$.

(**) $g(x)\le\frac{3\pi}{2}$ לכל $x^*\le x\le\frac{3\pi}{2}$, כאשר x^* השורש היחיד של f בקטע I שאת קיומו הוכחנו.

(***) $g(x)\le\frac{3\pi}{2}$ לכל $\pi\le x\le x^*$

הוכחת (*):

$$g(x)-\pi = x-\pi+e^{-2x}+\sin x = x-\pi-\sin(x-\pi)+e^{-2x}$$

$$\boxed{\begin{gathered}\sin t=\sin(\pi-t)\\ \sin t=-\sin(-t)\end{gathered}}$$

כידוע, לכל $t\ge0$ מתקיים $t\ge\sin t$

(כי הפונק' מתלכדות ב-$t=0$ ומשם שיפוע של t ≥ שיפוע של $\sin t$)

ולכן לכל $x\ge\pi$, $x-\pi-\sin(x-\pi)+e^{-2x}\ge e^{-2x}>0$.

הוכחת (**): לכל $x^*\le x\le\frac{3\pi}{2}$, $f(x)\le0$, ולכן

$$g(x)=x+f(x)\le x\le\frac{3\pi}{2}$$

הוכחת (***): f יורדת ב-I (הראינו) ולכן אם $\pi\le x\le x^*$ אז

$$f(x)\le f(\pi)=e^{-2\pi}<e^{-6}<2.5^{-6}=6.25^{-3}<6^{-3}<\frac{1}{200}$$

ולכן $g(x)=x+f(x)\le x^*+\frac{1}{200}$

מתקיים:

$$f\left(\frac{5\pi}{4}\right)=-\frac{1}{\sqrt2}+e^{-5\pi/2}<0$$

ולכן מיחידות x^*, $x^*<\frac{5\pi}{4}$ ולכן

$$g(x)\le x^*+\frac{1}{200}<\frac{5\pi}{4}+\frac{1}{200}<\frac{3\pi}{2}$$ כנדרש.

נראה את (2).

$$g'(x)=1-2e^{-2x}+\cos x$$

$\cos x$ עולה ב-I, e^{-2x} יורדת תמיד, לכן g' עולה ב-I. לכן בהכרח $\max_{x\in I}|g'(x)|$ מתקבל באחד הקצוות (שהם בעלי סימן הפוך זה לזה), ולכן

$$|g'(x)|\le\max_{x\in I}|g'(x)|=\max\left\{|g'(\pi)|,\left|g'\left(\tfrac{3\pi}{2}\right)\right|\right\}=\underbrace{\max\{-2e^{-2\pi},1-2e^{-3\pi}\}}<1$$ כנדרש.

(2) ה. הקוד מצורף. בשיטת החצייה זה לקח 18 איטרציות.
בשיטת נק' השבת זה לקח 3-4 איטרציות,
כתלות בניחוש ההתחלתי ב- $[\pi, \frac{3\pi}{2}]$
(למשל עבור $x_0 \approx 3.6199$ זה לקח 4 איטרציות).

זה לא מפתיע. נבין את קבועי ההתכנסות.
בשיטת החצייה ההתכנסות היא לינארית עם קבוע התכנסות $\lambda = \frac{1}{2}$ במממוצע (כי כל פעם חוצים את הקטע).

בשיטת נק' השבת, נרצה להראות שההתכנסות היא לינארית, עם קבוע התכנסות $\lambda << \frac{1}{2}$. נסמן ב-r את השורש היחיד של f בקטע $[\pi, \frac{3\pi}{2}]$ שקיומו הוכח.

$$g(r) = r, \quad g'(r) = 1 + \cos r - 2e^{-2r} = 1 + \cos r + 2\sin r$$

$e^{-2r} = -\sin r$
כי $f(r) = 0$

$$|g'(r)| = |1 + \cos r + 2\sin r| = \left|1 - \sqrt{1 - e^{-2r}} - 2e^{-2r}\right|$$

$\cos^2 + \sin^2 = 1$
$\cos < 0$
ב- $[\pi, \frac{3\pi}{2}]$

$$= \left|\frac{\left(1 - \sqrt{1 - e^{-2r}}\right)\left(1 + \sqrt{1 - e^{-2r}}\right)}{1 + \sqrt{1 - e^{-2r}}} - 2e^{-2r}\right|$$

הרחבה בצמוד

$$= \left|\frac{e^{-2r}}{1 + \sqrt{1 - e^{-2r}}} - 2e^{-2r}\right|$$

$$= e^{-2r}\left|\frac{1}{1 + \sqrt{1 - e^{-2r}}} - 2\right| \leq e^{-2r}(1 + 2) = 3e^{-2r}$$

אי"ש המשולש

$$< 3e^{-2\pi} < \frac{3}{200} << \frac{1}{2}$$

הראינו: $e^{-2\pi} < \frac{1}{200}$

(2) ה. (המשך) כלומר הראינו $\lambda = |g'(r)| << \frac{1}{2}$.

מכאן נובע: אם $g'(r) = 0$ אז ההתכנסות בשיטה זו ריבועית לפחות, ולכן הרבה יותר מהירה משיטת החצייה.
אם $g'(r) \neq 0$, אז מתקיימים תנאי המשפט שממנו נובע שההתכנסות בשיטה זו לינארית עם קבוע התכנסות $\frac{1}{2} >> \lambda = |g'(r)|$

כך או כך נקבל שבשיטת נק' השבת ההתכנסות אכן מהירה יותר משיטת החצייה.

נספח ל- (2)(ב): למעשה, ההתכנסות אכן לינארית (אלא אם ג' שלה לא קריטי לתרגיל). נראה זאת ע"י כך שנוכיח: $g'(r)<0$:

לבחירתך אם להכניס

$$g(r) = 1+\cos r + 2\sin r$$

הסברנו מדוע:

נסמן $h(u) = \sin u + \cos u$

$h'(u) = \cos u - \sin u$

$h''(u) = -\sin u - \cos u$

נתבונן ב- h בקטע $[0,\frac{\pi}{2}]$. $h'(u)=0 \iff u=\frac{\pi}{4}$

ומתקיים $h''(\frac{\pi}{4}) = -\frac{1}{\sqrt{2}}-\frac{1}{\sqrt{2}}<0$ ולכן $u=\frac{\pi}{4}$ נק' מקסימום.

יחידה. לכן נק' המינימום הקצוות: $h(0)=h(\frac{\pi}{2})=1$.

ולכן אם $0<u<\frac{\pi}{2}$ אז $h(u)>1$.

לכן עבור $r\in(\pi,\frac{3\pi}{2})$ (השורש בקטע שלנו),

$$1 < h(r-\pi) = \sin(r-\pi)+\cos(r-\pi) = -\sin r-\cos r$$

$$\boxed{\begin{aligned}\sin t &= -\sin(-t)\\ \cos t &= \cos(-t)\\ \sin t &= \sin(\pi-t)\\ \cos t &= -\cos(\pi-t)\end{aligned}}$$

ולכן $\sin r+\cos r< -1$

בנוסף

לכן $g(r) = 1+(\sin r+\cos r)+\sin r < 1-1+0 = 0$

$\sin<0$ ב- $(\pi,\frac{3\pi}{2})$

סיימנו.

שאלה 3- קטע קוד:

```
import matplotlib.pyplot as plt
import numpy as np
from scipy.linalg import lu

def quest_3a_test():

    A = np.array([[6, 2, 2], [4, -6, 1], [2, -3, 1]])

    p, l, u = lu(A)
    L_mine, U_mine = quest_3a_LU_decomposition(A)
    assert(np.all(l @ u == A))
    assert(np.all(L_mine @ U_mine == A))
    assert(np.all(l == L_mine))
    assert(np.all(u == U_mine))

def quest_3a_LU_decomposition(A):

    n = A.shape[0]
    U = A.astype(np.float).copy()
    L = np.eye(n)

    for col in range(n-1):
        for row in range(col+1, n):
            m_row_col = U[row, col] / U[col, col]
            U[row] = U[row] - m_row_col * U[col]
            L[row, col] = m_row_col

    return L, U


def substitute_U(U, y):

    n = U.shape[0]
    x = np.zeros_like(y)

    for i in reversed(range(n)):
        prev_x = x[i+1:]
        if not prev_x.size:
            prev_dot_prod = 0
        else:
            prev_U = U[i, i + 1:]
            prev_dot_prod = np.dot(prev_x, prev_U)
        divide_coeff = U[i, i]
        x[i] = (y[i] - prev_dot_prod) / divide_coeff
```

```
    return x

def substitute_L(L, b):

    n = L.shape[0]
    y = np.zeros_like(b)

    for i in range(n):
        prev_y = y[:i]
        if not prev_y.size:
            prev_dot_prod = 0
        else:
            prev_L = L[i, :i]
            prev_dot_prod = np.dot(prev_y, prev_L)
        y[i] = b[i] - prev_dot_prod

    return y

def vec_inf_norm(v):
    return np.max(np.abs(v))

def relative_err_inf_norm(x, x_c):

    inf_norm_diff = vec_inf_norm(x - x_c)
    inf_norm_x = vec_inf_norm(x)

    return inf_norm_diff / inf_norm_x

def mat_inf_norm(A):

    abs_A = np.abs(A)
    col_sum = np.sum(abs_A, axis=1)
    inf_norm = np.max(col_sum)

    return inf_norm

def cond(A):

    inv_A = np.linalg.inv(A)
    inf_norm_A = mat_inf_norm(A)
    inf_norm_inv_A = mat_inf_norm(inv_A)
    cond_num = inf_norm_A * inf_norm_inv_A

    return cond_num
```

```
def norm_inf_xc_bound(A, E):
    relative_E_A = mat_inf_norm(E) / vec_inf_norm(A)
    return cond(A) * relative_E_A

def quest_3b(As, ns, xs, bs, dbg_plot=0, quest=None):

    inf_norms_error, x_cs, Es = zip(*[quest3b_per_n(A, x, b)
for A, x, b in zip(As, xs, bs)])

    if dbg_plot:
        ttl = 'Quest. 3b: Inf Norms of Errors'
        if quest == 'd':
            ttl = 'Quest. 3d: Inf Norms of Errors'
        plot_with_labels(ns, inf_norms_error, ttl)

    return inf_norms_error, x_cs, Es

def plot_with_labels(ns, values, title):

    fig = plt.figure()
    ax = fig.add_subplot(111)
    plt.plot(ns, values)
    plt.plot(ns, values, '*r')
    [ax.annotate(val, xy=(n, val), xytext=(-7, 7),
textcoords='offset points') for n, val in zip(ns, values)]
    plt.title(title)
    plt.pause(10)

def quest_3c(As, ns, x_cs, xs, bs, Es, dbg_plot=0, quest=None):

    cond_nums = [cond(A) for A in As]
    rs = [A @ x_c - b for A, x_c, b in zip(As, x_cs, bs)]
    inf_norm_bounds = [norm_inf_xc_bound(A, E) for A, E in
zip(As, Es)]

    if dbg_plot:
        ttl = "Quest. 3c: Condition Numbers"
        if quest == 'd':
            ttl = "Quest. 3d: Condition Numbers"
        plot_with_labels(ns, cond_nums, ttl)
        ttl = 'Quest. 3c: Inf Norm Error Bounds'
        if quest == 'd':
            ttl = 'Quest. 3d: Inf Norm Error Bounds'
        plot_with_labels(ns, inf_norm_bounds, ttl)
```

```
    return inf_norm_bounds

def quest_3d(ns):
    for n in ns:
        quest_3_per_n(A)

def create_A(n):

    A = np.zeros((n, n))
    for i in range(n):
        for j in range(n):
            A[i, j] = 1 + abs(i - j)

    return A

def create_C(n):

    C = np.zeros((n, n))
    for i in range(n):
        for j in range(n):
            C[i, j] = np.sqrt((i - j) ** 2 + n / 10)

    return C

def quest3b_per_n(A, x, b):

    L, U = quest_3a_LU_decomposition(A)
    y_c = substitute_L(L, b)
    x_c = substitute_U(U, y_c)
    E = L @ U - A
    rel_inf_norm = relative_err_inf_norm(x, x_c)

    return rel_inf_norm, x_c, E

def question_3bc_steps(create_mat, ns, xs, dbg_plot=0,
quest=None):

    As = [create_mat(n) for n in ns]
    bs = [A @ x for A, x in zip(As, xs)]
    inf_norms_error, x_cs, Es = quest_3b(As, ns, xs, bs,
dbg_plot=dbg_plot, quest=quest)
    inf_norm_bounds = quest_3c(As, ns, x_cs, xs, bs, Es,
dbg_plot=dbg_plot, quest=quest)
```

```
    if dbg_plot:
        plt.figure()
        plt.plot(ns, inf_norms_error, 'r', label='Actual Inf
Norm Relative errors')
        plt.plot(ns, inf_norm_bounds, 'b', label='Theoretical
Bound on Inf Norm Relative errors')
        plt.legend()
        plt.pause(10)

def question_bc_combined(ns, xs, dbg_plot=0):
    question_3bc_steps(lambda n: create_A(n), ns, xs,
dbg_plot=dbg_plot)

def question_d(ns, xs, dbg_plot=0):
    question_3bc_steps(lambda n: create_C(n), ns, xs,
dbg_plot=dbg_plot, quest='d')

if __name__ == '__main__':

    quest_3a_test()

    ns = [100, 200, 300, 400, 500]
    xs = [np.ones((n, )) for n in ns]

    use_A = 0
    if use_A:
        question_bc_combined(ns, xs, dbg_plot=1)
    else:
        question_d(ns, xs, dbg_plot=1)
```

(3) א. הקוד מצורף. (הפונקציה quest_3a_LU_decomposition)

סה"כ 4 גרפים

(3) ב. הקוד מצורף, וגם צילומים של הפלט של נורמות אינסוף של השגיאה היחסית: $\frac{\|x-x_c\|_\infty}{\|x\|_\infty}$

(3) ג. הקוד מצורף, וגם צילומים של הפלט של cond(A): $\|A\|_\infty \cdot \|A^{-1}\|_\infty$

ושל החסמים על נורמות אינסוף של השגיאה היחסית:

$cond(A) \cdot \frac{\|E\|_\infty}{\|A\|_\infty}$, $E = LU - A$

(כי x_c הוא למעשה פתרון של $LUx=b$)

מצורף גם גרף השוואתי שמוכיח:

$$\frac{\|x-x_c\|}{\|x\|_\infty} \le cond(A)\frac{\|E\|_\infty}{\|A\|_\infty},$$

3 גרפים

(3) ד. הקוד מצורף, ואיתו עוד 3 גרפים:

אחד לנורמת אינסוף של השגיאה

אחד ל- cond(A)

וארוך לחסמים על נורמת אינסוף של השגיאה היחסית.

ההבדלים בין התוצאות: השגיאות עבור C גבוהות בכמה סדרי גודל מאשר השגיאות עבור A.

כמו שבאופן דומה $cond(C) >> cond(A)$

(פי 1000 בקירוב)

כלומר C היא מטריצה ill-conditioned ולכן נצפה לכך

שהשגיאה בפירוק LU תועצם יותר עבור C

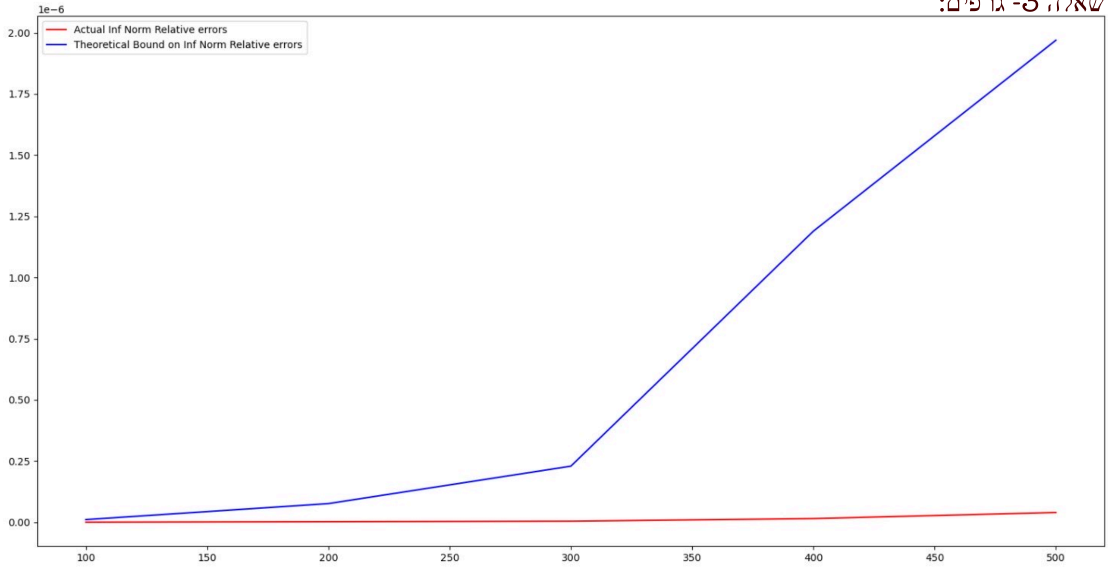
ותגרור שגיאה יחסית $\frac{\|x-x_c\|_\infty}{\|x\|_\infty}$ גדולה יותר.

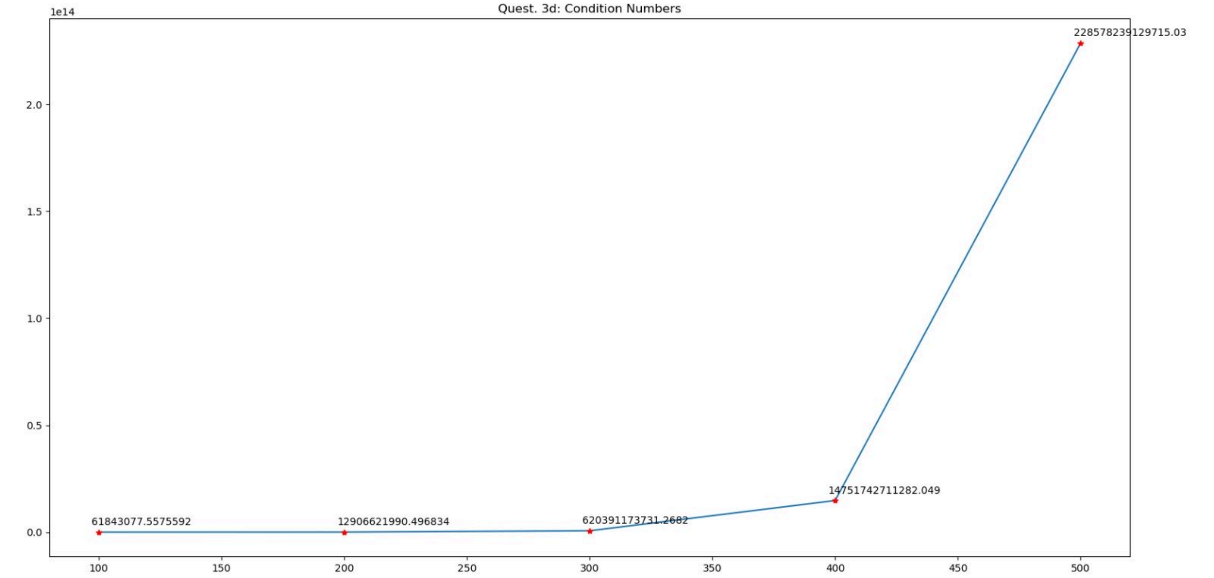
אגב, שגיאת פירוק LU עצמה $\frac{\|LU-A\|_\infty}{\|A\|_\infty}$

אפילו מעט יותר גבוהה עבור A מאשר עבור C (ובכל מקרה קטנה מאוד).

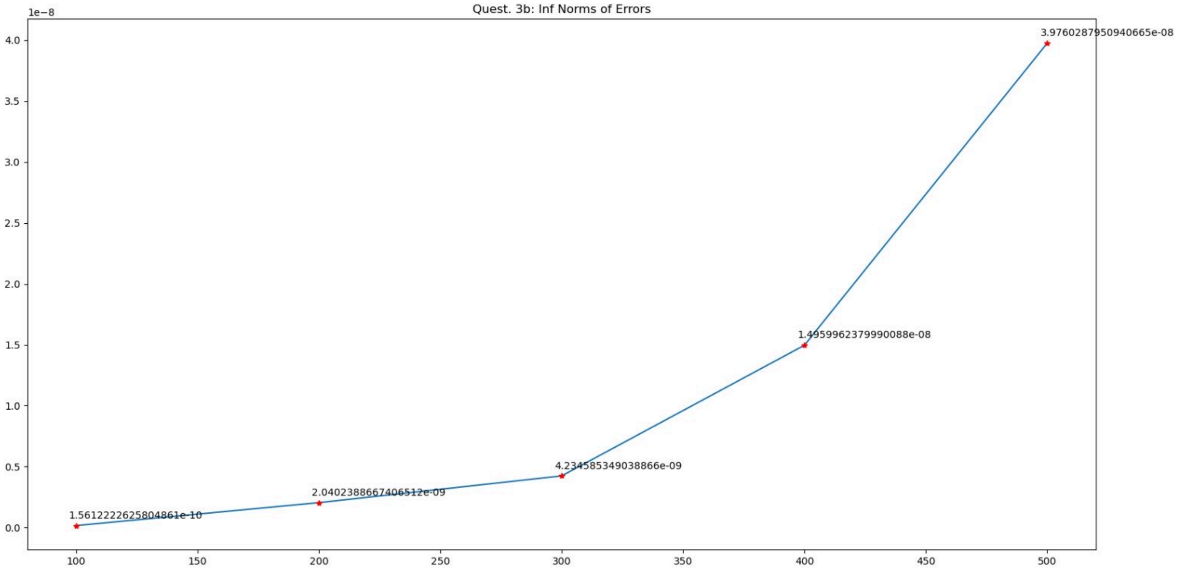
כלומר הסיבה היא באמת ה-condition number

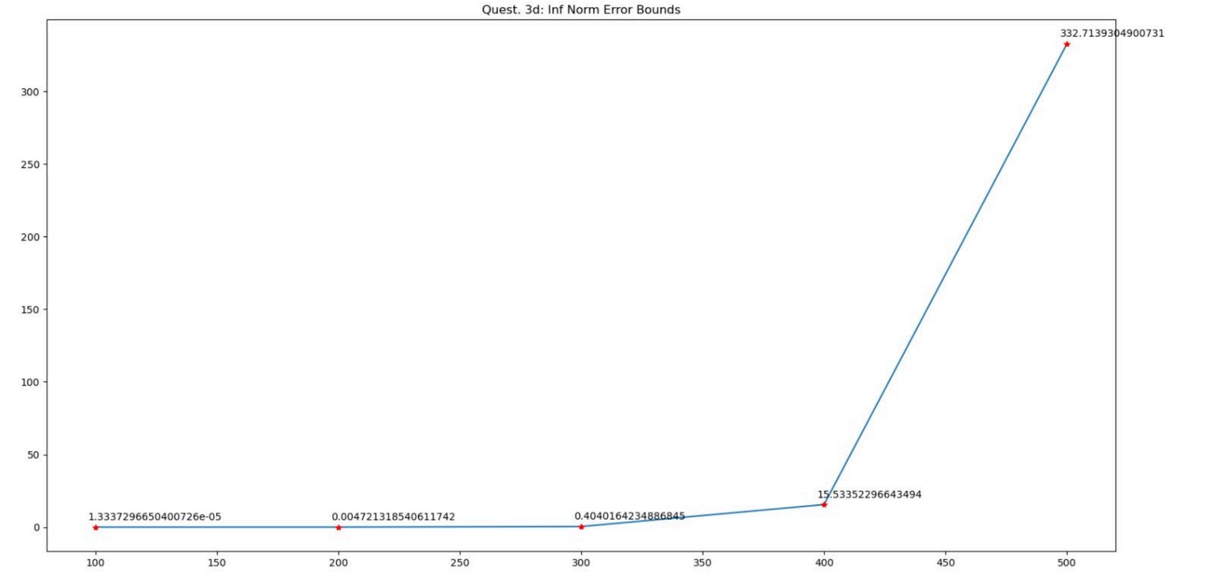
שאלה 3- גרפים:

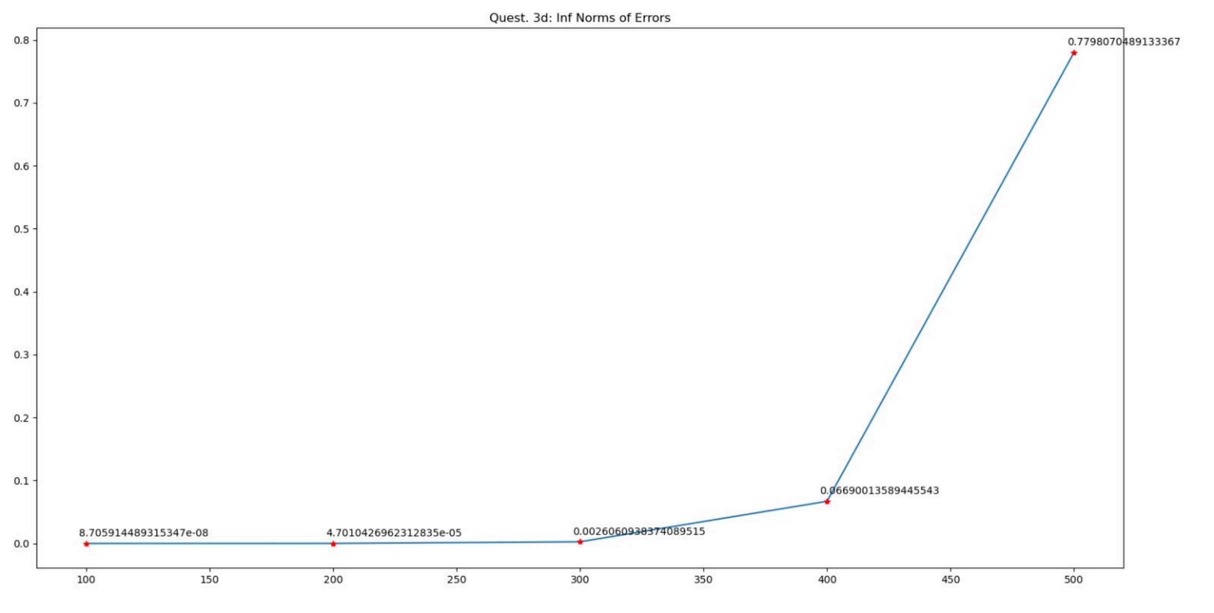

1e−6
Actual Inf Norm Relative errors
Theoretical Bound on Inf Norm Relative errors
2.00
1.75
1.50
1.25
1.00
0.75
0.50
0.25
0.00
100
150
200
250
300
350
400
450
500



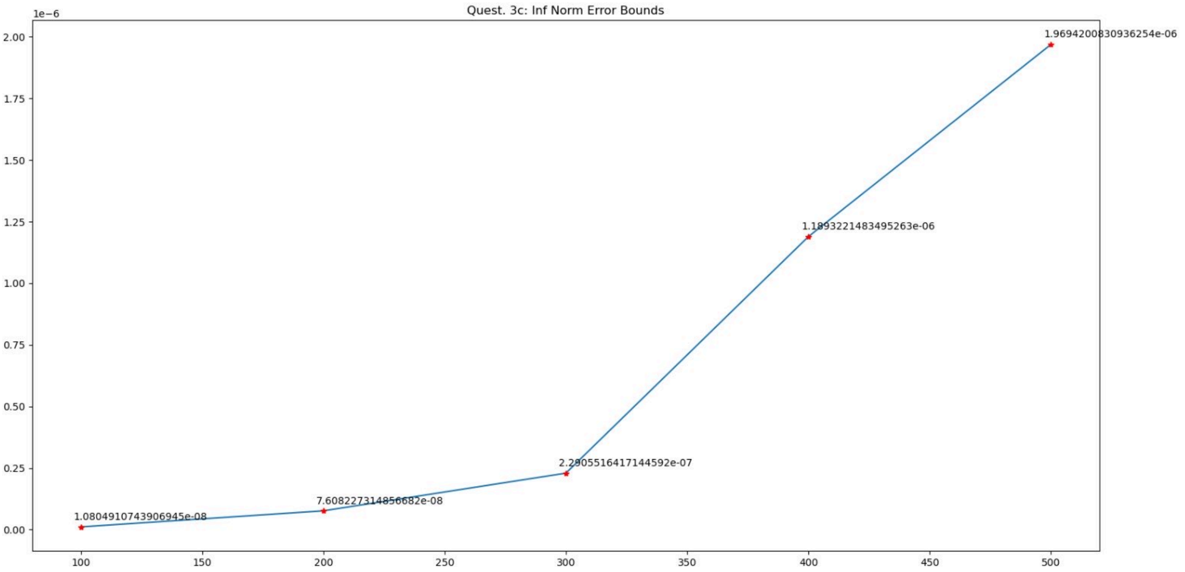
Quest. 3d: Condition Numbers
1e14
2.0
1.5
1.0
0.5
0.0
100
150
200
250
300
350
400
450
500
61843077.5575592
12906621990.496834
620391173731.2682
14751742711282.049
228578239129715.03

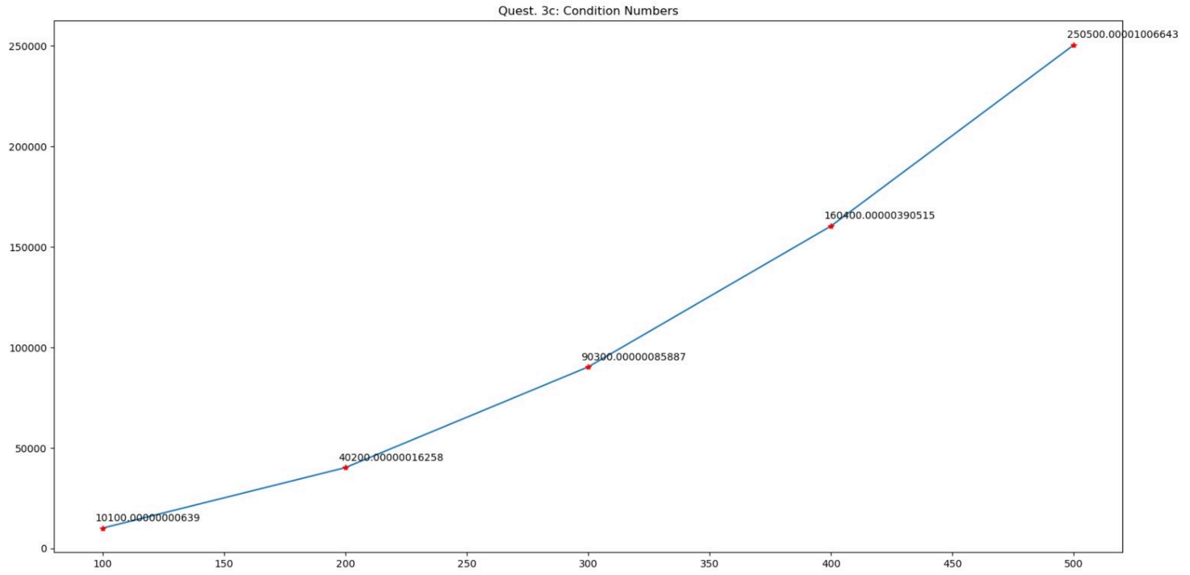

Quest. 3b: Inf Norms of Errors
1e−8
4.0
3.5
3.0
2.5
2.0
1.5
1.0
0.5
0.0
100
150
200
250
300
350
400
450
500
1.5612222625804861e-10
2.0402388667406512e-09
4.234585349038866e-09
1.4959962379990088e-08
3.9760287950940665e-08


Quest. 3d: Inf Norm Error Bounds
332.7139304900731
15.53352296643494
0.4040164234886845
0.004721318540611742
1.3337296650400726e-05
0
50
100
150
200
250
300
100
150
200
250
300
350
400
450
500



Quest. 3d: Inf Norms of Errors
8.705914489315347e-08
4.7010426962312835e-05
0.0026060938374089515
0.06690013589445543
0.7798070489133367
0.0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
100
150
200
250
300
350
400
450
500


Quest. 3c: Inf Norm Error Bounds
1e−6
2.00
1.75
1.50
1.25
1.00
0.75
0.50
0.25
0.00
100
150
200
250
300
350
400
450
500
1.9694200830936254e-06
1.1893221483495263e-06
2.2905516417144592e-07
7.608227314856682e-08
1.0804910743906945e-08


Quest. 3c: Condition Numbers
250500.00001006643
160400.00000390515
90300.00000085887
40200.00000016258
10100.00000000639
250000
200000
150000
100000
50000
0
100
150
200
250
300
350
400
450
500

שאלה 4- חלק קוד:

```
import numpy as np
import matplotlib.pyplot as plt

from fixed_pt import fixed_pt

def jacobi_decomposition(A):

    D, L, U = np.zeros_like(A), np.zeros_like(A), 
np.zeros_like(A)
    n = A.shape[0]
    for i in range(n):
        for j in range(n):
            curr = A[i, j]
            if i == j:
                D[i, j] = curr
            elif i > j:
                L[i, j] = curr
            else:
                U[i, j] = curr

    return D, L, U

def jacobi_iter_mat(A, b):

    D, L, U = jacobi_decomposition(A)
    D_inv = np.linalg.inv(D)
    W = -D_inv @ (L + U)
    c = D_inv @ b

    return W, c

def quest4_general(alpha, n, epsilon):

    A = np.zeros((n, n))
    for i in range(n):
        for j in range(n):
            if i == j:
                A[i, j] = 1
            elif i == j + 1:
                A[i, j] = -(1-alpha)
            elif j == i + 1:
                A[i, j] = -alpha

    b = np.zeros((n, 1))
```

```
    b[0] = 1 - alpha

    W, c = jacobi_iter_mat(A, b)

    x_0 = np.ones((n, 1)) * (1 / n)

    f = lambda x: W @ x + c
    x_sol, n_it, err = fixed_pt(f, x_0, epsilon, 1, sclr=0)

    return  x_sol, n_it, err


if __name__ == '__main__':

    dbg_plot = 0 #To compare Probability Decay for different 
alphas, same n

    epsilon = 0.00001

    n_c = 10
    alpha_c = 1/2
    x_sol_4c, n_it_4c, err_4c = quest4_general(alpha_c, n_c, 
epsilon)

    n_e = 50
    if dbg_plot:
        n_e = 10
    alpha_e = 1/3
    x_sol_4e, n_it_4e, err_4e = quest4_general(alpha_e, n_e, 
epsilon)

    if dbg_plot:
        x_sol_two_thirds, _, _ = quest4_general(2/3, n_c, 
epsilon)
        plt.figure()
        plt.plot(x_sol_two_thirds, 'g', label='quest 4c: 
alpha=2/3, n=' + str(n_c))
        plt.plot(x_sol_4c, 'b', label='quest 4c: alpha=1/2, n=' 
+ str(n_c))
        plt.plot(x_sol_4e, 'r', label='quest 4e: alpha=1/3, 
n='+str(n_e))
        plt.legend()
        plt.title('Probability Decay for Solution')
        plt.pause(10)
```

(4) <u>סעיף ראשון</u>: מתקיים $1\le i\le n-1$: $P_i = \frac{1}{2}P_{i-1} + \frac{1}{2}P_{i+1}$

שכן מהנקודה x_i הצפרדע יכולה עפ"י הנתונים לצעוד רק לנק' x_{i+1} או x_{i-1}. לכן

$$P_i = \text{ההסת' להגיע מ-}x_i\text{ ל-}x_0\text{ לפני }x_n = \underbrace{P(x_i \to x_{i+1})}_{\text{ההסתברות שבצעד הראשון יהיה ל-}x_{i+1}} \cdot P_{i+1} + \underbrace{P(x_i \to x_{i-1})}_{\text{באופן דומה}} \cdot P_{i-1}$$

$$= \frac{1}{2}P_{i+1} + \frac{1}{2}P_{i-1}$$

נעביר אגפים

$$P_i - \frac{1}{2}P_{i+1} = \frac{1}{2}P_{i-1}$$

כלומר

$$\begin{cases} P_1 - \frac{1}{2}P_2 = \frac{1}{2}\underbrace{P_0}_{=1 \text{ (כי הגע כבר נמצא ב-}x_0)} = \frac{1}{2} \\ -\frac{1}{2}P_1 + P_2 - \frac{1}{2}P_3 = 0 \\ \vdots \\ -\frac{1}{2}P_{n-3} + P_{n-2} - \frac{1}{2}P_{n-1} = 0 \\ -\frac{1}{2}P_{n-2} + P_{n-1} - \frac{1}{2}\underbrace{P_n}_{=0 \text{ (כי הגע כבר נמצא ב-}x_n)} = 0 \end{cases}$$

ובסה"כ

$$\begin{pmatrix} 1 & -\frac{1}{2} & 0 & \cdots & & 0 \\ -\frac{1}{2} & 1 & -\frac{1}{2} & 0 & \cdots & 0 \\ \vdots & & & & & \\ 0 & \cdots & 0 & -\frac{1}{2} & 1 & -\frac{1}{2} \\ 0 & \cdots & & 0 & -\frac{1}{2} & 1 \end{pmatrix} \begin{pmatrix} P_1 \\ \vdots \\ \vdots \\ P_{n-1} \end{pmatrix} = \begin{pmatrix} \frac{1}{2} \\ 0 \\ \vdots \\ 0 \end{pmatrix}$$

<u>סעיף לפני אחרון</u>: מאותם שיקולים בדיוק מתקיים
$P_0 = 1,\ P_n = 0,\quad P_i = \alpha P_{i+1} + (1-\alpha)P_{i-1} \quad (1\le i\le n-1)$
לכן מאותם שיקולים בדיוק מתקבלת מערכת המשוואות.

(4) (המשך)

$$\begin{cases} P_1 - \alpha P_2 = (1-\alpha)P_0 = 1-\alpha \\ -(1-\alpha)P_1 + P_2 - \alpha P_3 = 0 \\ \vdots \\ -(1-\alpha)P_{n-3} + P_{n-2} - \alpha P_{n-1} = 0 \\ -(1-\alpha)P_{n-2} + P_{n-1} - \underbrace{\alpha P_n}_{=0} = 0 \end{cases}$$

כלומר

$$\begin{pmatrix} 1 & -\alpha & 0 & \cdots & \cdots & 0 \\ -(1-\alpha) & 1 & -\alpha & 0 & \cdots & 0 \\ 0 & -(1-\alpha) & 1 & -\alpha & \cdots & 0 \\ \vdots & & & & & 0 \\ \vdots & & & & & 0 \\ 0 & \cdots & \cdots & -(1-\alpha) & 1 & -\alpha \\ 0 & \cdots & \cdots & 0 & -(1-\alpha) & 1 \end{pmatrix} \begin{pmatrix} P_1 \\ \vdots \\ \vdots \\ P_n \end{pmatrix} = \begin{pmatrix} 1-\alpha \\ 0 \\ \vdots \\ 0 \end{pmatrix}$$

~~[illegible]~~

~~[illegible]~~

~~[illegible]~~

סעיף שני: עבור A שלנו $D = D^{-1} = I$

ולכן מטריצת האיטרציה היא

$$W = -D^{-1}(L+U) = \begin{pmatrix} 0 & \frac{1}{2} & 0 & 0 & \cdots & 0 \\ \frac{1}{2} & 0 & \frac{1}{2} & 0 & \cdots & 0 \\ \vdots & \ddots & \ddots & \ddots & & \\ 0 & \cdots & & \frac{1}{2} & 0 & \frac{1}{2} \\ 0 & \cdots & & & \frac{1}{2} & 0 \end{pmatrix}, \quad c = D^{-1}b = b = \begin{pmatrix} 1/2 \\ 0 \\ \vdots \\ 0 \end{pmatrix}$$

המטריצה בשיטת יעקובי! היא Toeplitz, והע"ע המתאימים הם

$$\lambda_k = a + 2\sqrt{b \cdot c}\cos\left(\frac{k\pi}{n+1}\right) = 0 + 2\sqrt{\frac{1}{2}\cdot\frac{1}{2}}\cos\left(\frac{k\pi}{n+1}\right) = \cos\left(\frac{k\pi}{n+1}\right) : k = 1, \ldots, n$$

(4) (המשך II) $0 < \frac{k\pi}{n+1} < \pi$ עבור $k=1,\dots,n$, ולכן

$$|\cos\tfrac{k\pi}{n+1}| \le \cos\tfrac{\pi}{n+1} < -\cos\tfrac{n\pi}{n+1} < 1$$

(כי צורה נראית רק בקטע $[0,\pi]$

$\cos t = -\cos(\pi - t)$

כלומר $\boxed{\rho(W) < 1}$ ולכן במקרה זה יש <u>התכנסות</u> <u>לכל ניחוש התחלתי</u>.

<u>סעיף שלישי:</u> מצורף הקוד שמבצע את האלגו׳ עבור ניחוש התחלתי $\vec{x^0} = \begin{pmatrix} 1/n \\ \vdots \\ 1/n \end{pmatrix}$ עבור $n=10$. בחרתי סף שגיאה $\varepsilon = 10^{-4}$.

מס׳ האיטרציות שהאלגוריתם רץ: 213

<u>סעיף רביעי (לפני אחרון): השאלה</u> - באותם נימוקים בדיוק, מטריצת האיטרציה היא

$$W = \begin{pmatrix} 0 & \alpha & 0 & 0 & \cdots & 0 \\ 1-\alpha & 0 & \alpha & 0 & \cdots & 0 \\ 0 & 1-\alpha & 0 & \alpha & \cdots & 0 \\ 0 & & & & & \vdots \\ \vdots & & & 1-\alpha & 0 & \alpha \\ 0 & \cdots & & & 1-\alpha & 0 \end{pmatrix}, \quad c = D^{-1}b = \begin{pmatrix} 1-\alpha \\ 0 \\ \vdots \\ 0 \end{pmatrix}$$

ולכן המטריצה היא שוב טפליץ, והע״ע הם

$$\lambda_k = a + 2\sqrt{bc}\cdot\cos\left(\frac{k\pi}{n+1}\right) = 2\cdot\sqrt{\alpha(1-\alpha)}\cos\left(\frac{k\pi}{n+1}\right)$$

$(k=1,\dots,n+1)$

שוב גם כאן $0 < \left|\cos\left(\frac{k\pi}{n+1}\right)\right| < 1$

בנוסף, אם נסמן $h(\alpha) = \alpha(1-\alpha)$

$h''(\alpha) = -2 < 0$, $h'(\alpha) = (\alpha - \alpha^2)' = 1 - 2\alpha$

ונקבל שנק׳ המקסימום היא ב- $\alpha = \frac{1}{2}$

ומתקיים $\max_{\alpha\in[0,1]} \alpha(1-\alpha) = h(\tfrac{1}{2}) = \tfrac{1}{4}$.

ולכן

$$|\lambda_k| < 2\cdot\sqrt{\tfrac{1}{4}}\cdot 1 = 1$$

ולכן גם כאן $\rho(W) < 1$ ולכן שוב התכנסות השיטה מובטחת <u>לכל x^0</u>.

<u>סעיף חמישי ואחרון:</u> מצורף הקוד שמבצע את האלגו׳ עבור $\alpha = \frac{1}{3}$, $n=50$. שוב בחרתי ניחוש התחלתי $\begin{pmatrix} 1/n \\ \vdots \\ 1/n \end{pmatrix}$, $\varepsilon = 10^{-5}$ מס׳ האיטרציות שהיו: 332.