<u>אנליזה נומרית – עבודה 5</u>

פן אייל, אילנה פרבוי

שאלה 1

נתון: $I = \int_0^1 \ln\left(1 + x^2\right)$

א) נבצע קירוב לאינטגרל באמצעות שיטת הטרפז המרוכבת עבור m תתי קטעים בתחום $[0,1]$.

$$f(x) = \ln\left(1 + x^2\right)$$

$$h = \frac{b-a}{m} = \frac{1-0}{m} = \frac{1}{m}$$

ולכן:

$$I^{num} = \frac{h}{2}\big(f(0) + f(h)\big) + \frac{h}{2}\big(f(h) + f(2h)\big) + \cdots + \frac{h}{2}\left(f\left(\overbrace{(m-1)h}^{1-h}\right) + f\left(\overbrace{mh}^{1}\right)\right)$$

$$= \frac{h}{2}\Big(f(0) + 2f(h) + \cdots + 2f\big((m-1)h\big) + f(mh)\Big)$$

$$= \frac{h}{2}\big(f(0) + f(mh)\big) + h\left(f(h) + f(2h) + \cdots + f\big((m-1)h\big)\right)$$

$$= \frac{h}{2}\left(\overbrace{ln1}^{0} + \ln\left(\overbrace{1 + 1^2}^{2}\right)\right) + h\sum_{i=1}^{m-1} f(ih) \overset{h=\frac{1}{m}}{\cong} \frac{1}{2m} ln2 + \frac{1}{m}\sum_{i=1}^{m-1} f\left(\frac{i}{m}\right)$$

$$= \boldsymbol{\frac{ln2}{2m} + \frac{1}{m}\sum_{i=1}^{m-1} \ln\left(1 + \left(\frac{i}{m}\right)^2\right)}$$

ב) נמצא את m הקטן ביותר כך ש $\left|E^{total}\right| \leq 10^{-3}$.

ראינו בהרצאה שעבור שיטת הטרפזים המרוכבת מתקיים:

$$\left|E^{total}\right| \leq \frac{1}{12} h^3 * m * \sup_{\eta \in [0,1]} f''(\eta)$$

נחשב את הנגזרת השנייה של הפונקציה הנתונה:

$$f'(x) = \frac{2x}{1 + x^2}$$

$$f''(x) = \frac{2(1 + x^2) - 2x * 2x}{(1 + x^2)^2} = \frac{2(1 + x^2 - 2x^2)}{(1 + x^2)^2} = \frac{2(1 - x^2)}{(1 + x^2)^2} = \frac{2 - 2x^2}{1 + 2x^2 + x^4}$$

ונחפש את הסופרימום שלה בקטע $[0,1]$:

$$f^{(3)}(x)=\frac{-4x(1+2x^2+x^4)-(4x+4x^3)(2-2x^2)}{(1+2x^2+x^4)^2}$$
$$=\frac{-4x-8x^3-4x^5-8x+8x^3-8x^3+8x^5}{(1+2x^2+x^4)^2}=\frac{-12x-8x^3+4x^5}{(1+2x^2+x^4)^2}$$
$$=\frac{4x(x^2-3)}{(x^2+1)^3}\overset{?}{=}0$$

$$4x(x^2-3)\overset{?}{=}0 \leftrightarrow$$
$$x=0 \ or\ x^2=3 \ \leftrightarrow$$
$$x=0\ or\ x=\pm\sqrt{3}$$

קיבלנו שהנקודות החשודות לקיצון אינן בקטע $(0,1]$ ולכן הפונקציה מונוטונית בתחום.
נבדוק את ערכיה בקצוות:

$$f''(0)=\frac{2(1-0^2)}{(1+0^2)^2}=2$$

$$f''(1)=\frac{2(1-1^2)}{(1+1^2)^2}=0$$

ולכן $\sup_{\eta\in[0,1]} f''(\eta)=2$.
נציב ונקבל:

$$|E^{total}|\le\frac{1}{12}h^3*m*\sup_{\eta\in[0,1]}f''(\eta)=\frac{1}{12}\left(\frac{1}{m}\right)^3*m*2=\frac{1}{6m^2}\overset{?}{\lesssim}10^{-3}$$
$$\frac{1}{6m^2}\overset{?}{\lesssim}10^{-3} \ \leftrightarrow$$
$$1\le 0.006m^2 \leftrightarrow$$
$$\frac{500}{3}\le m^2 \leftrightarrow$$
$$m\gtrsim 12.91\ or\ m\lesssim -12.91$$

לכן $\boldsymbol{m=13}$

וע"י הרצת הקוד הבא בפייתון:

```python
import math

m = 70
sum = 0.
for i in range(1, m):
    sum += math.log(1 + (i / m)**2)
ans = math.log(2)/(2*m) + (1 / m) * sum
print(f"I_num: {ans}")
```

נקבל כי: $\boldsymbol{I^{num}=0.2644366525894482}$

ג) ראינו בסעיף ב׳ ש $0 \geq \frac{2(1-x^2)}{(1+x^2)^2} = f''(x)$ לכל $x \in [0,1]$, כלומר הפונקציה היא קמורה בקטע ומתכונות של קמירות, כל ישר שנעביר בין 2 נקודות על הפונקציה יהיה מעליה ולכן עבור $m \geq 1$, שיטת הטרפזים המורכבת תחשב קירוב לאינטגרל (=שטח של טרפז) שיהיה גדול משטח האינטגרל האמיתי בקטע.

לכן $E^{total} = I - I^{num} \leq 0$ לכל $m \geq 1$, כלומר, **השגיאה תמיד תהיה בעלת סימן שלילי.**

שאלה 2

נתון המודל $\frac{dy}{dx} = x + y = f(x, y(x))$ כאשר $y(0) = 1$.

נפתור אותו באמצעות שיטת אולר עם $h = 0.1$ ובאמצעות שיטת רנגה-קטה עם $h = 0.1,\ \lambda = \frac{2}{3}$.

עבור שיטת אולר נשתמש בכלל החישוב הבא:

$$y(x_{i+1}) = y(x_i) + h * f\big(x_i, y(x_i)\big)$$

ועבור שיטת רנגה-קטה נשתמש בכלל החישוב הבא:

$$y(x_{i+1}) = y(x_i) + h * (\alpha_1 k_1 + \alpha_2 k_2)$$

כאשר:

$$k_1 = f\big(x_i, y(x_i)\big), \quad k_2 = f(x_i + \lambda * h,\ y_i + \lambda * h * k_1)$$

$$\alpha_1 = 1 - \frac{1}{2\lambda}, \qquad \alpha_2 = \frac{1}{2\lambda}$$

בחרנו לממש את החישוב כקוד *python* (אשר יצורף בסוף השאלה) תוצאות ריצת הקוד הינם:

א)

```
Exact: x_1 = 0.1,     y_1 = 1.1103418361512953
Exact: x_2 = 0.2,     y_2 = 1.2428055163203395
Exact: x_3 = 0.3,     y_3 = 1.399717615152006
Exact: x_4 = 0.4,     y_4 = 1.5836493952825408
Exact: x_5 = 0.5,     y_5 = 1.7974425414002564
Exact: x_6 = 0.6,     y_6 = 2.0442376007810177
Exact: x_7 = 0.7,     y_7 = 2.327505414940952
Exact: x_8 = 0.8,     y_8 = 2.651081856984935
Exact: x_9 = 0.9,     y_9 = 3.0192062223138985
Exact: x_10 = 1.0,    y_10 = 3.43656365691809
```

ב)

```
EM: x_1 = 0.1,     y_1 = 1.1
EM: x_2 = 0.2,     y_2 = 1.2200000000000002
EM: x_3 = 0.3,     y_3 = 1.362
EM: x_4 = 0.4,     y_4 = 1.5282
EM: x_5 = 0.5,     y_5 = 1.72102
EM: x_6 = 0.6,     y_6 = 1.943122
EM: x_7 = 0.7,     y_7 = 2.1974342
EM: x_8 = 0.8,     y_8 = 2.48717762
EM: x_9 = 0.9,     y_9 = 2.8158953820000003
EM: x_10 = 1.0,    y_10 = 3.1874849202
```

```
RK2: x_1 = 0.1,     y_1 = 1.11
RK2: x_2 = 0.2,     y_2 = 1.24205
RK2: x_3 = 0.3,     y_3 = 1.39846525
RK2: x_4 = 0.4,     y_4 = 1.5818041012500001
RK2: x_5 = 0.5,     y_5 = 1.7948935318812502
RK2: x_6 = 0.6,     y_6 = 2.0408573527287817
RK2: x_7 = 0.7,     y_7 = 2.3231473747653038
RK2: x_8 = 0.8,     y_8 = 2.6455778491156607
RK2: x_9 = 0.9,     y_9 = 3.012363523272805
RK2: x_10 = 1.0,    y_10 = 3.42816169321645
```

ג) מספר הפעמים ש$f(x, y)$ נקראה על ידי EM הינו 10.
← פעם אחת בכל איטרציה.
אמנם, מספר הפעמים ש$f(x, y)$ נקראה על ידי $RK2$ הינו 20.
← פעמיים בכל איטרציה עבור חישוב k_1 ו- k_2.

ד) עבור $h = 0.05$, שיטת אולר תקרא ל $f(x, y)$ 20 פעמים, זאת מכיוון שבכדי להגיע לערך $x = 1.0$ ידרשו 20 איטרציות.

ה) נחשב את השגיאה של EM ושל $RK2$ ע״י חישוב $|Exact\ value - EM\ guess|$ ו- $|Exact\ value - RK2\ guess|$ בהתאמה.
נריץ את הקוד עם $h = 0.1$ עבור $RK2$ ועם $h = 0.05$ עבור EM:

```
-- For x_1 = 0.1:
EM:    y_1 = 1.1025
RK2:   y_1 = 1.11
EM  error is: 0.0078418361512953
RK2 error is: 0.00034183615129523837

-- For x_2 = 0.2:
EM:    y_2 = 1.22075625
RK2:   y_2 = 1.24205
EM  error is: 0.022049266320339544
RK2 error is: 0.0007555163203394333

-- For x_3 = 0.3:
EM:    y_3 = 1.356383765625
RK2:   y_3 = 1.39846525
EM  error is: 0.04333384952700614
RK2 error is: 0.0012523651520059964

-- For x_4 = 0.4:
EM:    y_4 = 1.5111631016015623
RK2:   y_4 = 1.5818041012500001
EM  error is: 0.07248629368097848
RK2 error is: 0.0018452940325406342

-- For x_5 = 0.5:
EM:    y_5 = 1.6870573195157226
RK2:   y_5 = 1.7948935318812502
EM  error is: 0.11038522188453381
RK2 error is: 0.0025490095190061623

-- For x_6 = 0.6:
EM:    y_6 = 1.9362306947660841
RK2:   y_6 = 2.0408573527287817
EM  error is: 0.10800690601493357
RK2 error is: 0.0033802480522360234

-- For x_7 = 0.7:
EM:    y_7 = 2.2161943409796074
RK2:   y_7 = 2.3231473747653038
EM  error is: 0.11131107396134476
RK2 error is: 0.004358040175648448

-- For x_8 = 0.8:
EM:    y_8 = 2.5301042609300173
RK2:   y_8 = 2.6455778491156607
EM  error is: 0.12097759605491776
RK2 error is: 0.005504007869274297

-- For x_9 = 0.9:
EM:    y_9 = 2.881439947675344
RK2:   y_9 = 3.012363523272805
EM  error is: 0.13776627463855462
RK2 error is: 0.0068426990410932476

-- For x_10 = 1.0:
EM:    y_10 = 3.2740375423120667
RK2:   y_10 = 3.42816169321645
EM  error is: 0.1625261146060235
RK2 error is: 0.00840196370164037
```

נשים לב כי בכל איטרציה, השגיאה של $RK2$ יותר קטנה מאשר השגיאה של EM.

קוד *python* עבור שאלה 2 :

```python
import math

# params
h = 0.1
lam = 2. / 3.
alpha2 = (1. / (2. * lam))
alpha1 = 1. - alpha2
iter = 10


def exact_res(x):
    return 2 * math.pow(math.e, x) - x - 1

print("----------------------TASK2:A----------------------")
x = 0.
for i in range(1, iter + 1):
    x = round(x + h, 2)
    print(f"Exact: x_{i} = {x},       y_{i} = {exact_res(x)}")
print("\n----------------------TASK2:B----------------------")

def f(x, y):
    return x + y


# Euler Method
def calc_EM_next_y(x, y, h):
    return y + h * f(x, y)


x = 0.
y_EM = 1.
for i in range(1, iter + 1):
    y_EM = calc_EM_next_y(x, y_EM, h)
    x = round(x + h, 2)
    print(f"EM: x_{i} = {x},       y_{i} = {y_EM}")
print("---------------------------------------------------")

# Runge-Kutta
def calc_RK2_next_y(x, y, h):
    k1 = f(x, y)
    k2 = f(x + lam * h, y + lam * h * k1)
    return y + h * (alpha1 * k1 + alpha2 * k2)

x = 0.
y_RK2 = 1.
for i in range(1, iter + 1):
    y_RK2 = calc_RK2_next_y(x, y_RK2, h)
    x = round(x + h, 2)
    print(f"RK2: x_{i} = {x},      y_{i} = {y_RK2}")

print("\n----------------------TASK2:E----------------------")
x = 0.
y_EM = 1.
y_RK2 = 1.
h_EM = 0.05
h_RK2 = 0.1
for i in range(1, iter + 1):
    y_EM = calc_EM_next_y(x, y_EM, h_EM)
    y_EM = calc_EM_next_y(round(x + h_EM), y_EM, h_EM)
    y_RK2 = calc_RK2_next_y(x, y_RK2, h_RK2)
    print(f"-- For x_{i} = {round(x + h_RK2, 2)}:")
    print(f"EM:     y_{i} = {y_EM}")
    print(f"RK2:    y_{i} = {y_RK2}")
    x = round(x + h_RK2, 2)
    y_exact = exact_res(x)
    print(f"EM  error is: {abs(y_exact - y_EM)}")
    print(f"RK2 error is: {abs(y_exact - y_RK2)}\n")
print("---------------------------------------------------")
```

שאלה 3

נפתור את בעיית המודל: $\frac{d^2y}{dx^2} - \left(1 - \frac{x}{5}\right)y = x$ כאשר $y(1) = 2,\ \ y(3) = -1$ בעזרת קוד *python* אותו כתבנו המבצע את שיטת ריילי-ריץ. נבצע את הקירוב ל- $y(x)$ בעזרת $u(x) = \sum_{i=0}^{4} c_i N_i$.

```python
goal_func = lambda x: 1 - (x / 5)
y_start = 2.
y_end = -1.
approximation_range = [1., 3.]
num_of_partitions = 5
```

תחילה, חילקנו את הקטע ל5 מקטעים והגדרנו את אברי הבסיס $N_0, \dots, N_4$ להיות $N_i = \begin{cases} \frac{x - x_{i-1}}{h}, & [x_{i-1}, x_i] \\ \frac{x_{i+1} - x}{h}, & [x_i, x_{i+1}] \end{cases}$

כאשר $h = \frac{approxamation\_range}{num\_of\_partitions - 1} = \frac{3-1}{4} = 0.5$.

```python
X = np.linspace(approximation_range[0], approximation_range[1], num_of_partitions)
h = X[1] - X[0]
def n0(x):
    if x >= X[0] and x <= X[1]: return (X[1] - x) / h
    return 0.
def n1(x):
    if x >= X[0] and x <= X[1]: return (x - X[0]) / h
    if x >= X[1] and x <= X[2]: return (X[2] - x) / h
    return 0.
def n2(x):
    if x >= X[1] and x <= X[2]: return (x - X[1]) / h
    if x >= X[2] and x <= X[3]: return (X[3] - x) / h
    return 0.
def n3(x):
    if x >= X[2] and x <= X[3]: return (x - X[2]) / h
    if x >= X[3] and x <= X[4]: return (X[4] - x) / h
    return 0.
def n4(x):
    if x >= X[3] and x <= X[4]: return (x - X[3]) / h
    return 0.
N = [n0, n1, n2, n3, n4]
```

כעת, נגזור כל N_i ונשמור במערך לצורך קריאה מהירה בהמשך כאשר $N_i' = \begin{cases} \frac{1}{h}, & [x_{i-1}, x_i] \\ \frac{-1}{h}, & [x_i, x_{i+1}] \end{cases}$.

```python
def n0_prime(x):
    if x >= X[0] and x <= X[1]: return -1/h
    return 0.
def n1_prime(x):
    if x >= X[0] and x <= X[1]: return 1/h
    if x >= X[1] and x <= X[2]: return -1/h
    return 0.
def n2_prime(x):
    if x >= X[1] and x <= X[2]: return 1/h
    if x >= X[2] and x <= X[3]: return -1/h
    return 0.
def n3_prime(x):
    if x >= X[2] and x <= X[3]: return 1/h
    if x >= X[3] and x <= X[4]: return -1/h
    return 0.
def n4_prime(x):
    if x >= X[3] and x <= X[4]: return 1/h
    return 0.
N_prime = [n0_prime, n1_prime, n2_prime, n3_prime, n4_prime]
    return 0.
N = [n0, n1, n2, n3, n4]
```

עתה נוכל לחשב את מטריצת המקדמים A כך שכל שורה במטריצה מהצורה:

$$A_{i,j} = \int_1^3 \left[N_i(x)'N_j(x)' + \left(1 - \frac{x}{5}\right) N_i(x)N_j(x)\right] dx$$

השתמשנו ב$scipy.integrate.quad$ לשם חישובי האינטגרל.

```
A = []
for i in range(1, num_of_partitions - 1):
    row_i = []
    for j in range(num_of_partitions):
        func_i_j = lambda x: N_prime[i](x) * N_prime[j](x) + goal_func(x) * N[i](x) * N[j](x)
        integral, *_ = quad(func_i_j, approximation_range[0], approximation_range[1])
        row_i.append(integral)
    A.append(row_i)
A = np.asmatrix(A)
print("\nCoefficient matrix A:")
print(A)
```

התוצאה המתקבלת בעבור מטריצת A הינה:

```
Coefficient matrix A:
[[-1.9375      4.23333333 -1.94583333  0.          0.        ]
 [ 0.         -1.94583333  4.2        -1.95416667  0.        ]
 [ 0.          0.         -1.95416667  4.16666667 -1.9625    ]]
```

נחשב כעת את ווקטור המקדמים הבלתי תלויים B:

```
B = []
for i in range(1, num_of_partitions - 1):
    integral, *_ = quad(lambda x: x*N[i](x), approximation_range[0], approximation_range[1])
    B.append(-integral)
B = np.asarray(B)
print("\nDependent variables B:")
print(B)
```

התוצאה המתקבלת בעבור ווקטור B הינה:

```
Dependent variables B:
[-0.75 -1.   -1.25]
```

כעת נרצה לפתור את מערכת המשוואות $A * C = B$. כלומר:

$$\begin{bmatrix} -1.9375 & 4.23333333 & -1.94583333 & 0. & 0. \\ 0. & -1.94583333 & 4.2 & -1.95416667 & 0. \\ 0. & 0. & -1.95416667 & 4.16666667 & -1.9625 \end{bmatrix} * \begin{bmatrix} y(1) = 2.0 \\ c_1 \\ c_2 \\ c_3 \\ y(3) = -1.0 \end{bmatrix} = \begin{bmatrix} -0.75 \\ -1.0 \\ -1.25 \end{bmatrix}$$

נבחר לחסר מ B_1 את $2.0 * -1.9375$ ומ B_2 את $-1.0 * -1.9625$ ונקבל מערכת משוואות שקולה $A^* * C^* = B^*$:

$$\begin{bmatrix} 4.23333333 & -1.94583333 & 0. \\ -1.94583333 & 4.2 & -1.95416667 \\ 0. & -1.95416667 & 4.16666667 \end{bmatrix} * \begin{bmatrix} c_1 \\ c_2 \\ c_3 \end{bmatrix} = \begin{bmatrix} -0.75 + 1.9375 * 2.0 \\ -1.0 \\ -1.25 - 1.9625 * 1.0 \end{bmatrix}$$

```
B[0] = B[0] - (A[0,0] * y_start)
B[-1] = B[-1] - (A[-1,-1] * y_end)
A = A[:, 1:-1]
```

פתרנו את מערכת המשוואות הנתונה בעזרת $numpy.linalg.solve$.

```
C = np.linalg.solve(A, B)
C = np.asarray([y_start, *C, y_end])

print("\nC's:")
print(C)
```

המקדמים $c_0, \ldots, c_4$ המתקבלים מהקוד הינם:

```
C's:
[ 2.          0.53227693 -0.44797995 -0.9811026  -1.        ]
```

כלומר קבלנו כי:

$$\begin{bmatrix} c_0 \\ c_1 \\ c_2 \\ c_3 \\ c_4 \end{bmatrix} = \begin{bmatrix} 2.0 \\ 0.53227693 \\ -0.44797995 \\ -0.9811026 \\ -1.0 \end{bmatrix}$$

והקירוב $u(x) \approx y(x)$ המתקבל הינו:

$$\boldsymbol{u(x) = \sum_{i=0}^{4} c_i N_i(x) =}$$
$$\boldsymbol{= 2.0 * N_0(x) + 0.53227693 * N_1(x) - 0.44797995 * N_2(x) - 0.9811026 * N_3(x) - 1.0 * N_4(x)}$$