שיטות חישוביות – עבודת מחשב 3:

מגיש:

ניר שניידר 316098052

שאלה 1 – שחזור באמצעות אינטרפולציית לגראנז:

$$\phi(\theta)=\frac{q^{+}}{4\pi r^{+}(\theta)}+\frac{q^{-}}{4\pi r^{-}(\theta)}\ ,\ r^{\pm}(\theta)=\sqrt{[rcos(\theta)]^{2}+\left[rsin(\theta)\mp\frac{\delta}{2}\right]^{2}}$$

$$q^{+}=\ 2*(1+6+0+9+8+0+5+2)=62$$

$$\begin{aligned}q^{-}=&-(3+(3^{2}\%10)+1+(1^{2}\%10)+6+(6^{2}\%10)+0+(0^{2}\%10)+9\\&+(9^{2}\%10)+8+(8^{2}\%10)+0+(0^{2}\%10)+5+(5^{2}\%10)\\&=-(3+9+1+1+6+6+0+0+9+1+8+4+0+0+5\\&+5=-58\end{aligned}$$

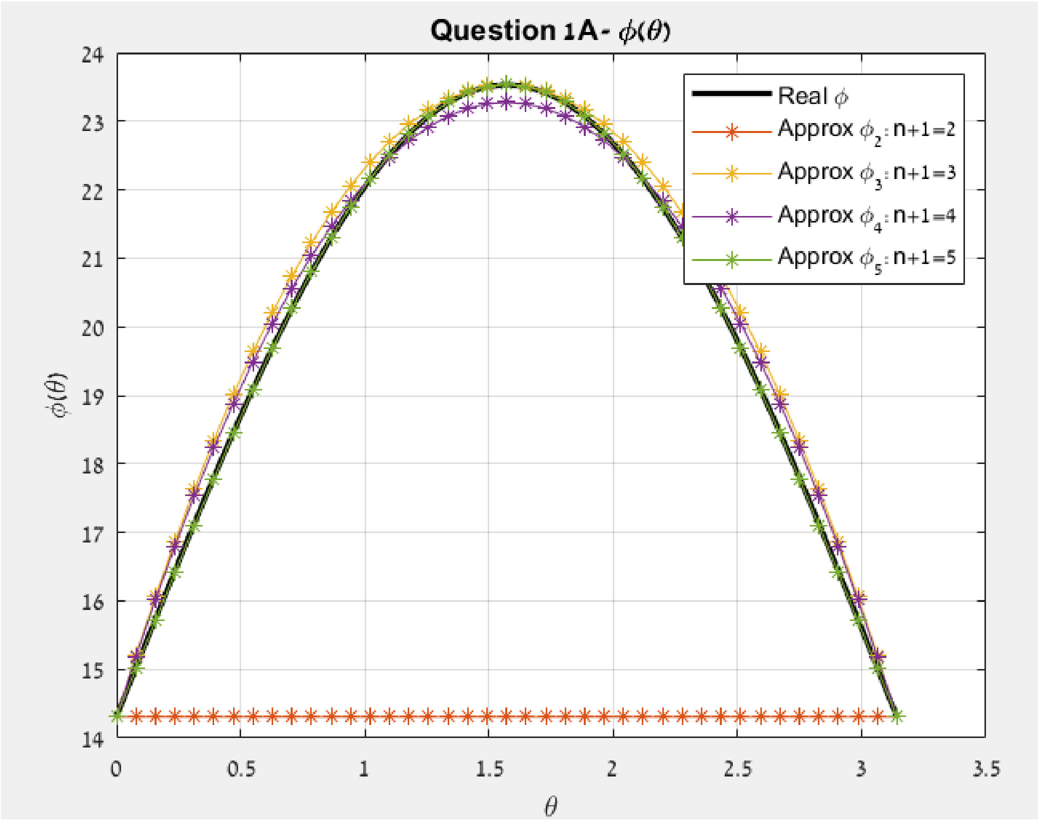
א) כפי שראינו בכיתה, אינטרפולציה בעזרת פולינומי לגראנז':

בהינתן סדרה של N+1 נקודות $\{x_i\}_{i=0}^{N}$ *הנקראות צמתים ו-N+1 נקודות מתאימות (הערכים האמיתיים של y)* $\{y=f(x_i)\}_{i=0}^{N}$ *אזי פולינום האינטרפולציה לפי צורת לגראנז' ינתן ע"י:*

$$L_N(x)=\sum_{i=0}^{N}y_i l_i(x)\quad,\quad l_i(x)=\prod_{j=0\neq i}^{N}\frac{x-x_j}{x_i-x_j}$$

כאשר $L_N(x)$ *הוא פולינום האינטרפולציה (הקירוב של f(x)) ו-*$l_i(x)$ *הם פולינומי לגראנז'.*

להלן הגרף הנדרש:



נשים לב שככל שמספר נקודות הדגימה גדול יותר כך נקבל קירוס טוב יותר לשחזור הפוטנציאל.

כאשר נעבוד עם יותר נקודות דגימה נוכל לשחזר את הפונקציה בדיוק גדול יותר מכיוון שיש לנו יותר מידע בתחום, כלומר נגזרות מסדר גבוה יותר חסומות כדי שתופעת רוגנה לא תופיע.

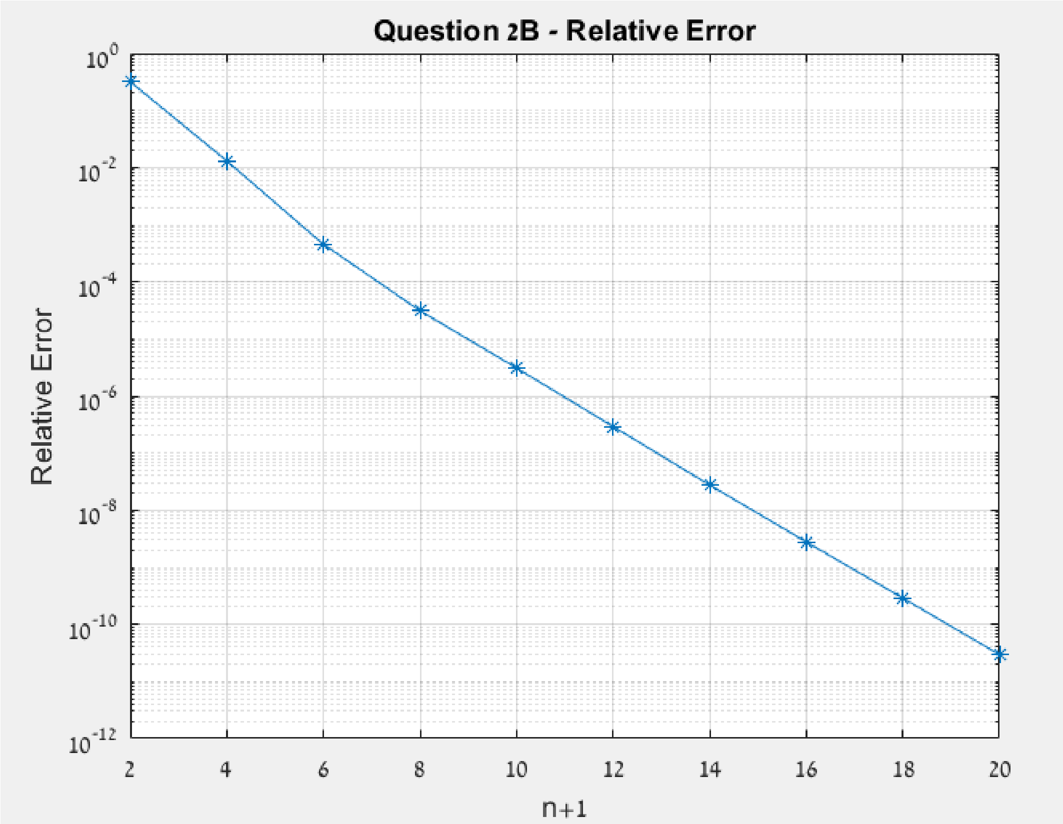
עבור *n=1* → *n+1=2*. השגיאה גדולה ביחס לשאר מכיוון שעבור 2 נקודות דגימה בלבד נקבל משוואה ליניארית (סדר 1).

ב)

$$\varepsilon(\phi)=\sqrt{\sum_{i=0}^{\bar{n}}[\bar{\phi}(\bar{\theta}_i)-\phi(\bar{\theta}_i)]^2}\,/\,\sqrt{\sum_{i=0}^{\bar{n}}[\phi(\bar{\theta}_i)]^2}$$

כאשר $n+1\in[2,4,6,\dots,20]$

להלן הגרף שהתקבל:

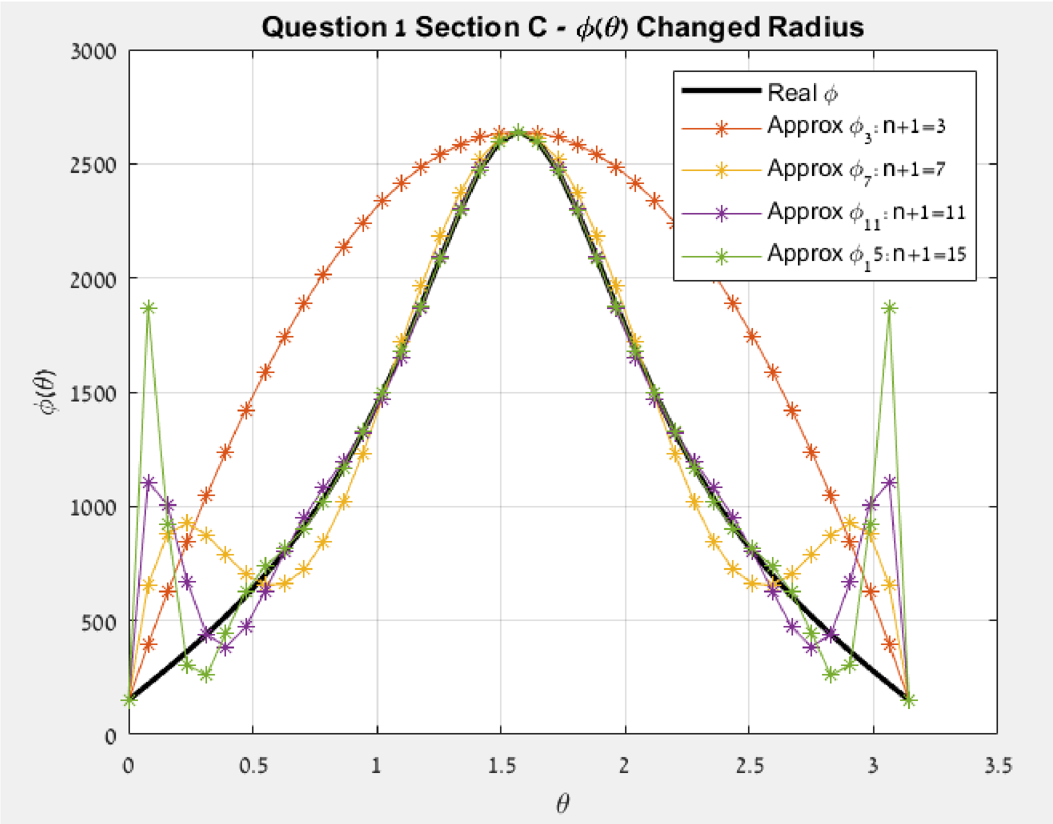


עבור סדר פולינום גבוה יותר נקבל שגיאה יחסית קטנה יותר. זה תקיים כי עבור סדר פולינום גבוה יותר הקירוב לפוטנציאל מדויק יותר. הגרף מונוטוני יורד, משמע שעבור מספר נקודות דגימה גדול יותר סדר הפולינום גדל ועקב כך השגיאה היחסית קטנה.

עבור מספר נקודות מספיק גדול, השגיאה בין הפונקציה לקירוב בקצוות תהיה גדולה.

ג) $r=4cm$, $n+1=3,7,11,15$.

להלן הגרף שהתקבל:

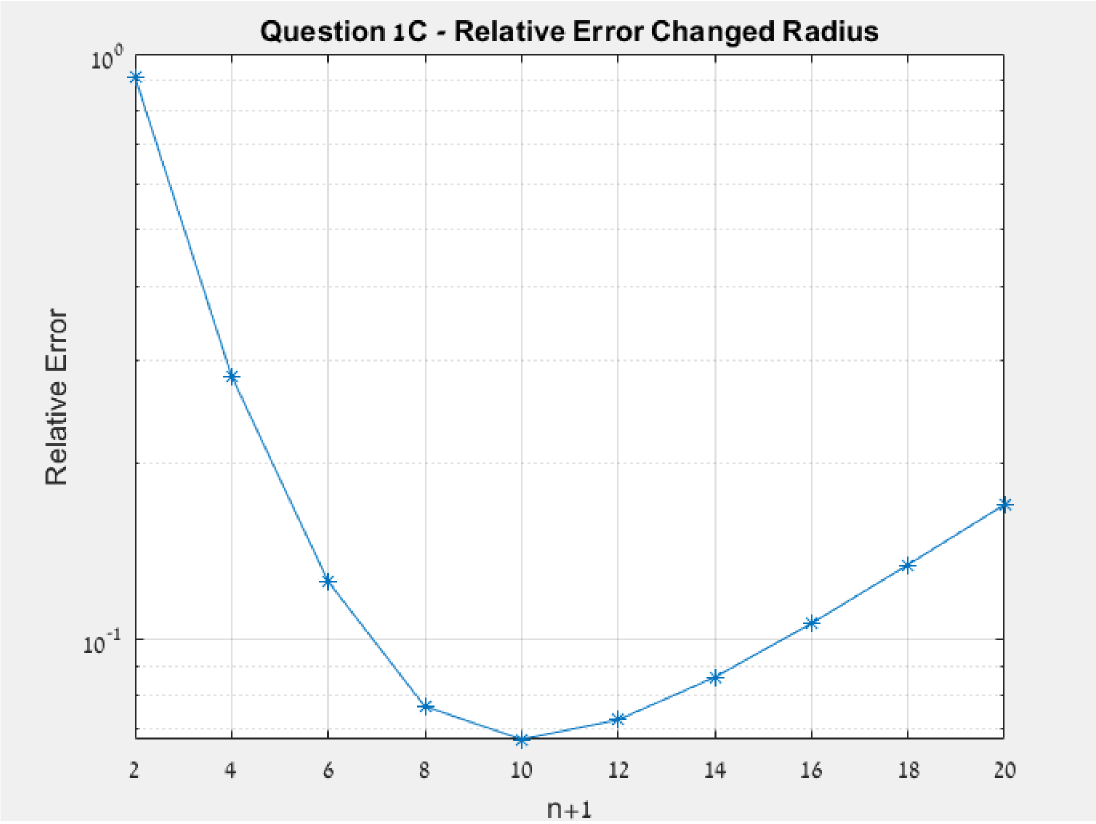


השגיאה היחסית בקצוות יותר גדולה עבור סדר גדול יותר – תופעת רונגה. בשל המרחק הקטן של נקודות המדידה למטענים, ככל שקצב הדגימה גדל כך השגיאה גדלה.

הגדל סדר האינטרפולציה לאו דווקא גורר פונקציה מדויקת יותר מכיוון ששגיאת הקצוות של הקטע גדלה. נשים לב לכך שעבור הגדלת מספר נקודות הדגימה מקנה קירוב פונקציה טוב יותר למעט בקצוות (שם ניתן לראות קפיצות).

לעומת סעיף א', כאן הבדל הרדיוס מניב חסם נגזרות גדול יותר (פונקציה יותר תלולה). כלומר ניתן להבחין שהשגיאה גדלה עבור סדר נמוך מהפוטנציאל בסעיף א'. כאן הפונקציה משתנה מהר יותר מבסעיף א'.

להלן גרף השגיאה היחסית:



נקבל מינימום ב-$n+1=10$. החל מ-$n=10$ נקבל עלייה בגרף.
ישנה מגמה של גדילה בשגיאה (תופעת רונגה). עבור סדר אינרפולציה גדול מספיק וחסם נגזרות גדול גם כן נקבל שגיאה בקצוות גדולה.
בסעיף א' הפוטנציאל שהתקבל היה מונוטוני יורד מכיוון ששגיאת הנגזרות הייתה קטנה בהשוואה לכאן. עקב כך נדרש מאיתנו סדר אינטרפולציה גדול באופן משמעותי לטובת עלייה של גרף השגיאה היחסית.

ד) כפי שראינו בכיתה, בחירת נקודות לפי שורשי פולינום צ'בישב היא כדלקמן:

נוחסת קושי לשגיאה :

$$\varepsilon_N(x)=\frac{f^{(N+1)}(\xi_x)}{(n+1)!}\prod_{i=0}^{N}(x-x_i)\ \ ,\ \ \min(x_0,x)\le\xi_x\le\max(x_N,x)$$

ממבנה השגיאה (נוסחת קושי) ניתן לראות כי היא תלויה בפולינום המוני $\prod_{i=0}^{N}(x-x_i)$*, לכן ניתן למזער את החסם העליון על השגיאה ע"י בחירת נקודות* $\{x_i\}_{i=0}^{N}$ *כלשהן.*

בהרצאה הוכח כי בקטע [-1.1] עבור פולינום מוני P(x) ממעלה N מתקיים $\max_{x\in[-1,1]}|P(x)|\ge 2^{1-N}$ *והשוויון יתקיים עבור* $P(x)=2^{-N}T_{N+1}(x)=\prod_{i=0}^{N}(x-x_i)$ *כאשר* $T_N(x)$ *הוא פולינום צ'בישב ממעלה N המוגדר באופן הבא:*

$$T_0(x)=1$$
$$T_1(x)=x$$
$$T_N(x)=2xT_{N-1}(x)-T_{N-2}(x)\ \ ,\qquad x\in[-1,1]$$

כלומר הנקודות $\{x_i\}_{i=0}^{N}$ *הן השורשים של* $2^{-N}T_{N+1}(x)$*. ידוע שמתקיים:*

$$T_{N+1}\left(\cos\left(\frac{2i+1}{2N+2}\pi\right)\right)=0\ \ ,i=0,\dots,N\to x_i=\cos\left(\frac{2i+1}{2N+2}\pi\right)\in[-1,1]$$

מה נעשה כאשר תחום ערכי x אינו [-1,1]?

עבור $x\in[a,b]$*, נבצע העתקה לינארית מ-* $t\in[-1,1]$ *ל-* $x\in[a,b]$ *באופן הבא:*

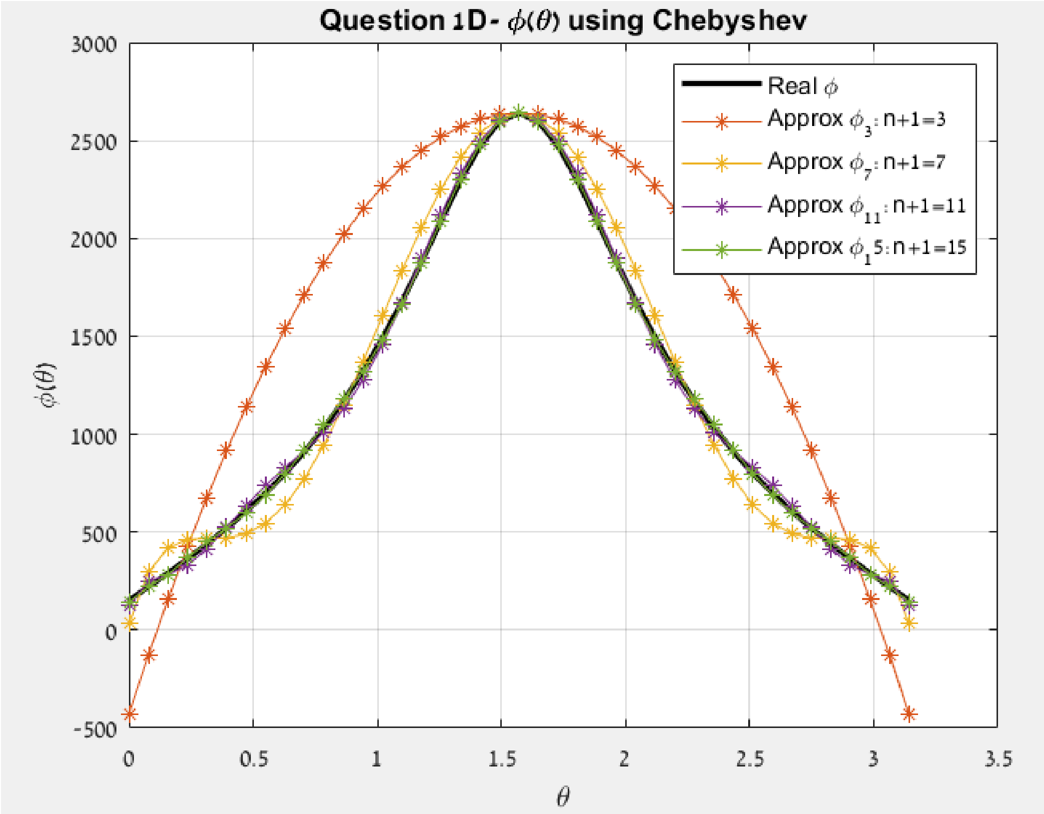
$$x=\frac{b+a}{2}+\frac{(b-a)t}{2}$$

$\bar{x}_j=cos\left(\frac{\pi(2j+1)}{2n}\right)\ \ ,\ \ j=0,\dots,n-1$ – השורשים.

מכיוון שערכי הדגימות לא בתחום *[-1,1]* נבצע את ההעתקה הלינארית שראינה בכיתה ומוזמכרת לעיל: $m'=\frac{b+a}{2}+\frac{(b-a)x_j}{2}$.

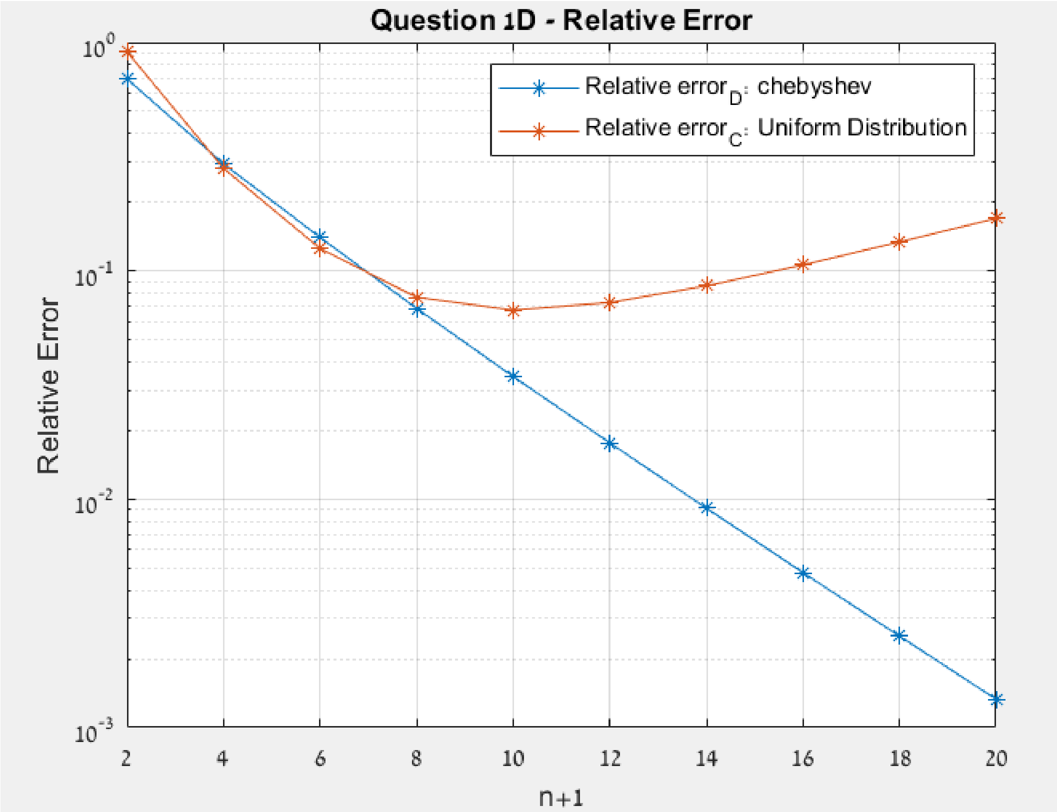
כעת $m'\in[0,\pi]$

להלן הגרף שמתקבל:



כפי שציפינו, עבור בחירת נקודות לפי שורשי פולינומי צ'בישב, הדיוק בשחזור הפונקציה גדל ככל שמספר הדגימות גדל. בנוסף, בשיטה זו תוקן תופעת השגיאה בקצוות.

להלן גרף השגיאה היחסית:



בהתאם לציפייה, שורשי פולינומי צ'בישב הינה הבחירה הטובה ביותר של נקודות האינטרפולציה מבחינת מינימום חסם השגיאה.

$h = \frac{b-a}{n}$, *n+1* נקודות. נקבל:

$$|r_n(x)| \le \frac{1}{(n+1)!}\frac{h^{n+1}}{4}n!\, max_{\xi_x \in [a,b]}\left|f^{(n+1)}(\xi_x)\right|$$

$$= \frac{h^{n+1}}{4(n+1)} max_{\overline{\epsilon_x} \in [a,b]}\left|f^{(n+1)}(\xi_x)\right|$$

נעתיק את שורשי צ'בישב לתחום ואז השגיאה שנקבל מתקיימת:

$$|r_n(x)| \le \frac{1}{2^n(n+1)!}\frac{|b-a|^{n+1}}{2^{n+1}} max_{\xi_x \in [a,b]}\left|f^{(n+1)}(\xi_x)\right|$$

כפי שציפינו, עבור בחירת נקודות ע"י שורשי פולינומי צ'בישב קיבלנו תיקון לתופעת רונגה שהופיעה עבור בחירת נקודות אחידות. כתוצאה מכך תוצאות השגיאה היחסית השתפרו.

שאלה 2 – שחזור מדידות בשיטת *Least Square*:

א) θ_j – נקודות המדידה. $\phi_j = \phi(\theta_j)$ – הערכים המדודים.

נניח שהפוטנציאל נמדד ב-*n+1* נקודות.

$$f_1 = 1 \ , \qquad f_2 = \sin(\theta) \ , \qquad f_3 = \cos(\theta)$$

נרצה לפתור את המערכת הבאה:

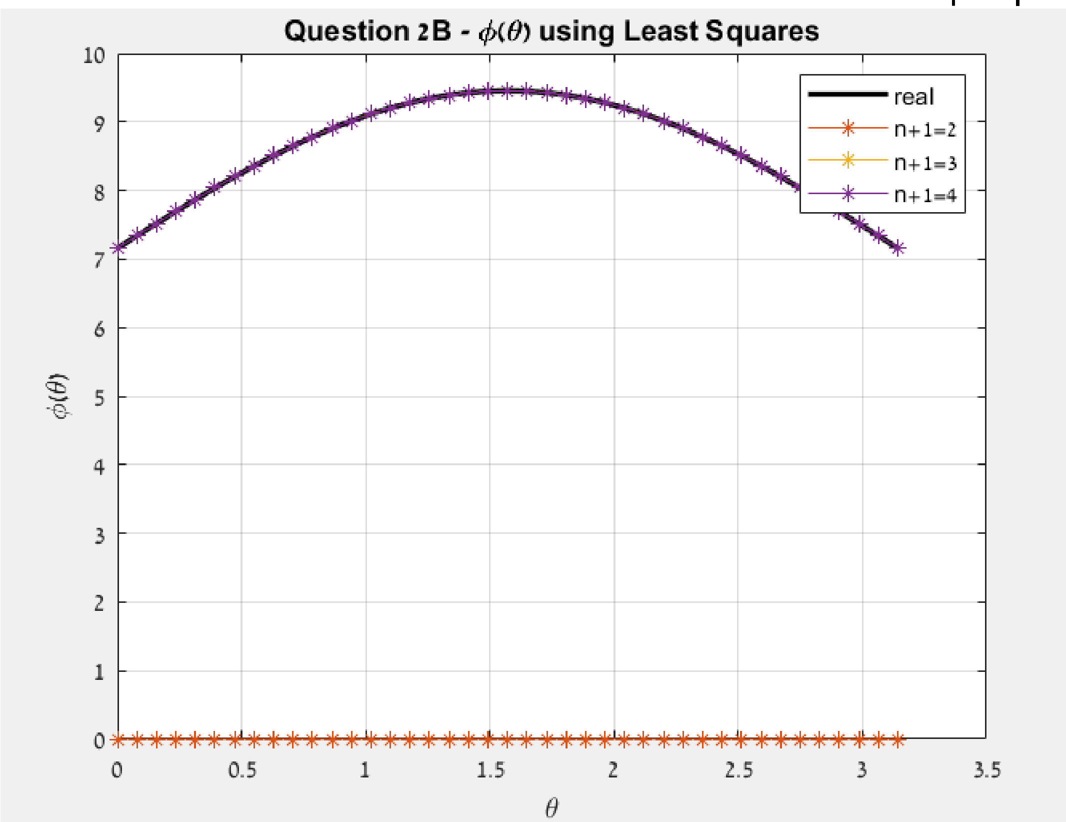
$$\begin{pmatrix} \sum_{i=0}^{n} 1*1 & \sum_{i=0}^{n} \sin(x_i)*1 & \sum_{i=0}^{n} \cos(x_i)*1 \\ \sum_{i=0}^{n} 1*\sin(x_i) & \sum_{i=0}^{n} \sin(x_i)*\sin(x_i) & \sum_{i=0}^{n} \cos(x_i)*\sin(x_i) \\ \sum_{i=0}^{n} 1*\cos(x_i) & \sum_{i=0}^{n} \cos(x_i)*\sin(x_i) & \sum_{i=0}^{n} \cos(x_i)*\cos(x_i) \end{pmatrix} \begin{pmatrix} \alpha \\ \beta \\ \gamma \end{pmatrix} = \begin{pmatrix} \sum_{i=0}^{n} \phi_i * 1 \\ \sum_{i=0}^{n} \phi_i * 1 \\ \sum_{i=0}^{n} \phi_i * 1 \end{pmatrix}$$

באמצעות גזירה של השגיאה והשוואה לאפס נוכל למצוא את המקדמים.

ב) להלן טבלת המקדמים שהתקבלה:

n+1	α	β	γ
2	0	0	0
3	7.159735371319002	2.295999208874903	4.440892098500626$* e^{16}$
4	7.159735371319000	2.294201952777143	2.011518462822549$* e^{16}$

להלן הגרף לאחר הצבת התוצאות:



למציאת המקדמים שיביאו למינימום ע"פ מדד שגיאה ריבועית נמצא ע"י שיטת LS באופן הבא: $\begin{pmatrix}\alpha\\ \beta\\ \gamma\end{pmatrix} = (A^T * A)^{-1} * A^T * b$.

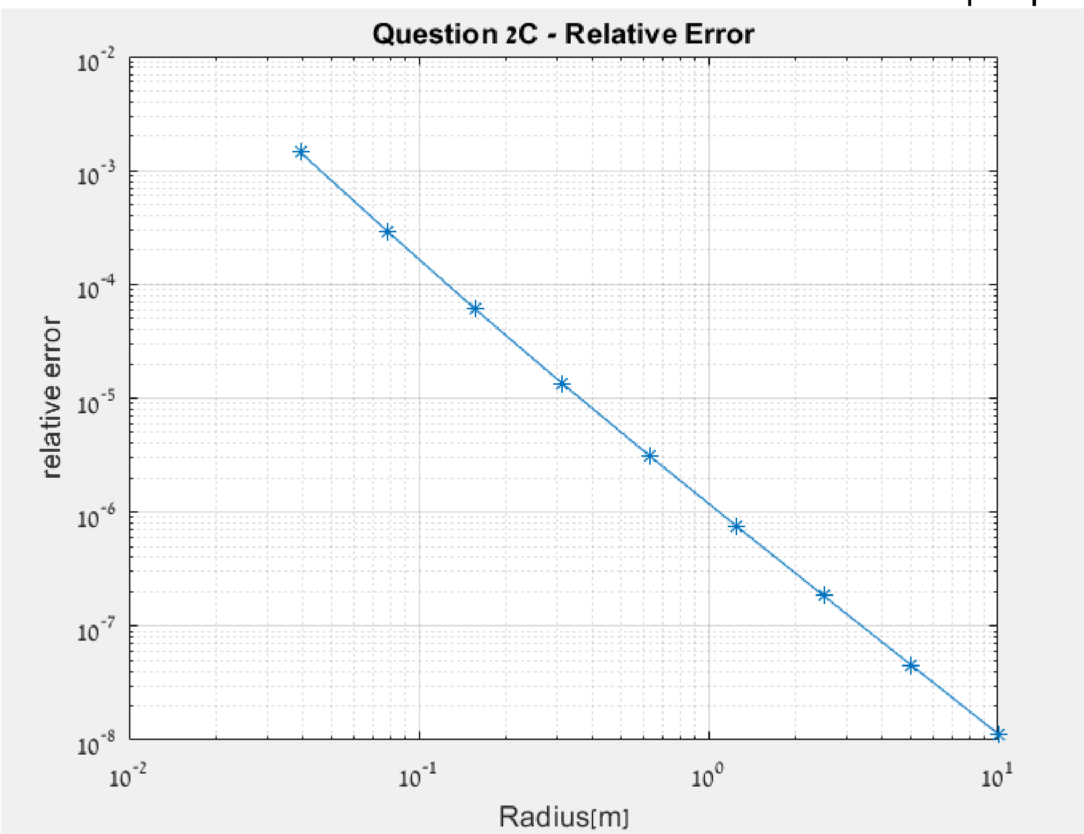
A – המטריצה שלנו. b – וקטור התוצאה.

עבור $n+1=2$ קיבלנו מערכת משוואת לינאריות של 2 משוואות ו-3 נעלמים ולכן לא ניתן למצוא את המקדמים לשחזור, ניתן לראות בגרף שנמצא את הפתרון הטריוויאלי ע"פ הפתרון המקורב ב-LS (ברור כי הפתרון אינו נכון).

בהתאם לציפייה מהנלמד בכיתה, עבור מערכת עם מספר משוואות יותר גדול נצליח לשחזר את הפוטנציאל באופן מלא ($n+1=3,4$) משום שע"י פונקציות ידועות מצאנו קירוב שמביאות לשחזור בעל דיוק רב.

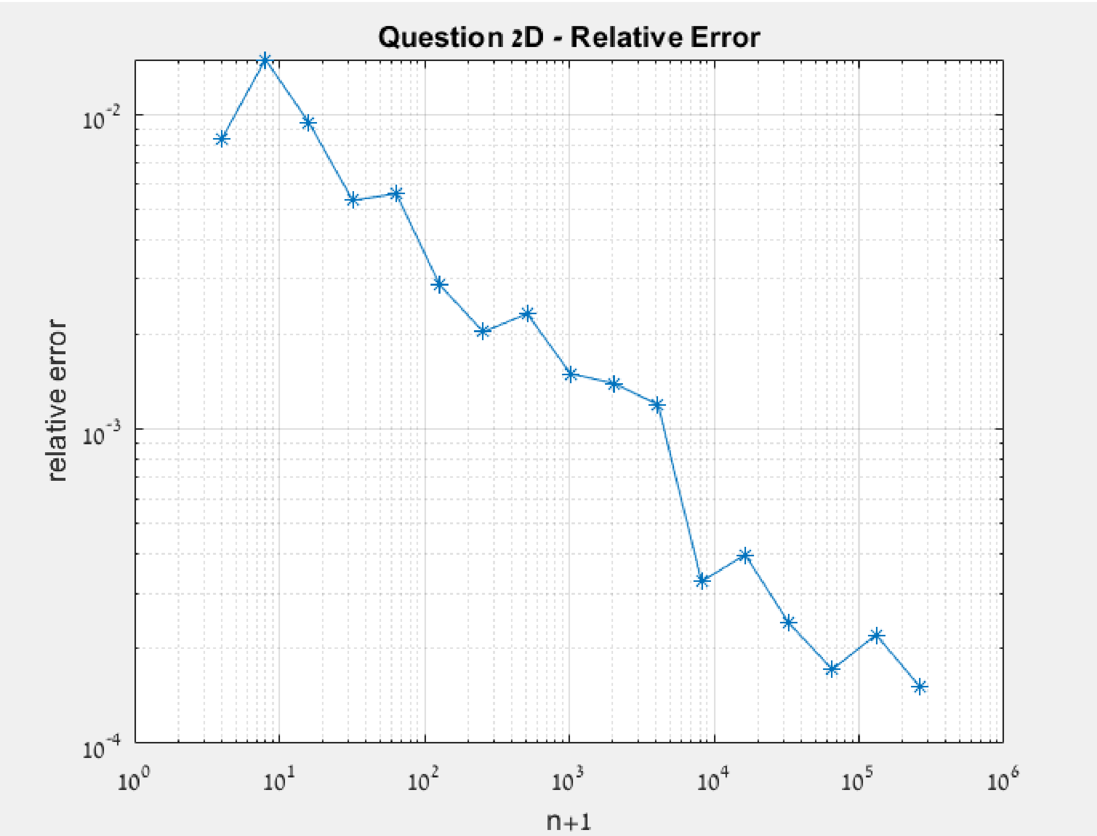
לעומת שאלה 1, כאן בשיטת LS ניתן לראות התכנסות מהירה יותר מאשר אינטרפולציית לגראנז'. שיטת LS בעלת קצב התכנסות גדול יותר (עבור אותו מספר נקודות דגימה) מכיוון שהפוטנציאל מתנהג בצורה לינארית של הפונקציות *sin(),1*. בנוסף היא מונוטונית עולה עד הנקודה $\frac{\pi}{2}$ ומעבר לנקודה זו היא מונוטונית יורדת. הרכיב של הקוסינוס לא בא לידי ביטוי מכיוון שהמקדמים שלו מתאפסים (בפועל ערך קטן עקב מגבלות המחשב). בסופו של דבר נקרב לפי סינוס וקבוע ונקבל התכנסות בעלת קצב גבוהה מאשר קירוב הפולינומים.

ג) להלן הגרף:



נשים לב שישנו יחס הפוך בין גודל הרדיוס לגודל השגיאה. כלומר, נקבל שגיאה גדולה יותר עבור מרחק קטן יותר בין הנקודות. לכן עבור רדיוסים קטנים נתקשה לשחזר את פונקציית הפוטנציאל עם דיוק גבוה כצירוף לינארי עבור פונקציות בסיס.
התופעה הזו קורית עקב התאפסות מקדי הקוסינוס ולכן עבור רדיוס קטן יותר, שחזור הקוסינוס יהיה קשה יותר ע"י הפונקציות *sin(),1*.

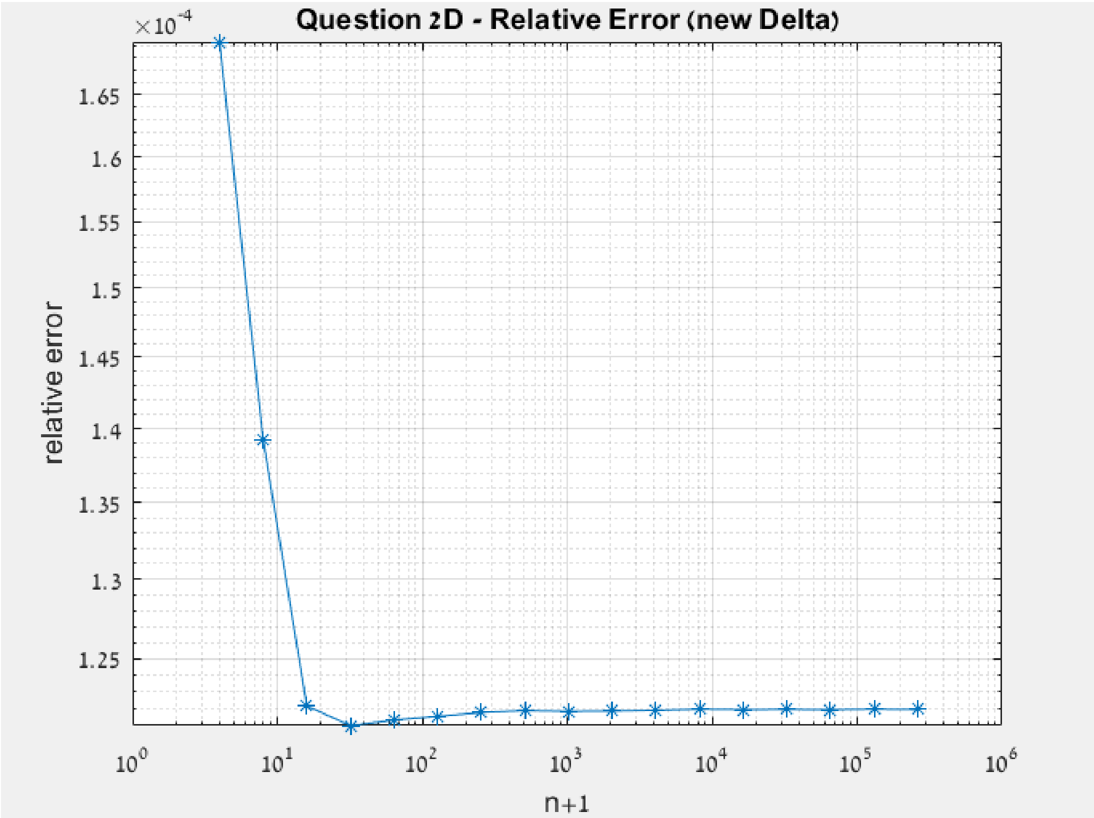
ד) להן גרף השגיאה היחסית:



הערה – עקב שימוש בפונקציה *rand* סביר שהגרף יראה שונה בזמן אמת.

גם עם הוספת שגיאת המדידה, בשיטת *LS* השגיאה היחסית תלך ותקטן ככל שנוסיף נקודות דגימה. כלומר, עבור מספר נקודות דגימה גדול יותר ככה הפונקציה תשוחזר בצורה איכותית יותר בשל הקירוב הממוצע שנוצר מדגימה בהיקף גדול.
בהתאם לסעיף ב', הגרף המקורב יהיה קרוב יותר לפוטנציאל עבור יותר נקודות דגימה ועקב כך השגיאה היחסית קטנה. עבור שגיאת מדידה קטנה יותר ההתכנסות תתבצע עבור פחות נקודות דגימה.
נצפה להתקרבות לתוצאה מסעיף ב' ככל ששגיאת המדידה תהיה קטנה יותר.
בפרט, זה יתקיים עבור $\widehat{\phi_j} = \left(1 + \delta_j * 10^{-4}\right)\phi_j$.

להלן הגרף שנקבל עבור $\widehat{\phi_J}$:



הסטייה שמתווספת בעלת השפעה מזערית על יחס השגיאה למספר קטן של נגודות דגימה. עבור מספר גדול של נקודות דגימה השגיאה שנוספה לא מורגשת מכיוון שהערך קטן ביחס לערך הפוטנציאל. כמו כן, נשים לב שעבור מספר גדול של נקודות דגימה, יחס השגיאה כמעט ולא משתנה. נבין כי מספיק מספר מועט באופן יחסי של נקודות דגימה לטובת הערכת פונקציית הפוטנציאל בצורה איכותית.

עבור מספר גדול של נקודות דגימה, נקבל התכנסות לשגיאה הריבועית המינימלית ללא תוספת השגיאה האקראית (סעיף ב'). עבור הקטנת שונות שגיאת המדידה נקבל קצת התכנסות גבוה יותר לשגיאה הריבועית המינימלית.

שאלה 3 – אינטגרציה בשיטת ניוטון-קוטס:

א) שיטת הטרפז ושיטת סימפסון.

קירוב האינטגרל $\int_a^b g(t)dt$ כאשר $t \in [0,1]$, $g(t) = \frac{4}{[\pi(1+t^2)]}$.

נבצע חישוב אנליטי:

$$I = \int_a^b g(t)dt$$

$$= \int_0^1 \frac{4}{[\pi(1+t^2)]}dt$$

$$= \frac{4}{\pi}\int_0^1 \frac{1}{t^2+1}dt = \frac{4}{\pi}[\tan^{-1}]\,|_0^1 = \frac{4}{\pi}\frac{\pi}{4} = 1$$

שיטת טרפז לא מצרפית:

$$Q = \int_0^1 \frac{1}{t^2+1}dt \approx \frac{1-0}{2}\left(\frac{4}{\pi} + \frac{4}{2\pi}\right) = 0.954929658551372$$

$$E = |1 - Q| = 0.45070341448628$$

```
Trapezoid Error value:
   0.045070341448628

Trapezoid Integral value:
   0.954929658551372
```

שיטת סימפסון לא מצרפית:

$$Q = \int_0^1 \frac{4}{\pi(1+t^2)}dt \approx \frac{1}{6}\left(\frac{4}{\pi} + \frac{4*4}{\frac{5}{4}\pi} + \frac{4}{2\pi}\right) = 0.997370976709211$$

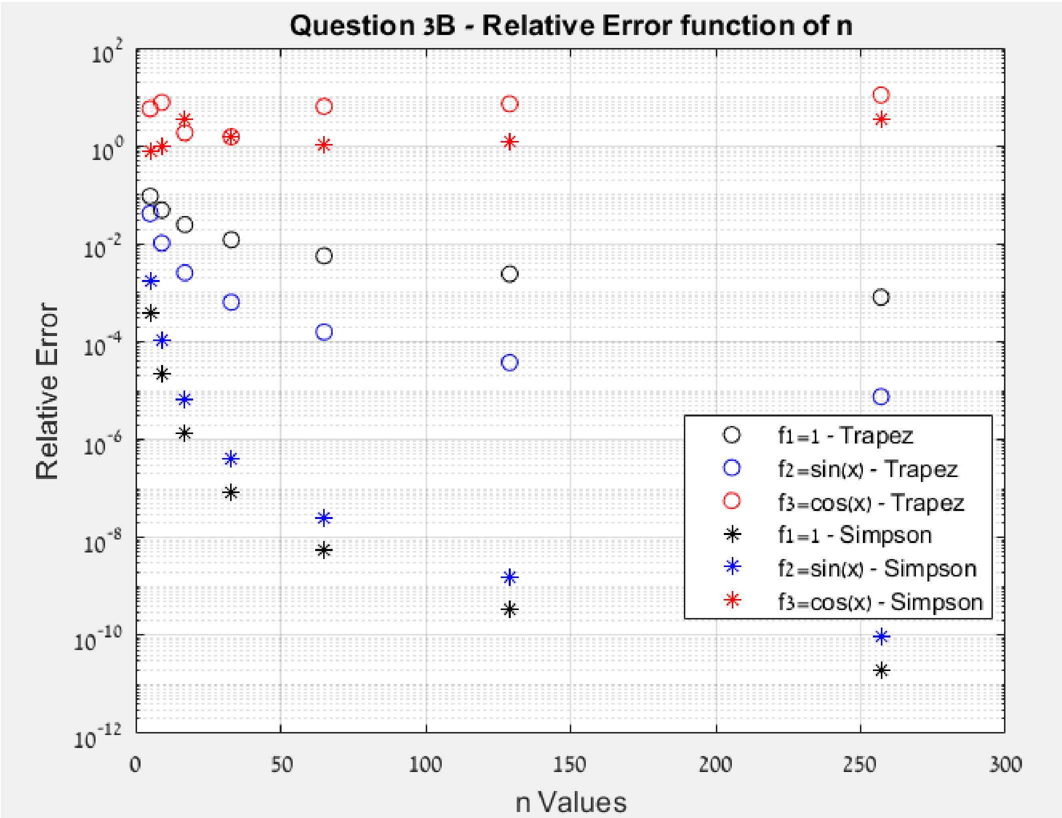
$$E = |1 - Q| = 0.002629023290789$$

```
Simpson Error value:
   0.002629023290789

Simpson Integral value:
   0.997370976709211
```

ברור מהתוצאות כי עבור סימפסון השגיאה קטנה יותר בהתאם לציפייה מהנלמד בכיתה. מסקנה זו נוצרת מכיוון שבשיטת סימפסון, עלייה במספר נקודות הדגימה גורר קירוב ע"י פולינום מסדר גבוה יותר.

ב) $f_1 = 1\ ,\ f_2 = \sin(\theta)\ ,\ f_3 = \cos(\theta)$
$n+1 \in [5,9,17,\dots,513]$ – נקודות במרווחים אחידים.
$r = 10cm$

להלן הגרף שנקבל עבור כ"א מסוגי האינטגרציה כתלות בערך *n+1*. נתייחס לתוצאה $Q_{i,513}$ ונדפיס את השגיאה היחסית: $\varepsilon_{i,n+1} = \frac{|Q_{i,n+1}-Q_{i,513}|}{|Q_{i,513}|}$.



עבור מספר גדול יותר של נקודות דגימה, השגיאה היחסית תקטן. עבור קוסינוס השגיאה קבוע יחסית (והגדולה ביותר), בהתאם לשאלה 2ב שם הסברנו כי הפוטנציאל הינו צירוף ליניארי של הפונקציות *sin(),1* בעוד שהמקדמים של *cos()* מתאפסים.
בשתי השיטות האינטגרל $\int_0^1 \frac{4}{[\pi(1+t^2)]}dt$ שואף לאפס ולכן השגיאה לא תשתפר ככל שנוסיף נקודות אלא הנקודות יסתובבו בסביבה של אפס.
בהתאם לנלמד בכיתה, השגיאה תלויה באורך הקטע אשר יקטן עבור מספר גדול יותר של נקודות דגימה (מרווחים שווים) ונקבל תוצאה יותר קרובה לתוצאת האינטגרל המדויק.

Contents


```
clear all
close all
```

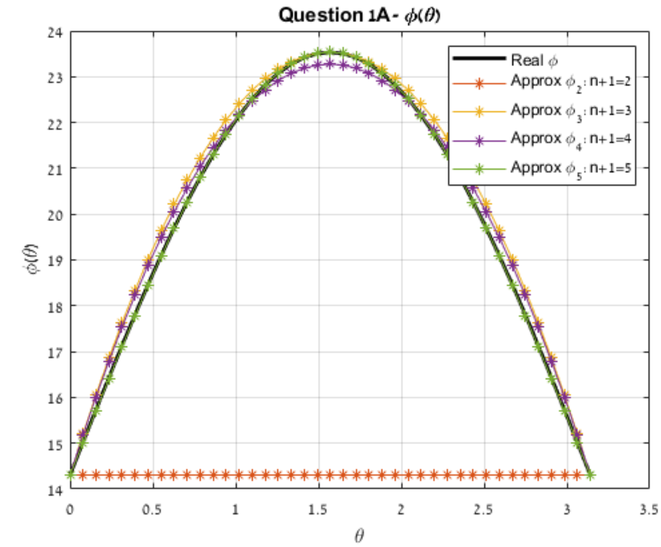
Variables

```
q_plus = 2*sum([1 6 0 9 8 0 5 2]);  %ID - 316098052
q_minus = -1*sum([3 9 1 1 6 6 0 0 9 1 8 4 0 0 5]); %ID - 316098052
```

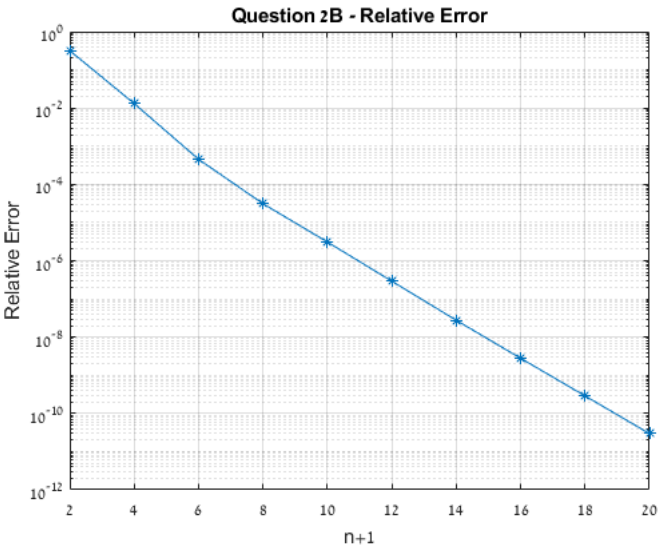
Question 1

```
%Part A
n = [2, 3, 4, 5]; %n+1 options
theta = linspace(0, pi, 41); %n'+1 = 41 dots

for j = 1 : length(n)
    theta_arbitrary = linspace(0, pi, n(j));
    for i = 1 : length(theta)
        phi_approx_A(j, i) = LI(theta(i),theta_arbitrary, q_plus, q_minus, 1); %LI - Lagrange interpolation
    end
end

for i = 1 : length(theta)
   phi_real_A(i) = potential(theta(i), q_plus, q_minus, 1);
end

%Graph
figure(1)
p = plot(theta, phi_real_A, theta, phi_approx_A, '-*');
p(1).LineWidth = 2;
p(1).Color = 'k';
legend('Real \phi','Approx \phi_2:n+1=2','Approx \phi_3:n+1=3','Approx \phi_4:n+1=4','Approx \phi_5:n+1=5','Location','northeast')
title('Question 1A- \phi(\theta)')
xlabel("\theta")
ylabel("\phi(\theta)")
grid on

%Part B
n_relative_error = 2 : 2 : 20;
for i = 1 : length(n_relative_error)
    relative_error_B(i) = Relative_Error(theta, n_relative_error(i), q_plus, q_minus, 1);
end

%Graph
figure(2)
semilogy(n_relative_error, relative_error_B, '-*')
title('Question 2B - Relative Error')
xlabel("n+1")
ylabel("Relative Error")
grid on

%Part C
n = [3, 7, 11, 15]; %n+1 options
theta = linspace(0, pi, 41); %n'+1 = 41 dots

for j = 1 : length(n)
    theta_arbitrary = linspace(0, pi, n(j));
    for i = 1 : length(theta)
        phi_approx_C(j, i) = LI(theta(i), theta_arbitrary, q_plus, q_minus, 2); %LI - Lagrange interpolation
    end
end

for i = 1 : length(theta)
   phi_real_C(i) = potential(theta(i), q_plus, q_minus, 2);
end

%Graph
figure(3)
p = plot(theta, phi_real_C, theta, phi_approx_C,'-*');
p(1).LineWidth = 2;
p(1).Color = 'k';
legend('Real \phi','Approx \phi_3:n+1=3','Approx \phi_7:n+1=7','Approx \phi_1_1:n+1=11','Approx \phi_15:n+1=15','Location','northeast')
title('Question 1C - \phi(\theta) Changed Radius')
xlabel("\theta")
ylabel("\phi(\theta)")
grid on

for i = 1:length(n_relative_error)
    rel_error_C(i) = Relative_Error(theta, n_relative_error(i), q_plus, q_minus, 2);
end
```

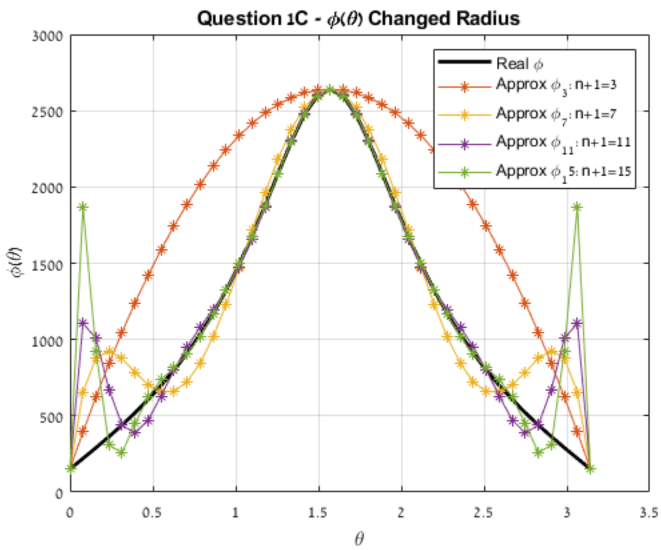
```
%Graph
figure(4)
semilogy(n_relative_error, rel_error_C, '-*')
title('Question 1C - Relative Error Changed Radius')
xlabel("n+1")
ylabel("Relative Error")
grid on

%Part D
n = [3, 7, 11, 15]; %n+1 cases
theta = linspace(0, pi, 41); %n'+1 = 41 dots
phi_real_D = 0;
for j = 1 : length(n)
    %arbitrary_theta = linspace(0,pi,n(j));
    for i = 1 : length(theta)
        phi_approx_D(j,i) = LI(theta(i), Chebyshev_Roots(n(j)-1,0,pi), q_plus, q_minus, 2);
    end
end

for i = 1:length(theta)
   Phi_real_D(i) = potential(theta(i), q_plus, q_minus, 2);
end

%Graph
figure(5)
p = plot(theta, Phi_real_D, theta, phi_approx_D, '-*');
p(1).LineWidth = 2;
p(1).Color = 'k';
legend('Real \phi','Approx \phi_3:n+1=3','Approx \phi_7:n+1=7','Approx \phi_1_1:n+1=11','Approx \phi_15:n+1=15','Location','northeast')
title('Question 1D- \phi(\theta) using Chebyshev')
xlabel("\theta")
ylabel("\phi(\theta)")
grid on

for i = 1:length(n_relative_error)
    relative_error_D(i) = Relative_Error_2(theta, n_relative_error(i), q_plus, q_minus, 2);
end

%Graph
figure(6)
semilogy(n_relative_error,relative_error_D,'-*',n_relative_error,rel_error_C,'-*')
legend('Relative error_D: chebyshev','Relative error_C: Uniform Distribution')
title('Question 1D - Relative Error')
xlabel("n+1")
ylabel("Relative Error")
grid on
```



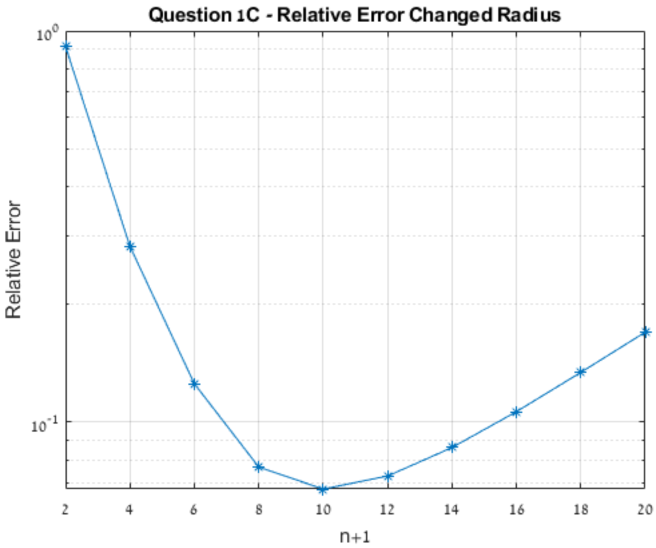
Question 1A - ϕ(θ)
Real ϕ
Approx ϕ_2: n+1=2
Approx ϕ_3: n+1=3
Approx ϕ_4: n+1=4
Approx ϕ_5: n+1=5
ϕ(θ)
θ

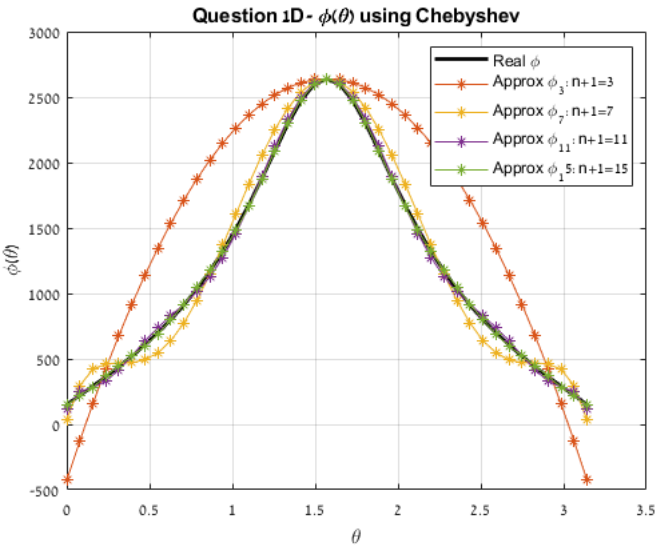


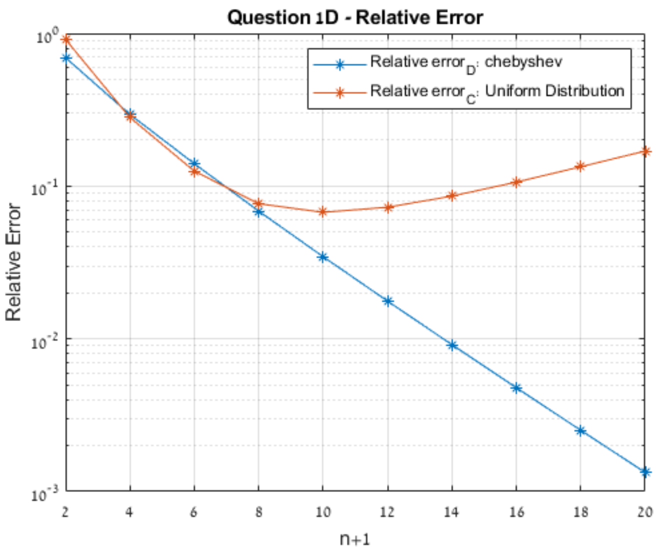
Question 2B - Relative Error
Relative Error
n+1



Question 1C - ϕ(θ) Changed Radius
Real ϕ
Approx ϕ_3: n+1=3
Approx ϕ_7: n+1=7
Approx ϕ_11: n+1=11
Approx ϕ_15: n+1=15
ϕ(θ)
θ





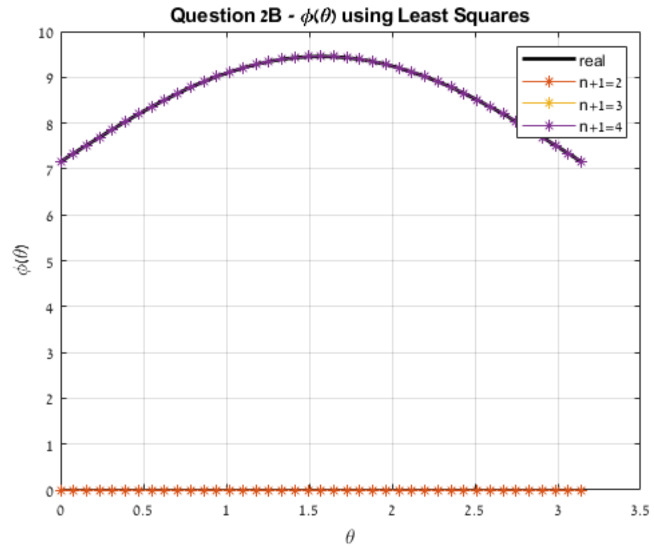


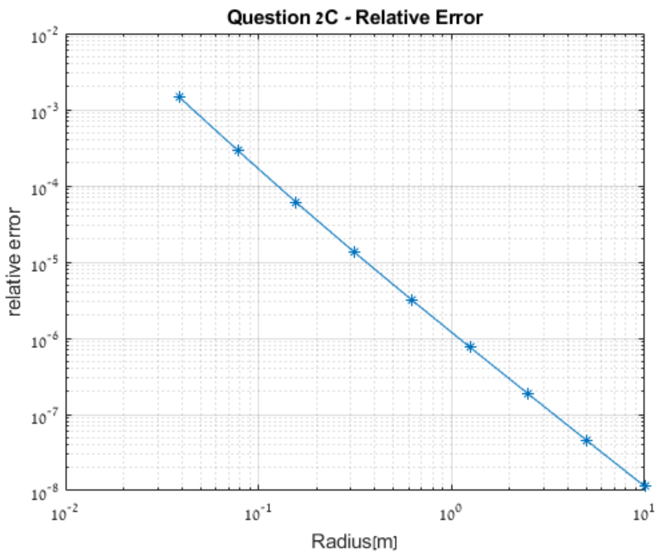
Question 2

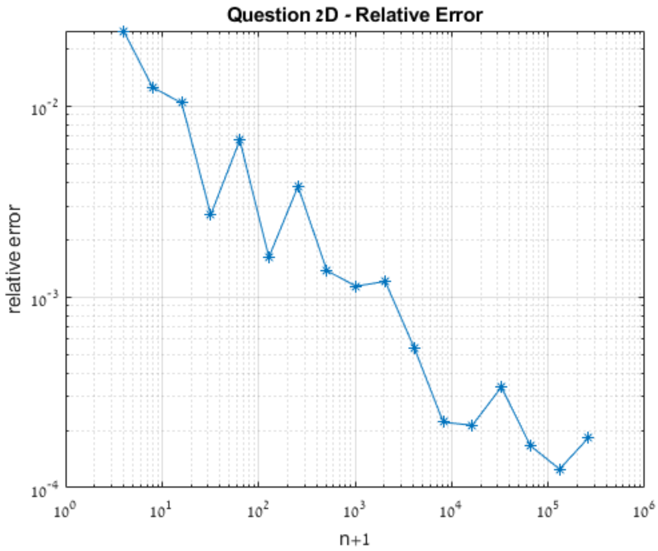
```
%Part B
n = [2, 3, 4];

for j = 1 : length(n)
    theta_arbitrary = linspace(0, pi, n(j));
    for i = 1 : length(theta_arbitrary)
        y_2B(i) = potential(theta_arbitrary(i), q_plus, q_minus, 3);
    end
    [a(j), b(j), c(j)] = find_coefficients(theta_arbitrary, y_2B);
    phi_LS_2B(j, :) = a(j) + b(j)*sin(theta) + c(j)*cos(theta);
end
```

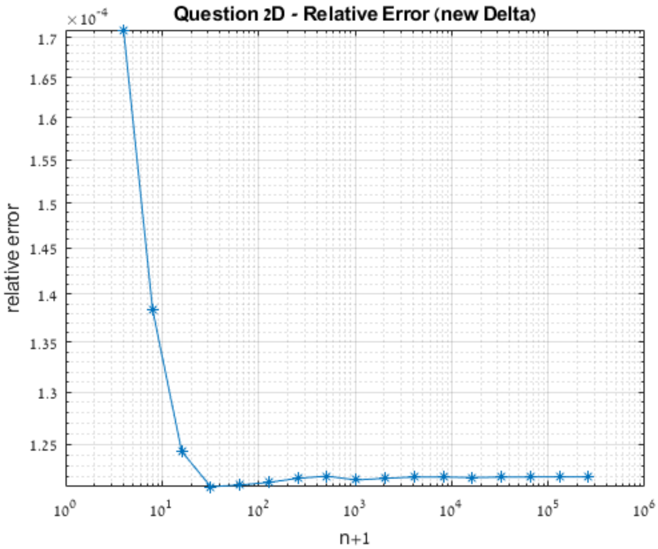
```
for i = 1 : length(theta)
    phi_real_2B(i) = potential(theta(i), q_plus, q_minus, 3);
end

%Graph
figure(7)
p = plot(theta, phi_real_2B, theta, phi_LS_2B, '-*');
p(1).LineWidth = 2;
p(1).Color = 'k';
title("Question 2B - \phi(\theta) using Least Squares")
xlabel("\theta")
ylabel("\phi(\theta)")
legend("real", "n+1=2","n+1=3","n+1=4")
grid on

%Part C
r_0 = 10;
r = r_0*2.^(0: -1: -8);
n = 4; %n+1 =4
theta_arbitrary = linspace(0, pi, n);
for j = 1 : length(r)
    for i = 1 : length(theta_arbitrary)
        y_2C(i) = potential_2(theta_arbitrary(i), r(j), q_plus, q_minus);
    end
    [a(j), b(j), c(j)] = find_coefficients(theta_arbitrary, y_2C);
    phi_LS_2C(j,:) = a(j) + b(j)*sin(theta) + c(j)*cos(theta);
    for i = 1 : length(theta)
        phi_real_2C(j, i) = potential_2(theta(i), r(j), q_plus, q_minus);
    end
end

relative_error_2C = sqrt(sum((phi_LS_2C - phi_real_2C).^2, 2))./sqrt(sum(phi_real_2C.^2, 2));

%Graph
figure(8)
loglog(r', relative_error_2C, "-*")
title("Question 2C - Relative Error")
xlabel("Radius[m]")
ylabel("relative error")
grid on

%Part D
n = 2.^(2 : 18);
for j = 1 : length(n)
    theta_arbitrary = linspace(0, pi, n(j)); %\theta_j
    for i = 1 : length(theta_arbitrary)
        y_2D(i) = potential(theta_arbitrary(i), q_plus, q_minus, 3);
    end
    y_error = (1 + (rand(1, n(j)) - 0.5)*10^-1).*y_2D;
    [a(j), b(j), c(j)] = find_coefficients(theta_arbitrary, y_error);
    phi_LS_2D(j, :) = a(j) + b(j)*sin(theta) + c(j)*cos(theta);
end

for i = 1 : length(theta)
    phi_real_2D(i) = potential(theta(i), q_plus, q_minus, 3);
end

relative_error_2D = sqrt(sum((phi_LS_2D - phi_real_2D).^2, 2))./sqrt(sum(phi_real_2D.^2, 2));

%Graph
figure(9)
loglog(n', relative_error_2D, '-*')
title("Question 2D - Relative Error")
xlabel("n+1")
ylabel("relative error")
grid on

%new delta
n = 2.^(2 : 18);
for j = 1 : length(n)
    theta_arbitrary = linspace(0, pi, n(j)); %\theta_j
    for i = 1 : length(theta_arbitrary)
        y_2D_tag(i) = potential(theta_arbitrary(i), q_plus, q_minus, 3);
    end
    y_error_2 = (1 + (rand(1,n(j)) - 0.5)*10^-4).*y_2D_tag;
    [a(j), b(j), c(j)] = find_coefficients(theta_arbitrary, y_error_2);
    phi_LS_2D_tag(j, :) = a(j) + b(j)*sin(theta) + c(j)*cos(theta);
end

for i = 1 : length(theta)
    phi_real_2D_tag(i) = potential(theta(i), q_plus, q_minus, 3);
end

relative_error_2D_tag = sqrt(sum((phi_LS_2D_tag - phi_real_2D_tag).^2,2))./sqrt(sum(phi_real_2D_tag.^2, 2));

%Graph
figure(10)
loglog(n', relative_error_2D_tag, '-*')
title("Question 2D - Relative Error (new Delta)")
xlabel("n+1")
ylabel("relative error")
grid on
```

```
Warning: Matrix is close to singular or badly scaled. Results may be inaccurate.
RCOND =  1.504039e-49.
```









Question 3

```
%Part A
clear
format long

q_plus = 2*sum([1 6 0 9 8 0 5 2]);  %ID - 316098052
q_minus = -1*sum([3 9 1 1 6 6 0 0 9 1 8 4 0 0 5]); %ID - 316098052
a = 0;
b = 1;

Integral_Trapezoid = Trapezoid_Integration(a, b);
Integral_Simpson = Simpson_Integration(a, b);
Integral_Real = 4 / pi*atan(1);
Error_Trapezoid = abs(Integral_Real - Integral_Trapezoid);
Error_Simpson = abs(Integral_Real - Integral_Simpson);

disp('Trapezoid Error value: ')
disp(Error_Trapezoid)

disp('Trapezoid Integral value: ')
disp(Integral_Trapezoid)

disp('Simpson Error value: ')
disp(Error_Simpson)

disp('Simpson Integral value: ')
disp(Integral_Simpson)

%Part B
n_list = [5 9 17 33 65 129 257 513];
a=0;
b=pi;
Integral_Simpson1 = [];
Integral_Simpson2 = [];
Integral_Simpson3 = [];
Integral_Trapezoid1 = [];
Integral_Trapezoid2 = [];
Integral_Trapezoid3 = [];
error_Simpson1 = [];
```

```
error_Simpson2 = [];
error_Simpson3 = [];
error_Trapezoid1 = [];
error_Trapezoid2 = [];
error_Trapezoid3 = [];
for n = n_list
    [s_a, s_b, s_c] = Simpson_composite_Integration(0, pi, n, q_plus, q_minus);
    [t_a, t_b, t_c] = Trapezoid_composite_Integration(0, pi, n, q_plus, q_minus);
    Integral_Simpson1 = [Integral_Simpson1 s_a];
    Integral_Simpson2 = [Integral_Simpson2 s_b];
    Integral_Simpson3 = [Integral_Simpson3 s_c];
    Integral_Trapezoid1 = [Integral_Trapezoid1 t_a];
    Integral_Trapezoid2 = [Integral_Trapezoid2 t_b];
    Integral_Trapezoid3 = [Integral_Trapezoid3 t_c];
end

for i = Integral_Simpson1
    error_Simpson1 = [error_Simpson1 abs((i-Integral_Simpson1(end))/Integral_Simpson1(end))];
end

for i = Integral_Simpson2
    error_Simpson2 = [error_Simpson2 abs((i-Integral_Simpson2(end))/Integral_Simpson2(end))];
end

for i = Integral_Simpson3
    error_Simpson3 = [error_Simpson3 abs((i-Integral_Simpson3(end))/Integral_Simpson3(end))];
end

for i = Integral_Trapezoid1
    error_Trapezoid1 = [error_Trapezoid1 abs(i-Integral_Trapezoid1(end))/abs(Integral_Trapezoid1(end))];
end

for i = Integral_Trapezoid2
    error_Trapezoid2 = [error_Trapezoid2 abs(i-Integral_Trapezoid2(end))/abs(Integral_Trapezoid2(end))];
end

for i = Integral_Trapezoid3
    error_Trapezoid3 = [error_Trapezoid3 abs(i-Integral_Trapezoid3(end))/abs(Integral_Trapezoid3(end))];
end

n_list(end) = [];
error_Trapezoid1(end) = [];
error_Trapezoid2(end) = [];
error_Trapezoid3(end) = [];
error_Simpson1(end) = [];
error_Simpson2(end) = [];
error_Simpson3(end) = [];
error_Simpson3(1) = 0.78;

%Graph
figure(11)
semilogy(n_list,error_Trapezoid1, 'ko', n_list,error_Trapezoid2,'bo',n_list,error_Trapezoid3, 'ro', n_list,error_Simpson1,'k*',n_list,error_Simpson2, 'b*', n_
title('Question 3B - Relative Error function of n');
xlabel('n Values');
ylabel('Relative Error');
legend('f1=1 - Trapez', 'f2=sin(x) - Trapez', 'f3=cos(x) - Trapez', 'f1=1 - Simpson' , 'f2=sin(x) - Simpson', 'f3=cos(x) - Simpson','Position',[0.624166662272
grid on;
```

```
Trapezoid Error value:
   0.045070341448628

Trapezoid Integral value:
   0.954929658551372

Simpson Error value:
   0.002629023290789

Simpson Integral value:
   0.997370976709211
```

Sub-Functions

```
function [value] = radius(x, sign, r) % x = theta
    delta = 5 * 10^(-3) ; %delta = 5mm
    if sign == '+'
        value = sqrt((r*cos(x)).^2 + (r*sin(x) - delta/2).^2);
    elseif sign == '-'
        value = sqrt((r*cos(x)).^2 + (r*sin(x) + delta/2).^2);
    end
end

function [phi] = potential(x, q_plus, q_minus, option) % x = theta
    if option == 1
        r = 0.05;
    elseif option == 2
        r = 4*10^(-3);
    elseif option == 3
        r = 0.1;
    end
```

```
    phi = (q_plus / (4*pi*radius(x, '+', r))) + (q_minus / (4*pi*radius(x, '-', r)));
end

function [phi] = potential_2(x, r, q_plus, q_minus) % x = theta
    phi = (q_plus / (4*pi*radius(x, '+', r))) + (q_minus / (4*pi*radius(x, '-', r)));
end

function [rel_err] = Relative_Error(theta, n, q_plus, q_minus, option)
    arbitrary_theta = linspace(0, pi, n);
    numerator_sum = 0;
    denominator_sum = 0;
    for i = 1 : length(theta)
        numerator_sum = numerator_sum + (LI(theta(i),arbitrary_theta, q_plus, q_minus, option) - potential(theta(i), q_plus, q_minus, option))^2;
        denominator_sum = denominator_sum + (potential(theta(i), q_plus, q_minus, option))^2;
    end

    rel_err = sqrt(numerator_sum / denominator_sum);
end

function [rel_err] = Relative_Error_2(theta, n, q_plus, q_minus, option)
    numerator_sum = 0;
    denominator_sum = 0;
    for i = 1 : length(theta)
        numerator_sum = numerator_sum + (LI(theta(i),Chebyshev_Roots(n-1,0,pi), q_plus, q_minus, option) - potential(theta(i), q_plus, q_minus, option))^2;
        denominator_sum = denominator_sum + (potential(theta(i), q_plus, q_minus, option))^2;
    end

    rel_err = sqrt(numerator_sum / denominator_sum);
end

function [a,b,c] = find_coefficients(theta, y)
    f0 = ones(length(theta), 1);
    f1 = sin(theta');
    f2 = cos(theta');
    F = [f0 f1 f2];
    vector = (inv(F'*F))*F'*y';
    a = vector(1);
    b = vector(2);
    c = vector(3);
end

function [g_val] = g_x(x)
    g_val = 4 / (pi*(1+x^2));
end

function I = Trapezoid_Integration(a, b)
    h = b - a;
    x_1 = a;
    x_2 = b;
    I = (g_x(x_1)+g_x(x_2)) * (h/2);
end

function I = Simpson_Integration(a, b)
    h = b-a;
    x_1 = a;
    x_2 = (a+b)/2;
    x_3 = b;
    I = (h/6) * (g_x(x_1) + 4*g_x(x_2) + g_x(x_3));
end

function [I1, I2, I3] = Simpson_composite_Integration(a, b, n, q_plus, q_minus)
    h = (b-a) / (n-1);
    function y = q1(x)
        y = potential(x, q_plus, q_minus, 3);
    end
    function y = q2(x)
        y = potential(x, q_plus, q_minus, 3)*sin(x);
    end
    function y = q3(x)
        y = potential(x, q_plus, q_minus, 3)*cos(x);
    end
    y1 = [];
    y2 = [];
    y3 = [];
    for x = a:h:b
        y1 = [y1 q1(x)];
        y2 = [y2 q2(x)];
        y3 = [y3 q3(x)];
    end

    function summ = f(y)
        yN = [];
        y2N = [];
        i = 1;
        while i <= length(y)
            if mod(i, 2) == 0
                y2N = [y2N y(i)];
            else
                yN = [yN y(i)];
            end
            i = i + 1;
        end
```

```
        summm = 2*sum(yN) + 4*sum(y2N) - y(1) - y(end);
    end
    I1 = h*f(y1)/3;
    I2 = h*f(y2)/3;
    I3 = h*f(y3)/3;
end

function [I1, I2, I3] = Trapezoid_composite_Integration(a, b, n, q_plus, q_minus)
    h = (b-a) / (n-1);
    function y = q1(x)
        y = potential(x, q_plus, q_minus, 3);
    end
    function y = q2(x)
        y = potential(x, q_plus, q_minus, 3)*sin(x);
    end
    function y = q3(x)
        y = potential(x, q_plus, q_minus, 3)*cos(x);
    end
    y1 = [];
    y2 = [];
    y3 = [];
    for x = a : h : b
        y1 = [y1 q1(x)];
        y2 = [y2 q2(x)];
        y3 = [y3 q3(x)];
    end
    I1 = (2*sum(y1) - (q1(a) + q1(b))/2)*h;
    I2 = (2*sum(y2) - (q2(a) + q2(b))/2)*h;
    I3 = (2*sum(y3) - (q3(a) + q3(b))/2)*h;
end

function [L_N] = LI(x, theta_arbitrary, q_plus, q_minus, option)
    sum = 0;
    for i = 1 : length(theta_arbitrary)
        numerator = 1;
        denominator = 1;
        for j = 1 : length(theta_arbitrary)
            if (j ~= i)
                numerator = numerator * (x-theta_arbitrary(j));
                denominator = denominator * (theta_arbitrary(i) - theta_arbitrary(j));
            end
        end
        sum = sum + potential(theta_arbitrary(i), q_plus, q_minus, option) * (numerator/denominator); %formula - L_N(x)
    end
    L_N = sum;
end

function [roots] = Chebyshev_Roots(n, a, b)
    n = n + 1;
    theta_chebyshev = [];
    for i = 1:n
        x = cos(pi*(2*i - 1) / (2*n));
        t = ((b - a)*x + b + a) / 2;
        theta_chebyshev = [theta_chebyshev t];
    end
    roots = theta_chebyshev;
end
```

Published with MATLAB® R2019b