תרגיל בית 2 – מודלים לא ליניאריים בחקר ביצועים

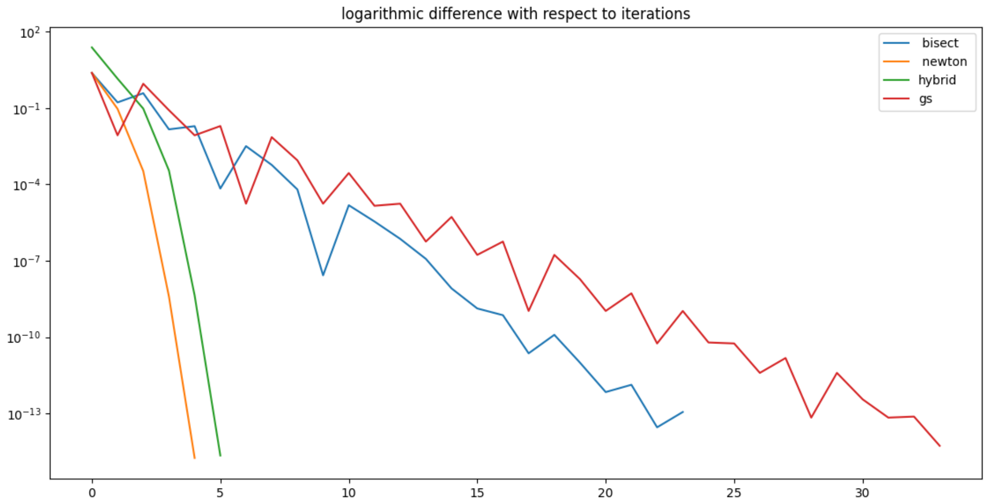
מגישים :

אלעד בוכריס – 206202426

משה דידי – 311395834

שאלה 1 :

Bisect - התכנסנו תוך 24 צעדים. פירוט : תוך איטרציה אחת לספרה הראשונה, 5 איטרציות לספרה שניה ושלישית, ומשם בין 2-3 איטרציות לדיוק נוסף לספרה.
Newton- ההתכנסות הסתיימה לאחר 4 איטרציות.
hybrid - נדרשו 5 איטרציות על מנת להגיע להתכנסות, נשים לב כי מכיוון שהערך ההתחלתי רחוק יותר מהאופטימלי, נדרשות יותר איטרציות ביחס לשיטת ניוטון הסטנדרטית.
Golden section - התכנסות לאחר 33 איטרציות.



ניתן לראות על פי הגרף את ההבדל בין קצב ההתכנסות בין השיטות, כאשר שיטת newton מתכנסת בקצב הכי מהיר (קצב ריבועי), אחריה ההיברידית מכיוון שהערך ההתחלתי שונה, לאחר מכן שיטת החציה ולבסוף שיטת חתך הזהב, אשר משתמש בערכי הפונקציה בלבד.
קוד :

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


def f(x):
    """Calculating f(x) """
    return np.square(x) + (np.square(x) - 3 * x + 10) / (2 + x)


def df(x):
    """Calculating the derivative for f(x)"""
    return 2 * x + (np.square(x) + 4 * x - 16) / np.square(2 + x)


def ddf(x):
    """Calculating the second derivative for f(x)"""
    return 2 + 2 / (2 + x) - 2 * (2 * x - 3) / np.square(2 + x) + 2 * (np.square(x) - 3 * x +
10) / np.power(2 + x, 3)

def generic_gs(f, l, u, eps, k):
    """Running the golden section algorithm """
    Validation.validation(k,l, u, eps)
    fv = []
```

```python
    counter = 0
    tau = (3 - np.sqrt(5)) / 2
    x2 = l + tau * (u - l)
    x3 = l + (1 - tau) * (u - l)
    f_x2 = f(x2)
    f_x3 = f(x3)
    while np.abs(u - l) >= eps and counter <= k:
        fv.append(f((u + l) / 2))
        if f_x2 < f_x3:
            u = x3
            x3 = x2
            f_x3 = f_x2
            x2 = l + tau * (u - l)
            f_x2 = f(x2)
        else:
            l = x2
            x2 = x3
            f_x2 = f_x3
            x3 = l + (1 - tau) * (u - l)
            f_x3 = f(x3)
        counter += 1
    fv.append(f((u + l) / 2))
    return (l + u) / 2, fv


def generic_hybrid(f, df, ddf, l, u, eps1, eps2, k):
    """Running the hybrid newton algorithm"""
    Validation.validation(k,l, u, eps1,eps2)
    fv = []
    x = u
    counter = 0
    while counter <= k and np.abs(df(x)) > eps2 and np.abs(u - l) >= eps1:
        fv.append(f(x))
        x_new = x - df(x) / ddf(x)
        if (l <= x_new <= u) and np.abs(df(x_new)) < 0.99 * np.abs(df(x)):
            x = x_new
        else:
            x = (l + u) / 2
        if df(u) * df(x) > 0:
            u = x
        else:
            l = x
    fv.append(f(x))
    return x, fv


def generic_newton(f, df, ddf, x0, eps, k):
    """Running the generic newton algorithm"""
    Validation.validation(k, eps)
    fv = []
    x = x0
    counter = 0
    while counter <= k and np.abs(df(x)) > eps:
        fv.append(f(x))
        counter += 1
        x = x - df(x) / ddf(x)
    fv.append(f(x))
    return x, fv


def generic_bisect(f, df, l, u, eps, k):
    """Running the generic bisect algorithm"""
    Validation.validation(k,l, u, eps)
    fv = []
    x = (u + l) / 2
    counter = 0
    while np.abs(u - l) >= eps and counter < k:
        fv.append(f(x))
        if df(u) * df(x) > 0:
            u = x
        else:
            l = x
        counter += 1
        x = (u + l) / 2
    fv.append(f(x))
    return x, fv
```

```python
def main():
    l = -1
    u = 5
    x0 = (u + l) / 2
    k = 50
    eps = eps1 = eps2 = 1 / np.power(10, 6)
    x_bisect, fv_bisect = generic_bisect(f, df, l, u, eps, k)
    plt.semilogy(np.arange(len(fv_bisect)), np.array(fv_bisect) - 3.5825439993037,
                 label=" bisect ")
    x_newton, fv_newton = generic_newton(f, df, ddf, x0, eps, k)
    plt.semilogy(np.arange(len(fv_newton)), np.array(fv_newton) - 3.5825439993037,
                 label=" newton ")
    x_hybrid, fv_hybrid = generic_hybrid(f, df, ddf, l, u, eps1, eps2, k)
    plt.semilogy(np.arange(len(fv_hybrid)), np.array(fv_hybrid) - 3.5825439993037,
                 label="hybrid")
    x_gs, fv_gs = generic_gs(f, l, u, eps, k)
    plt.semilogy(np.arange(len(fv_gs)), np.array(fv_gs) - 3.5825439993037,
                 label="gs")
    plt.title("logarithmic difference with respect to iterations")
    plt.legend()
    plt.show()
```

שאלה 2:

נתונה מטריצה $Q \in \mathbb{R}^{n\times n}$ סימטרית ומוגדרת אי שלילית. נתון $a \neq 0 \in \mathbb{R}^n$
תהי

$$f: \mathbb{R} \to \mathbb{R}$$
$$f(\mu) = a^T(Q + \mu I_n)^{-1}a$$

א. יש להוכיח כי f מוגדרת לכל $\mu > 0$.
הוכחה:
Q סימטרית ולכן קיימת מטריצה אורתונורמלית $U \in \mathbb{R}^{n\times n}$ שעמודותיה הן הו״ע של Q כך ש- :

$$Q = Udiag(\lambda_1, \dots, \lambda_n)U^T$$

בנוסף, מכיוון ש $UU^T = I$ מתקיים גם :

$$\mu I = U\mu I U^T$$

ולכן :

$$Q + \mu I_n = Udiag(\lambda_1, \dots, \lambda_n)U^T + U\mu I_n U^T = U(diag(\lambda_1, \dots, \lambda_n) + \mu I_n)U^T$$

כך ש- λ_i הע״ע ה- i של Q.

נתון כי Q מוגדרת אי שלילית ולכן $Q + \mu I > 0\ \forall \mu > 0$ מכיוון ש-I מט׳ מוגדרת חיובית, לכן כל הע״ע העצמיים $\lambda_i + \mu > 0$. לכן $Q + \mu I$ הפיכה.
כלומר קיימת מטריצה $(Q + \mu I_n)^{-1}$ עבור כל $\mu > 0$.
לכן, f מוגדרת בתחום זה.
מש״ל.
.

ב. נשים לב כי הע״ע של המטריצה ההופכית הינם: $\frac{1}{\mu+\lambda_i}$ וכי מתקיים $\lambda_i + \mu > 0$. לכן, מטריצה זו מוגדרת חיובית ולפי הגדרה מתקיים כי $f(\mu) > 0\ \forall \mu > 0$ עבור $a \neq \vec{0}$.
מש״ל.

ג. מכיוון ש-$Q + \mu I$ סימטרית אז גם $A = (Q + \mu I)^{-1}$ סימטרית ולכן קיים לה פירוק ספקטרלי מהצורה:

$$f(\mu) = a^T M\left(diag\left(\frac{1}{\mu + \lambda_1}, \dots, \frac{1}{\mu + \lambda_n}\right)\right)M^T a \overset{like\ hw1}{=} \sum_{k=1}^{n} \frac{1}{\mu + \lambda_k}\left(\sum_{i=1}^{n} a_i m_{ik}\right)^2$$

נגזור לפי μ ונקבל :

$$f'(\mu)=\sum_{k=1}^{n}\frac{-1}{(\mu+\lambda_k)^2}\left(\sum_{i=1}^{n}a_i m_{ik}\right)^2<0\quad\forall\mu>0$$

הנגזרת קטנה מאפס ולכן הפונקציה יורדת בתחום זה.

ד. יש למצוא α כך ש-

$$f(\mu)=\alpha.$$

נפעיל את שיטת החציה על הפונקציה :

$$g(\mu)=f(\mu)-\alpha$$

אם נמצא שורש ל-g, נמצא μ המקיים את הדרישה.

בתרגיל בית 1, הראנו כי תבנית ריבועית *של מטריצה סימטרית* בעלת פירוק ספקטרלי חסומה מלמעלה על ידי

$$\frac{1}{\mu+\lambda_{min}}\|a\|^2$$

ומלמטה על ידי

$$\frac{1}{\mu+\lambda_{max}}\|a\|^2$$

לכן, f חסומה על ידיהם

כאשר אלו הערכים העצמיים המינימליים והמקסימליים של A.

לכן

$$\frac{1}{\mu+\lambda_{max}}\|a\|^2\le\alpha\le\frac{1}{\mu+\lambda_{min}}\|a\|^2$$

פסאדו קוד למציאת קצוות קטע התחלתי :

$1.\ initiate\ some\ random\ \ t>0$
$2.\ if\ \ g(t)<0:$
$2.1\qquad u\leftarrow t$
$2.2\qquad t\leftarrow\frac{t}{2}$
$2.3\qquad while\ g(t)<0:$
$2.3.1\qquad\qquad t\leftarrow\frac{t}{2}$
$2.4\qquad l\leftarrow t$
$3.\ else:$
$3.1\qquad l\leftarrow t$
$3.2\qquad while\ g(t)>0:$
$3.2.1\qquad\qquad t\leftarrow 2t$
$3.3\qquad u\leftarrow t$

ה. כאשר מצאנו קטע התחלתי $[l_0,u_0]$, ניתן לראות כי בכל איטרציה אנו חותכים את אורך הקטע בחצי.

כלומר, אורך הקטע באיטרציה ה-k יהיה

$$u_k-l_k=\left(\frac{1}{2}\right)^k(u_0-l_0)$$

אנו עוצרים ברגע שאורך הקטע קטן או שווה לשגיאה ε.

כלומר, כאשר :

$$\left(\frac{1}{2}\right)^k(u_0-l_0)\le\varepsilon$$

נכתוב בצורה הבאה :

$$2^{-k}\le\frac{\varepsilon}{(u_0-l_0)}$$

נפעיל $\log_2(x)$ על שני האגפים ונקבל

$$-k \leq \log_2 \frac{\varepsilon}{(u_0 - l_0)} \rightarrow \boldsymbol{k \geq \log_2 \frac{u_0 - l_0}{\varepsilon}}$$

ניתן לראות כי ככל שאורך הקטע ההתחלתי גדל כך גם מספר האיטרציות וככל ש-ε גדל גם כך מספר האיטרציות גדל.

<u>שאלה 3 :</u>

נתון פולינום $p: \mathbb{R} \rightarrow \mathbb{R}$.

$$p(x) = -3.55x^3 + 1.1x^2 + 0.765x - 0.74$$

ונתונה $x^0 = 0.5554$.

א. ניתן לראות כי

$$\lim_{x\rightarrow\infty} f(x) = -\infty$$

וכי

$$\lim_{x\rightarrow-\infty} f(x) = \infty$$

ולכן ל-p יש לפחות שורש אחד.

ניתן לראות כי

$$f(-1) = 3.15 > 0$$
$$f(-0.5) = -0.404 < 0$$

ולכן

$$f(-0.5) \cdot f(-1) < 0$$

ואלו יכולים לשמש כקצוות הקטע הסגור $[-1, -0.5]$ המכיל שורש של p.

ב.

$$p(x) = -3.55x^3 + 1.1x^2 + 0.765x - 0.74$$

$$p'(x) = -10.65x^2 + 2.2x + 0.765$$
$$p''(x) = -21.3x + 2.2$$

<u>שלוש איטרציות :</u>

1.

$$x^1 = x^0 - \frac{p(x^0)}{p'(x_0)} = 0.5554 - \frac{-0.5840}{-1.2983} = 0.1056$$

2.

$$x^2 = 0.1056 - \frac{-0.6511}{0.8785} = 0.8467$$

3.

$$x^3 = 0.8467 - \frac{-1.4585}{-5.0072} = 0.5554$$

על מנת שהסדרה $\{x^n\}_{(n\in\mathbb{N})}$ תתכנס ראינו כי צריך להתקיים כי

$$\forall x \in I, p'(x) \geq m : m > 0, I = [x^* - \delta, x^* + \delta]$$

אנו יודעים כי $x^* \in [-1, -0.5]$

ניתן לראות כי

$$p'(-1) = -12.1$$

$$p'(-0.5) = -2.99$$

וכי $p'(x)$ מונוטונית עולה בתחום זה (פרבולה קעורה לפני נקודת הקיצון)

מתקיים כי

$$p'(x) < 0 \ \forall x \in [-1, -0.5]$$

ולכן לא קיימת סביבה I עבורה התנאי מתקיים.

ג.

$$-10.65x^2 + 2.2x + 0.765 = 0 \rightarrow \boldsymbol{x_1} = -\mathbf{0.184}\,, \boldsymbol{x_2} = \mathbf{0.391}$$

הסיבה שלא כל האיטצריות בוודאות מוגדרות היטב היא מכיוון שקיימות הנקודות הסטציונריות בהן $p'(x) = 0$ ולכן, אם "ניפול" בנקודות אלו, x^{k+1} לא מוגדר.

ד. באיטרציות בהן שיטת ניוטון לא מוגדרת, השיטה ההיברידית משתמשת בשיטת החציה, אשר תמיד מוגדרת, לכן שיטה זו תמיד מוגדרת. מכיוון שהאיטרציות תמיד מוגדרות שיטת החציה תמיד מוגדרת היטב, אזי בהינתן והתחלנו בקטע המכיל את השורש השיטה תמיד תתכנס.

קוד :

```
import numpy as np
import Validation


def hybrid(f, df, l, u, eps1, eps2):
    """Running the hybrid newton algorithm"""
    Validation.validation(l=l, u=u, eps1=eps1, eps2=eps2)
    x = u
    while np.abs(df(x)) > eps2 and np.abs(u - l) >= eps1:
        x_new = x - f(x) / df(x)
        if (l <= x_new <= u) and np.abs(f(x_new)) < 0.99 * np.abs(f(x)):
            x = x_new
        else:
            x = (l + u) / 2
        if f(u) * f(x) > 0:
            u = x
        else:
            l = x
    return x


def ex2(l, u, eps):
    r = hybrid(f=lambda x: -3.55 * np.power(x, 3) + 1.1 * np.square(x)
                           + 0.765 * x - 0.74, df=lambda x: -10.65 * np.square(x) +
2.2 * x + 0.765
               , l=l, u=u, eps1=eps, eps2=eps)
    return r
```

השורש המתקבל :

$$r = -0.6081345342894493$$

שאלה 4 :

א. מוגדרת הפונקציה :

$$\phi: \mathbb{R} \to \mathbb{R}$$

$$.\phi_k(t) = f(\mathbf{x}_1, \ldots, \mathbf{x}_{k-1}, t, \mathbf{x}_{k+1}, \ldots, \mathbf{x}_n)$$

נגדיר

$$t = x_k$$
$$C = 1$$
$$\mu = \alpha$$
$$b = x_{k-1}$$
$$c = x_{k+1}$$
$$a = s_k$$
$$D = \sum_{i=2}^{k-1} |x_i - x_{i-1}| + \sum_{i=k+2}^{n} |x_i - x_{i-1}| + \alpha\left(\sum_{i=1}^{k-1}(x_i - s_i)^2 + \sum_{i=k+1}^{n}(x_i - s_i)^2\right)$$

עבור המקרים בהם $n \in \{1, N\}$:

$$\mu = 2\alpha \ \ , b = c \ \ , C = \frac{1}{2}$$

ונקבל את הביטוי :

$$,\phi_k(t) = C\left(\mu(t-a)^2 + |t-b| + |t-c|\right) + D$$

ב. נתונה הפונקציה $\varphi: \mathbb{R} \to \mathbb{R}$.

$$.\varphi(t) = \mu(t-a)^2 + |t-b| + |t-c|$$

נגדיר $l = \min\{a, b, c\} - 1$ $u = \max\{a, b, c\} + 1$.

נניח כי קיים מינימום גלובלי ל φ כאשר :

$$argmin\ \varphi(t) = t^*$$

אזי :

$$\varphi(t^*) \le \varphi(t) \quad \forall t \in \mathbb{R}$$

נתסכל כעת על $t \le l$, עבורו מתקיים כי

$$t - b < 0\,, t - c < 0 \to |t-b| = b - t, |t-c| = c - t$$

ולכן :

$$\varphi(t) = \mu(t-a)^2 + b - t + c - t$$
$$\varphi'(t) = 2\mu(t-a) - 2$$

בסעיף הקודם הגדרנו $\mu = \alpha > 0$ ולכן $\varphi'(t) < 0\ \forall t \le l$.

ולכן הפונקציה יורדת בתחום זה.

כעת, נסתכל $t \geq u$.
מתקיים כי

$$|t-b| = t-b\,, |t-c| = t-c$$

ולכן

$$\varphi(t) = \mu(t-a)^2 + t - b + t - c$$

$$\varphi'(t) = 2\mu(t-a) + 2$$

מאותן סיבות כמו קודם מתקיים כי

$$\varphi'(t) > 0\; \forall t \geq u$$

לכן, הפונקציה עולה בתחום זה.

לפי הנתון כי קיים מינימום גלובלי, אז מכיוון שהפונקציה יורדת לפני l ועולה אחרי u, המינימום חייב להתקבל בתחום $[l, u]$.

ג. קוד :

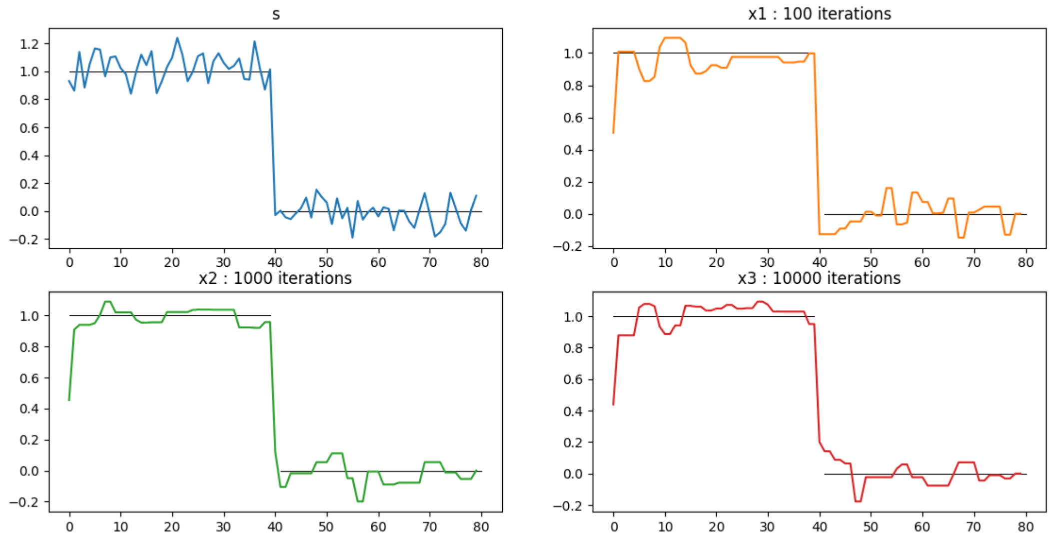
```
def phi(t, mu, a, b, c):
    return mu * np.square(t - a) + np.abs(t - b) + np.abs(t - c)


def gs_denoise_step(mu, a, b, c):
    """Running the golden section algorithm """
    eps = 1 / np.power(10, 10)
    tau = (3 - np.sqrt(5)) / 2
    l = np.min([a, b, c]) - 1
    u = np.max([a, b, c]) + 1
    t2 = l + tau * (u - l)
    t3 = l + (1 - tau) * (u - l)
    f_t2 = phi(t2, mu, a, b, c)
    f_t3 = phi(t3, mu, a, b, c)
    while np.abs(u - l) >= eps:
        if f_t2 < f_t3:
            u = t3
            t3 = t2
            f_t3 = f_t2
            t2 = l + tau * (u - l)
            f_t2 = phi(t2, mu, a, b, c)
        else:
            l = t2
            t2 = t3
            f_t2 = f_t3
            t3 = l + (1 - tau) * (u - l)
            f_t3 = phi(t3, mu, a, b, c)
    return (l + u) / 2
```

ניתן להפעיל את השיטה מכיוון ש- φ רציפה בקטע (סכום של פונקציות רציפות).
מובטחת התכנסות מהסיבה שהיא יונימודלית (נתון כי קיים מינימום יחיד בקטע).

ד. קוד:

```
def gs_denoise(s, alpha, N):
    x = s
    for i in range(N):
        x[0] = (gs_denoise_step(2 * alpha, s[0], x[1], x[1])) / 2
        for k in range(1, len(s) - 1):
            x[k] = gs_denoise_step(alpha, s[k], x[k - 1], x[k + 1])
        x[len(x) - 1] = (gs_denoise_step(2 * alpha, s[len(x) - 1],
x[len(x) - 1], x[len(x) - 1])) / 2
    return x
```

ה.



ניתן לראות כי אין הבדל מהותי באיכות הקירוב לפי כמות האיטרציות.

מכיוון ש-x חלק יותר אז לדעתנו הקירוב שלו לאות האמיתי טוב יותר מאשר הקירוב s.

מכיוון שאנו יודעים כי האות המקורי בדיד וניתן לראות הבדל משמעותי בין ערכי f סביב 0 לבין ערכי f סביב 1, ניתן לשחזר את האות הבדיד בקלות ובקירוב טוב.

לכן, לדעתנו הקירוב שהאלגוריתם מצב הוא קירוב טוב של האות.