**תוכן**

[דיון לגבי מקור האלגוריתם 2](#_heading=h.gjdgxs)

[דיון מתמטי מעמיק ברקע ובתאוריה, תוך הצגה מפורטת של כל הפרטים טכניים ביישום 2](#_heading=h.30j0zll)

[קוד מפורט 5](#_heading=h.1fob9te)

[מדריך למשתמש 10](#_heading=h.3znysh7)

[הצגה מפורטת 11](#_heading=h.2et92p0)

[הוכחה לנכונות התוכנה 12](#_heading=h.tyjcwt)

[קישור קוד ל GIT 12](#_heading=h.3dy6vkm)

## דיון לגבי מקור האלגוריתם

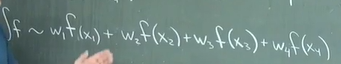
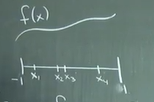
תרבוע גאוסיאני(gaussian quadrature) היא שיטת קירוב אנליטית לחישוב של האינטגרל המסוים של פונקציה. קארל פרדריך גאוס פרסם את השיטה במאמר בשנת 1814. גאוס זיהה את הפוטנציאל של חישוב האינטגרל המסוים ופיתח שיטה לחישוב האינטגרל באופן לינארי.

## דיון מתמטי מעמיק ברקע ובתאוריה, תוך הצגה מפורטת של כל הפרטים טכניים ביישום

הסבר:[Gaussian Quadrature 3: The Explanation of the Technique](https://www.youtube.com/watch?v=cKKrGr93f6c&t=141s&ab_channel=MathTheBeautiful)

שיטת תרבוע הגאוסיאנית מחשבת את האינטגרל על ידי פיתוח משוואה אלטרנטיבית לאינטגרל. שיטה זו מתבססת על שיטה שמחשבת את האינטגרל בתחום בין 1- ל1 ולכן תחילה צריך להמיר את תחום האינטגרל המסוים בתחום בין a ל b לתחום בין 1- ל1.

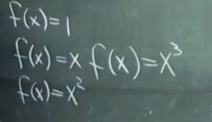
נשווה את האינטגרל בערכי הפונקציה בנקודות מסוימות ונכפול במשקלים כך שתתקבל התוצאה הנכונה:



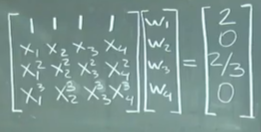
מספר הנקודות והמשקלים שנבחר יהיה שווה למעלה של הפונקציה שאנחנו מחפשים את האינטגרל שלה, ורמת הדיוק של השיטה תהיה תלויה ברמת הקירוב של הפונקציה ……

נגדים את השיטה על פונקציה ממעלה רביעית.

נזכור שהמטרה שלנו היא למצוא את ערכי המשקלים,ה wים כך שניישם את המשוואה על פונקציות מסוימות שידועה לנו תוצאת האינטגרל בתחום 1- ל1 ונקבל משוואות לינארית. ניקח לדוגמא את

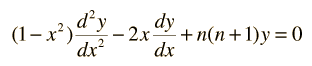


ידועות לנו תוצאות האינטגרל של הפונקציות האלו בתחום 1- ל1,נבנה משוואה לכל פונקציה ונקבל את מערכת המשואות הבאה:

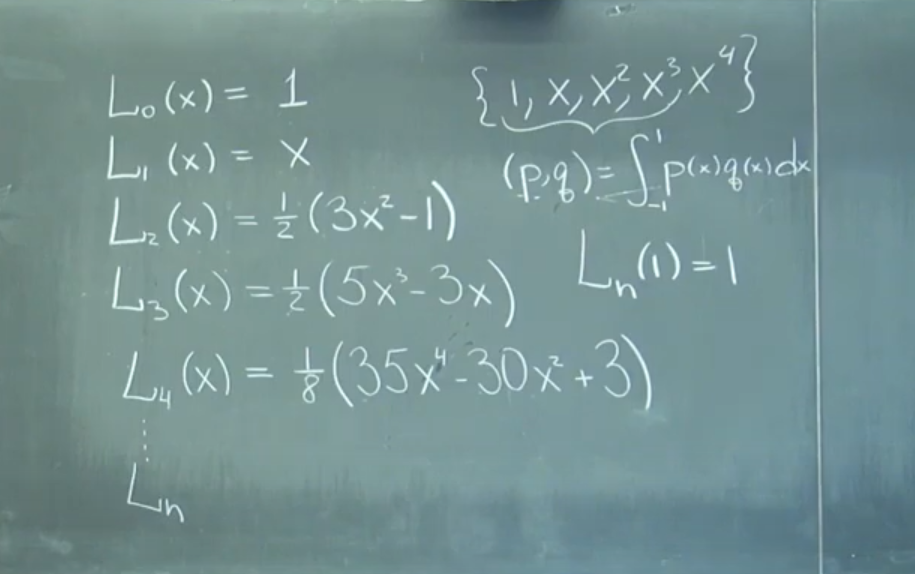


הבעיה שנוצרת לנו היא שערכי ה W לא יוצאים מדויקים ולכן המקדמים w מאוד גדולים (לדוגמא 100,-100,500) כאשר מכפילים שגיאה קטנה במספר גדול (המספרים גדלים ככל שיש יותר משתנים) השגיאה נהיית גדולה מאוד.

אז איך אפשר לפתור את הבעיה הזאת ? נעזר בשיטת לג'נדרה שרמת הדיוק בה היא יותר גבוהה.

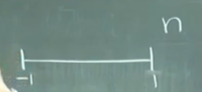
פולינום לג'נדרה הוא פולינום מהמבנה הבא:

נמצא את פולינומי לג'נדרה עבור {x^n, ... ,1} כאשר n הוא מספר הנקודות שבהן נשתמש כדי למצוא את האינטגרל:



נגדיר את הפולינום :

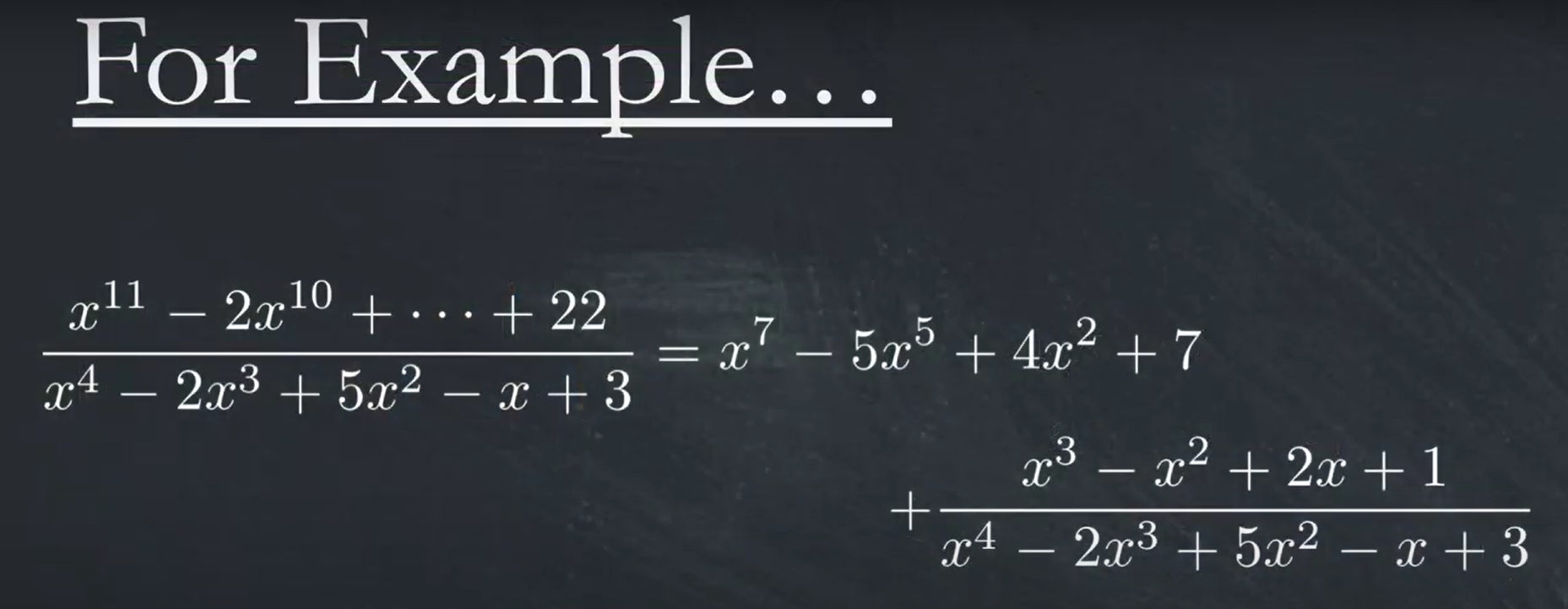


הטווח שלנו עדיין מ-1 ל +1 ואנחנו עדיין לא יודעים לבחור בצורה חכמה את הנקודות.

**איך נבחר בצורה הכי טובה את הנקודות?**

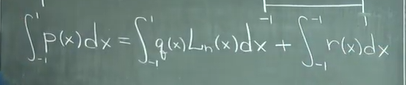
ראשית הפולינום P בחזקת 2n-1 ונחלק אותו ב Ln ונקבל פולינום ממעלה נמוכה ביותר + שארית

לדוגמא : חילוק פולינום ממעלה 11 בפולינום ממעלה 4 , נקבל פולינום ממעלה 7 +שארית

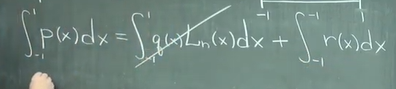


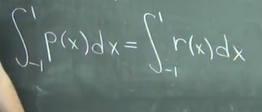
זאת החלוקה שנקבל 

ה r x זאת השארית שלנו מהחלוקה



אנו יודעים שהצבה של x ב q מתאפסת ולכן נשארת לנו רק השארית.





כדי שזה יעבוד אני אבחר את הנקודות n בהם ה Y מתאפס ואת ה w -ים אני אתאים בהתחשב בנקודות אלו שאפתור את מערכת המשוואות לינארית מטריקס וכך אתאים לכל x את ה w שלו.

בחירת הxים לפי פולינום לג'נדרה מגדילה את רמת הדיוק בשתי דרכים. הראשונה היא שערכי המשקלים המתקבלים יותר מכווצים וקרובים אחד לשני יחסית. שנית נוכל רמת דיוק גבוהה יותר עבור פולינום כלשהו. אם בשיטה הרגילה כדי קירוב בטווח אפסילון של כ1% בעזרת השיטה הזו הקירוב יהיה הרבה יותר מדויק עבור אותו מספר נקודות, כלומר משוואות.

## קוד מפורט

import numpy as np

import sympy as sp

from numpy import \*

from appJar import gui

import matplotlib.pyplot as plt

from matplotlib.patches import Polygon

*# Calc valor of equation send by user*

def funcao(eq, ti):

x = ti

return eval(eq)

*##*

def int\_symbolic(integral\_symbolic, a, b):

x = a

la = eval(integral\_symbolic)

x = b

lb = eval(integral\_symbolic)

return lb - la

*# GUI*

def press(click):

app.setSize(**"550x800"**)

eq = app.getEntry(**"Equation on python syintax"**)

a = float(app.getEntry(**"Lower limit"**))

b = float(app.getEntry(**"Upper limit"**))

points = int(app.getEntry(**'Enter the amount of points: '**))

*# Def points wi and ti*

witi = np.polynomial.legendre.leggauss(points)

pontoseq = []

*# Calc of integral*

integral\_numerical = 0

for i in range(points):

integral\_numerical += witi[1][i] \* funcao(eq, a + (b - a) / 2 \* (witi[0][i] + 1))

pontoseq.append(integral\_numerical)

integral\_symbolic = sp.integrate(eq)

longstring = str(**'Integral by symbolic resolution:**\n**'**

+ **'f(x) = '** + eq

+ **'**\n**Integral symbolic: '** + str(integral\_symbolic)

+ **'**\n**Numerical integral by gauss = '**

+ str(((b - a) / 2) \* integral\_numerical)

+ **'**\n**Exact integral = '**

+ str(int\_symbolic(str(integral\_symbolic), a, b))

+ **'**\n\n**Absolute error: '** +

str(abs(((b - a) / 2) \* integral\_numerical -

int\_symbolic(str(integral\_symbolic), a, b)))

+ **'**\n**Relative Error: '** + str(abs(

(((b - a) / 2) \* integral\_numerical - int\_symbolic(str(integral\_symbolic), a, b)) / int\_symbolic(

str(integral\_symbolic), a, b)) \* 100) +

**"%"**)

*#######plotting##########*

*# range*

def f(x):

return eval(str(eq))

x = np.linspace(a \* 0.8, b \* (1.2))

y = f(x)

fig, ax = plt.subplots()

ax.plot(x, y, **'b'**, linewidth=2)

ax.set\_ylim(bottom=0)

plt.title(**"Integral of f(x)"**)

ix = np.linspace(a, b)

iy = f(ix)

verts = [(a, 0), \*zip(ix, iy), (b, 0)]

poly = Polygon(verts, facecolor=**'0.9'**, edgecolor=**'0.5'**)

ax.add\_patch(poly)

ax.text(0.4 \* (a + b), f(b \* 1.1), **r"$\int\_"** + str(int(a)) + **"^"** + str(int(b)) + **"\hspace{1}"** + sp.latex(

integral\_symbolic) + **"\hspace{1}\mathrm{d}x$"**, horizontalalignment=**'center'**, fontsize=14)

fig.text(0.9, 0.05, **"$x$"**)

fig.text(0.1, 0.9, **'$y$'**)

ax.spines[**'right'**].set\_visible(False)

ax.spines[**'top'**].set\_visible(False)

ax.xaxis.set\_ticks\_position(**'bottom'**)

ax.set\_xticks((a, b))

ax.set\_xticklabels((**'$'** + str(int(a)) + **"$"**, **"$"** + str(int(b)) + **'$'**))

ax.set\_yticks([])

plt.savefig(**'figure.png'**)

app.addLabelEntry(longstring)

app.addImage(**"Grapich"**, **"figure.png"**)

*# End of def*

app = gui(**"GaussPy"**, **"550x200"**)

app.addLabel(

**"title"**, **"Calculator Integral by Gauss-Legendre Quadrature"**)

app.addLabelEntry(**"Equation on python syintax"**)

app.addLabelEntry(**"Upper limit"**)

app.addLabelEntry(**"Lower limit"**)

app.addLabelEntry(**'Enter the amount of points: '**)

app.addButtons([**"Run"**], press)

app.go()

## מדריך למשתמש

תמונה שמכילה טקסט

התיאור נוצר באופן אוטומטי



## הצגה מפורטת

הפונקציה : x^3+2x^2+4x+1

גבול עליון :1

גבול תחתון: -1

נקודות : 4

הנקודות והגובה שלהם:



חישוב אינטגרל אחרי שיש נקודות וגבהים :

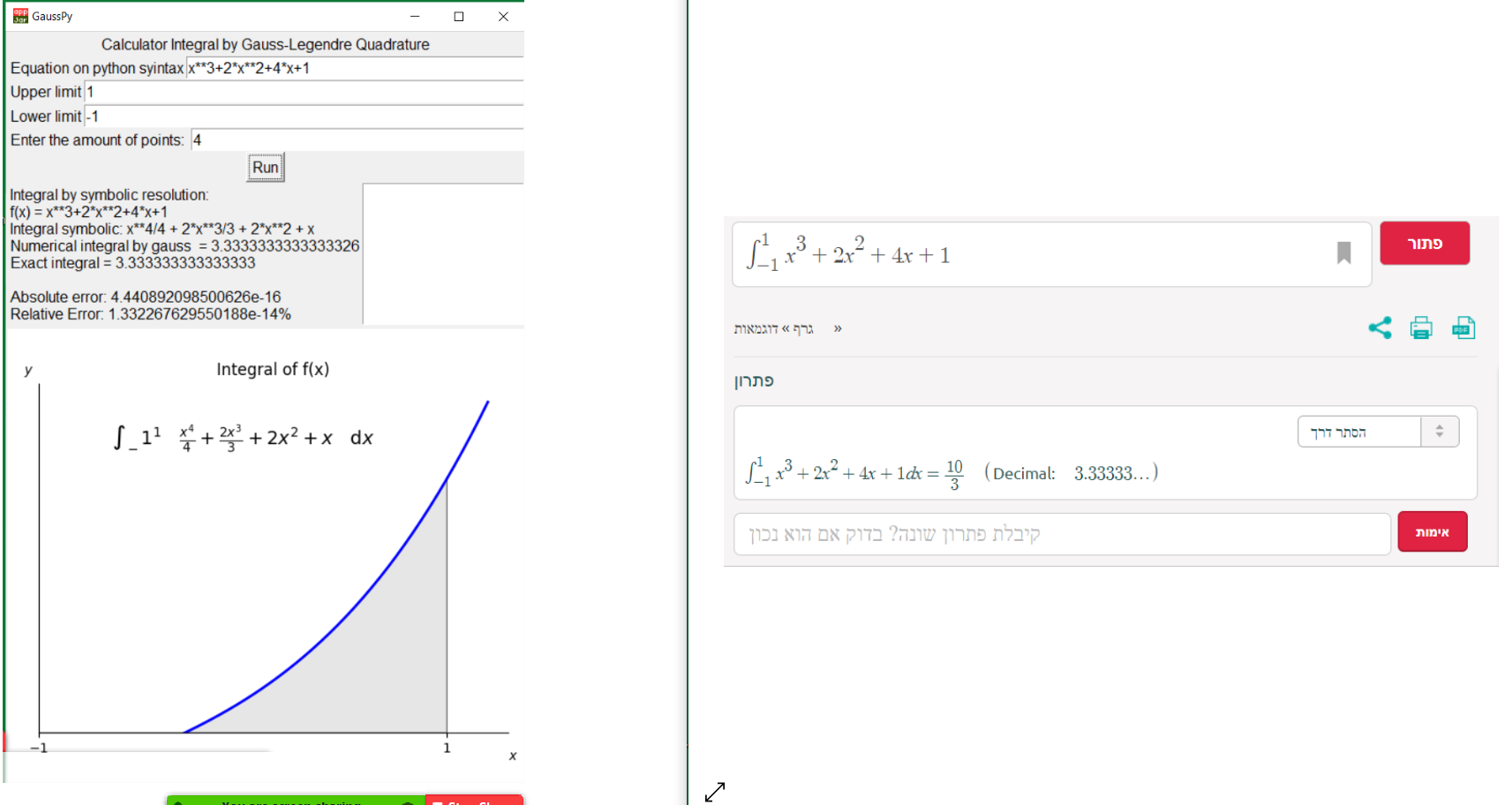
תמונה שמכילה טקסט

התיאור נוצר באופן אוטומטי

תמונה שמכילה טקסט

התיאור נוצר באופן אוטומטי

## הוכחה לנכונות התוכנה



## קישור קוד ל GIT

https://github.com/AlexSerdukov12/ProjectAnaliza