Numerical Analysis - Midterm

שאלה 1:

```
import numpy as np
# 1. A function that constructs the coefficients required for evaluating
the Newton form of the interpolation polynomial
def divdiff(x,y):
    n = x.size
    f = np.zeros((n,n),float)
    c = np.empty((1,n))
    for i in range(n):
        f[i,0] = y[0,i]
    for j in range(1,n):
        for i in range(j,n):
            f[i,j] = float(f[i,j-1] -f[i-1,j-1])/float(x[0,i]-x[0,i-j])
    for i in range(n):
        c[0,i] = f[i,i]
    return c
# 2. A function which evaluates Newton's interpolation polynomial at a
new set of points
def interpnewt(c,x,xnew):

    def newton_polynomial(c,x,r):
        n = x.size - 1
        p = c[0,n]
        for i in range(1,n + 1):
            p = c[0,n - i] + (r - x[0,n - i])*p
        return p
    m = xnew.size
    ynew = np.empty((1,m))
    for i in range(m):
        ynew[0,i] = newton_polynomial(c,x,xnew[0,i])
    return ynew
```

שאלה 2:

הסבר על החישוב: בבחירת n כלשהו, ניתן לקבל את הנקודות השוות מרחק ע"י חילוק הקטע ל- n נקודות, כלומר $\frac{(b-a)}{n}$ הוא ההפרש בין כל 2 נקודות. ואת נקודות צ'בישב ע"י הנוסחא:

$$x_k = \frac{1}{2}(a+b) + \frac{1}{2}(b-a)cos(\frac{2k-1}{2n}\pi) \ ,k \in \{1,...,n\}$$

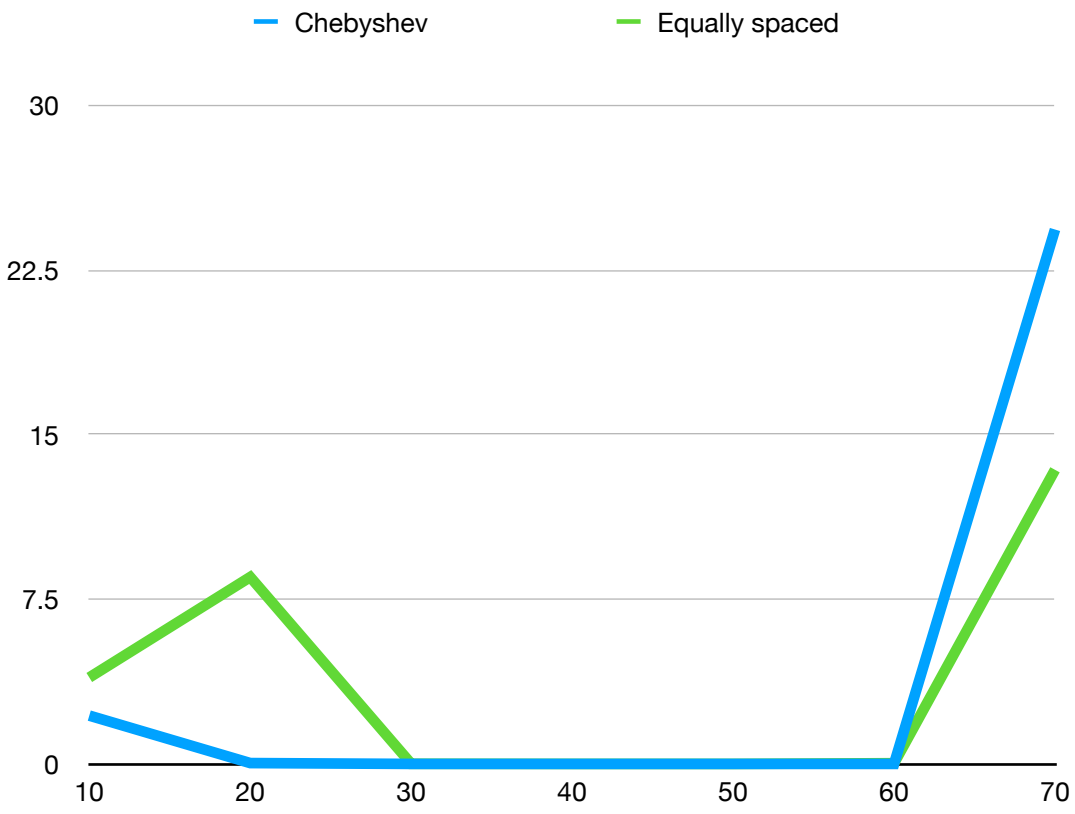
בשימוש בפונקציות משאלה אחת ופונקציות שימושיות של numpy, בניתי פולינומי אינטרפולציה מתאימים. הכנסתי מערך של 100 נקודות רנדומליות בתחום הנתון, הצבתי בפולינום האינטרפוציה שמתקבל בכל אחת מהשיטות (נקודות שוות מרחק, ונקודות צ'פישב).

הקוד:

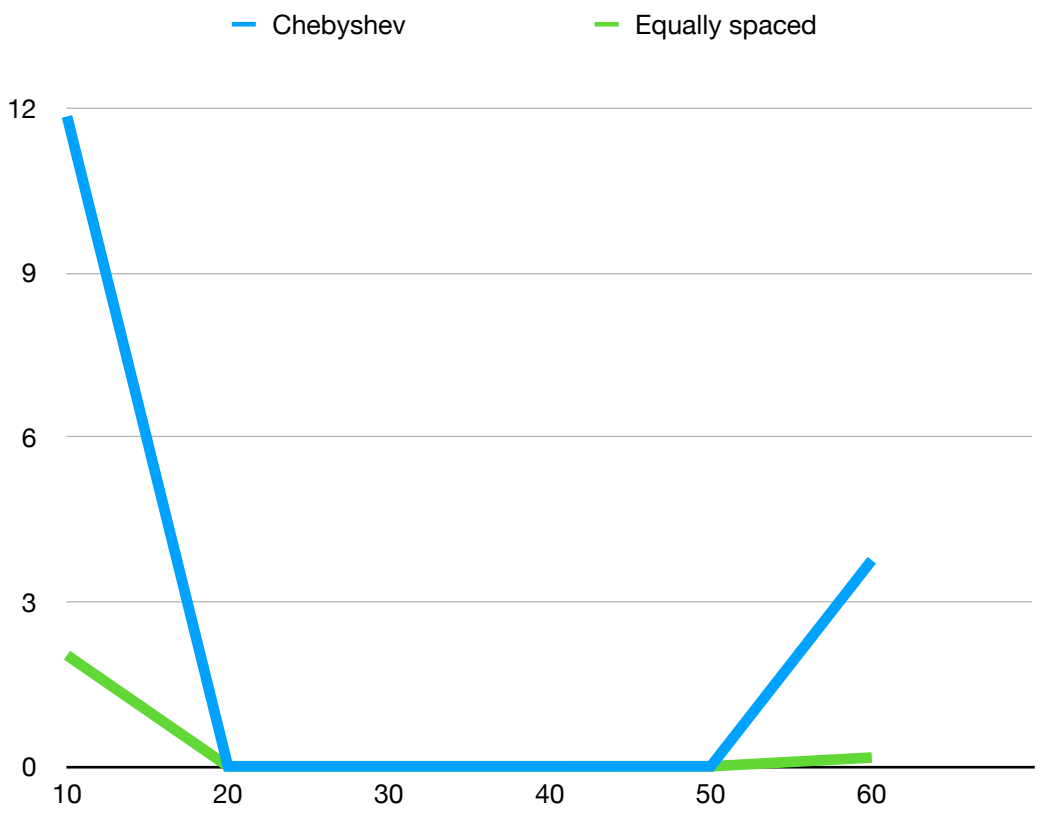
```
import numpy as np
def Q2(f , n , r , x0 , xn):
    fr = f(r)
    print(n)
    # equally spaced points:
    x_eq = np.linspace(x0, xn, n)
    x_eq = x_eq.reshape(1,n)
    y_eq = f(x_eq).reshape(1, n)
    c_eq = divdiff(x_eq, y_eq)
    P_eq = interpnewt(c_eq, x_eq, r)
    #print(P_eq)
    e_eq = np.absolute(fr - P_eq)
    # Chebyshev points:
    k = np.arange(n)
    k = k + 1
    x_ch = 0.5 * (xn + x0) + 0.5 * (xn - x0) * (np.cos((2 *
k - 1) * np.pi / (2 * n)))
    x_ch = x_ch.reshape(1, n)
    y_ch = (f(x_ch)).reshape(1, n)
    c_ch = divdiff(x_ch, y_ch)
    P_ch = interpnewt(c_ch, x_ch, r)
    e_ch = np.absolute(fr - P_ch)
    return np.amax(e_eq), np.amax(e_ch)

    m = xnew.size
    ynew = np.empty((1,m))
    for i in range(m):
        ynew[0,i] = newton_polynomial(c,x,xnew[0,i])
    return ynew
```

הגרפים:

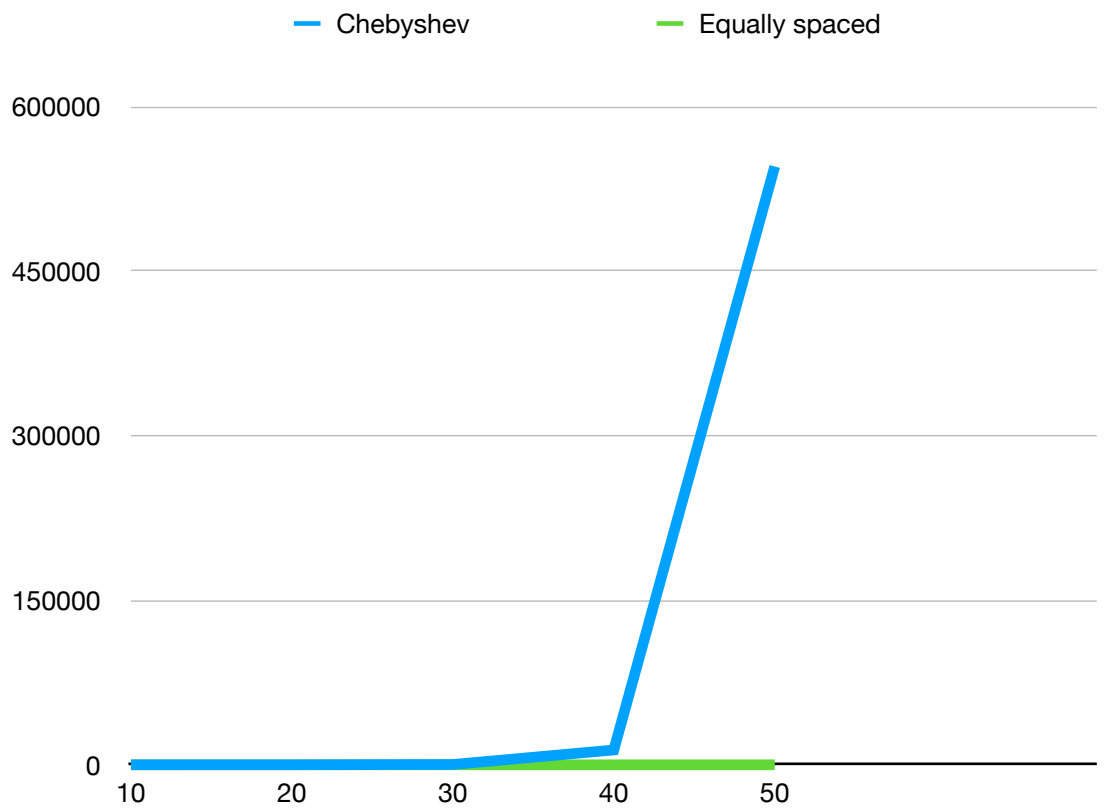
1. עבור הפונקציה $f(x) = cos(5x), x \in [0,2\pi]$



2. עבור הפונקציה $f(x) = e^x$ בתחום $x \in [0,10]$.

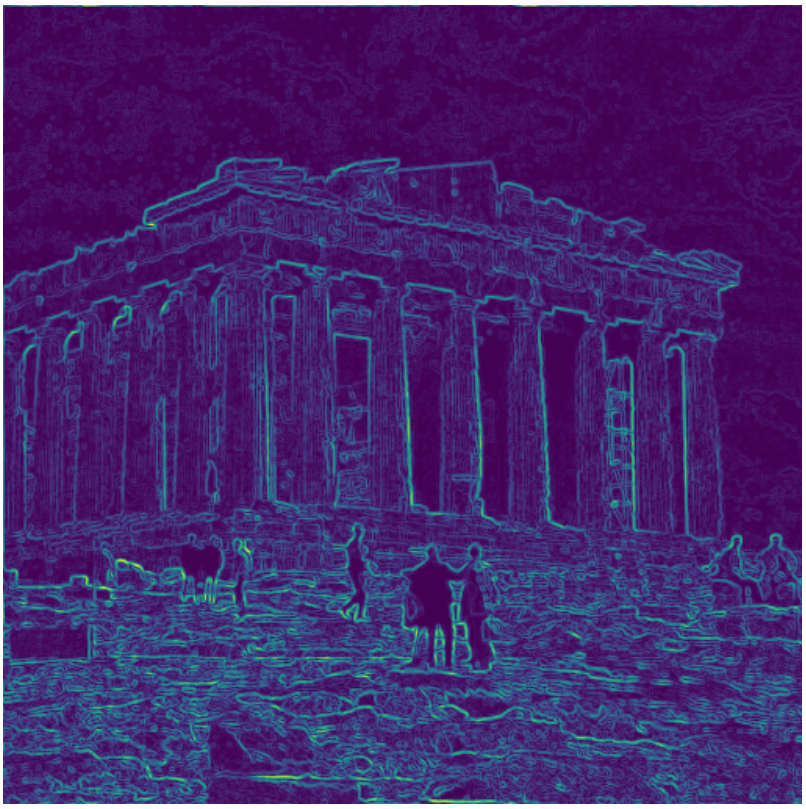


3. עבור הפונקציה $f(x) = \frac{1}{x^2+1}$ בתחום $x \in [-5,5]$.



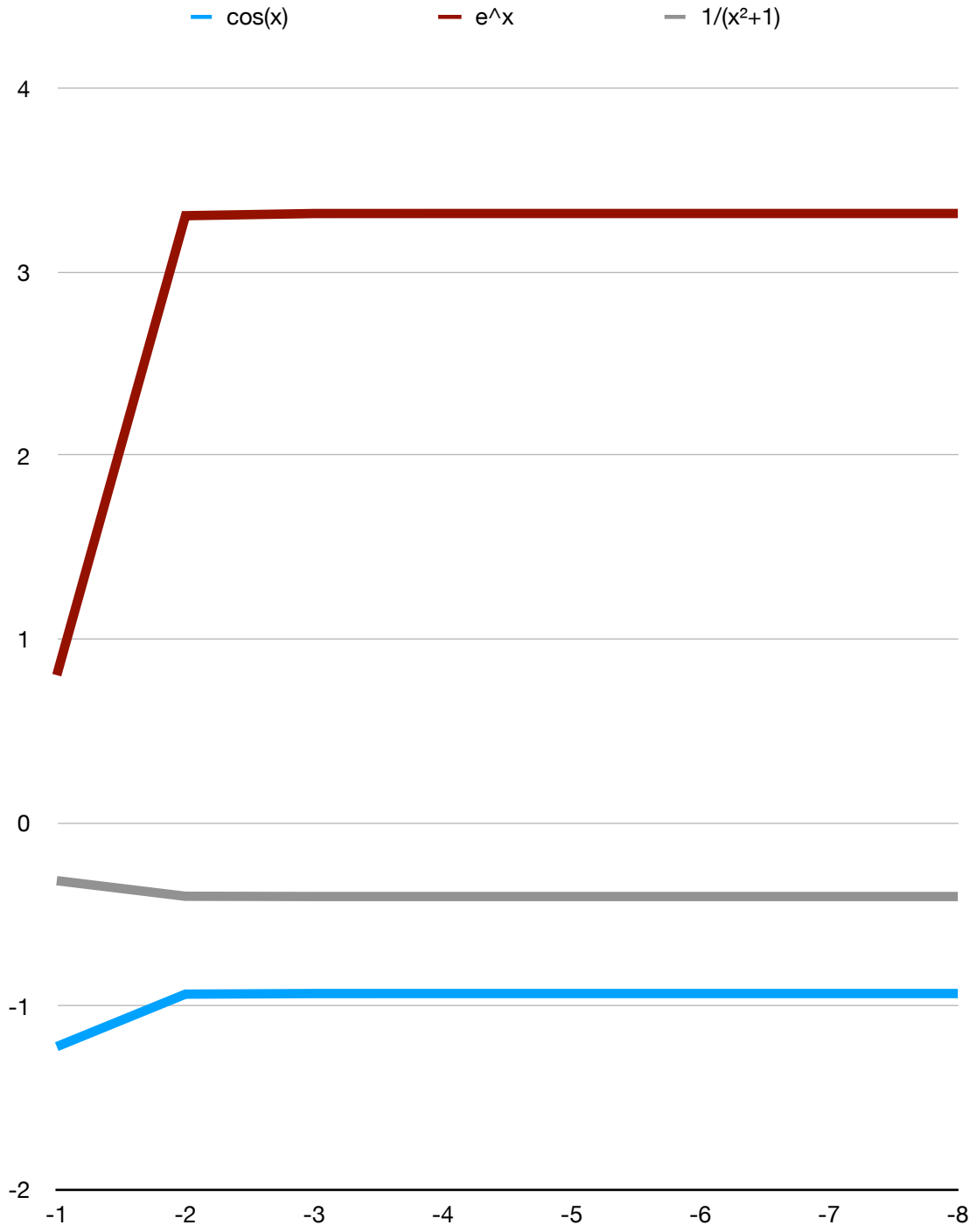
שאלה 3:

אנחנו רואים שינוי צבעי התמונה לצבעי תשליל, וקווי מיתר חדים יותר. זאת קורה בגלל שהאופרטור נותן ערך גבוה לפיקסל שיש הפרש גדול בערכים של הפיקסלים בסביבה שלו, אופקית ואנכית. אפשר לאתר ככה קווי מתאר של עצמים בתמונה כי סביר להניח שיש הבדל בין פיקסל בתוך האובייקט שמצולם לבין פיקסל מחוץ לאובייקט.



שאלה 4:

בגרפים נוכל לראות כי כאשר h גדול יחסית אנחנו מקבלים שגיאה משמעותית למרות שאנחנו מורידים את השגיאה הרנדומלית (שהערך המוחלט שלה קטן מה $0.5 \cdot 10^{-5}$), וזאת כיוון שראינו בשיעור ובתרגול, כי השגיאה תלויה ב h (יותר נכון בMh^2 כאשר $|f'''(x)| < M$ $\forall x,$) ולכן כאשר השגיאה אינה בסדר גודל המתאים אנחנו מתרחקים מהתוצאה האמיתית של הנגזרת. (למשל כאשר $h = 1$ במקרה של $cos(x)$ כאשר הנגזרת היא $-sin(x)$ אנחנו מקבלים שגיאה משמעותית שאף "עובר" את החסם של הנגזרת) וככל שאנחנו מקטינים את h בחסם מתקרב לתחום השגיאה שמוגדרת ולכן התוצאה ניהיית יותר מדוייקת (למשל כאשר $h = 10^{-3}$ בריבוע אנחנו נקבל $h^2 = 10^{-6}$ מספר זה כבר נמצא בתחום השגאה). הגרפים שיוצאים לפי ההדפסות:



הקוד שרשמתי בכדי לקבל את התוצאות:

```
import numpy as np
def ThreePoint(f,x,h):
    error = np.random.uniform(-0.000005,0.000005)
    dx = 0.5*(1/h)*(-3*f(x)+4*f(x+h)-f(x+2*h))
    return dx - error

f = lambda x: 1 / (x**2 + 1)
k = np.arange(8)
k = -k
h = np.float_power(10,k)
x = 1.2
for h in h:
    print("Dx(exp):", ThreePoint(np.exp,x,h) ,h)
    print("Dx(cos(x):",ThreePoint(np.cos,x,h) ,h)
    print("Dx((x^2 + 1)^(-1):",ThreePoint(f,x,h) ,h)
```