פרויקט סופי

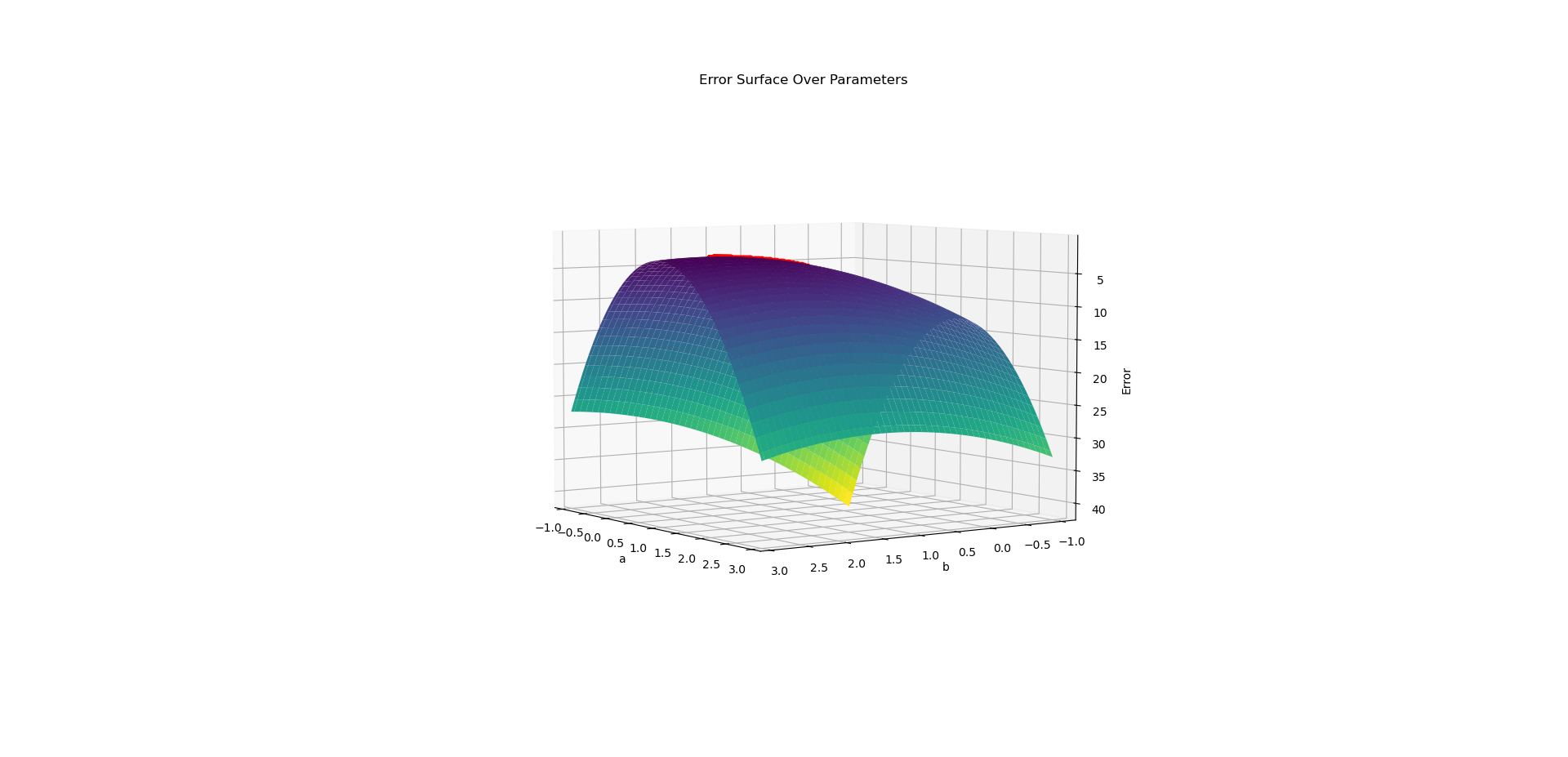
שאלה 1

1. Final parameters: a=0.997333333849981, b=2.367999994270291

2. ו- 3.

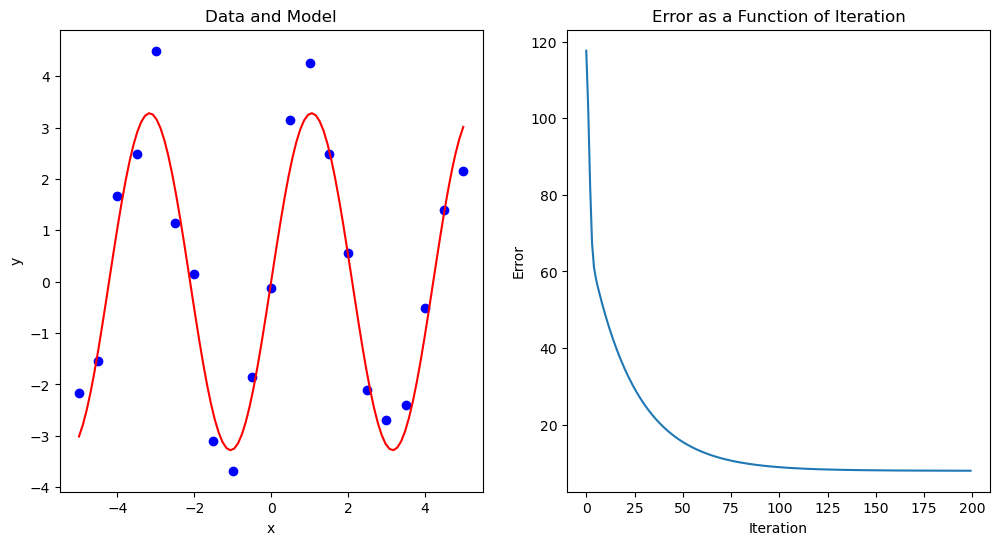
תמונה שמכילה טקסט, צילום מסך, תרשים, קו

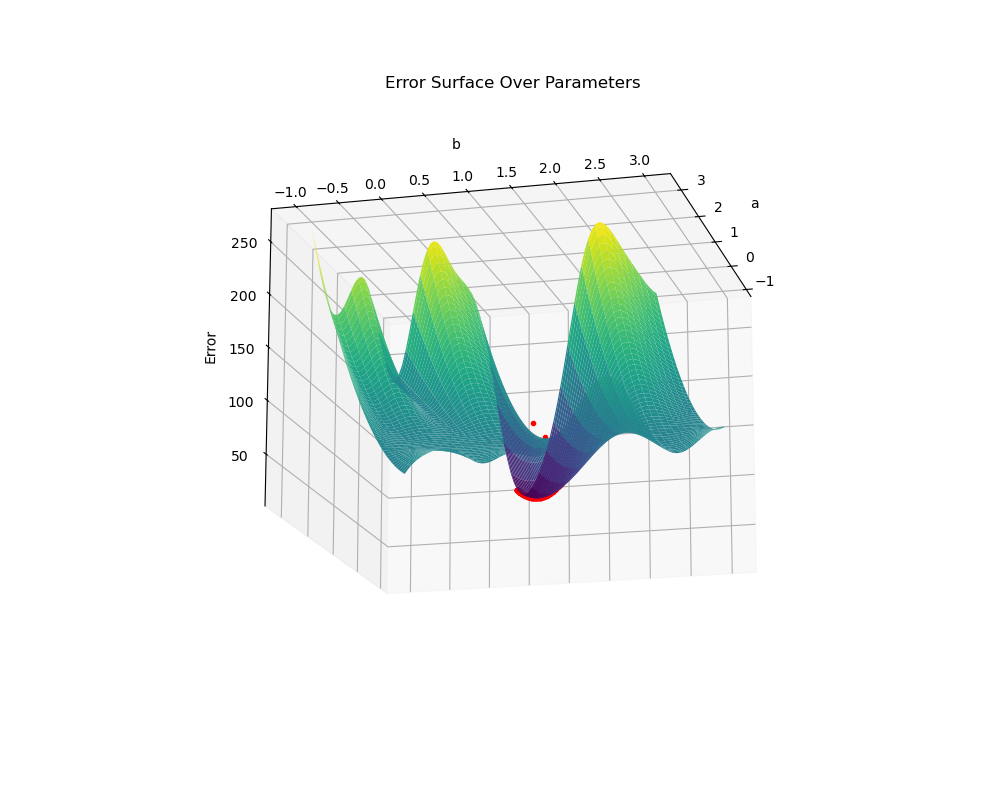
התיאור נוצר באופן אוטומטי



4.

\

1. 



קצב למידה: 0.001

# Learning rate

lr = 0.001

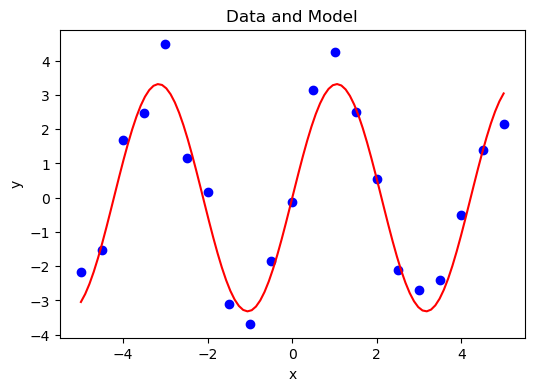
איטרציות: 200

# Number of iterations

iterations = 200

הפרמטרים שמצאנו הם:

Final parameters: a=3.280345856906753, b=1.4895161671221442

1.   
   פרמטרים סופיים:  
   Final parameters: a=3.318145436025057, b=1.488977981749276

נראה שהמודל מתאים באופן טוב מאוד לנתונים. הקו האדום (המודל) עובר דרך רוב הנקודות הכחולות (הנתונים). זה מראה שהפונקציה הסינוסואידלית מתאימה כמעט היטב למבנה של הנתונים.

הפרמטרים הסופיים שהפונקציה curve\_fit  מצאה הם:

* a = 3.318145436025057
* b = 1.488977981749276

הפרמטר a הוא הערך שבו הפונקציה הסינוסואידלית מתחילה (הגובה שלה), והפרמטר b הוא התדירות של הפונקציה. אם נשווה את הערכים האלה לערכים האמיתיים שהשתמשת בהם כדי ליצור את הנתונים, נראה שהפונקציה curve\_fitמצאה פרמטרים שדי קרובים לערכים האמיתיים. זה מראה שהפונקציה curve\_fit יכולה להיות כלי יעיל למציאת הפרמטרים של מודל שמתאים לנתונים.

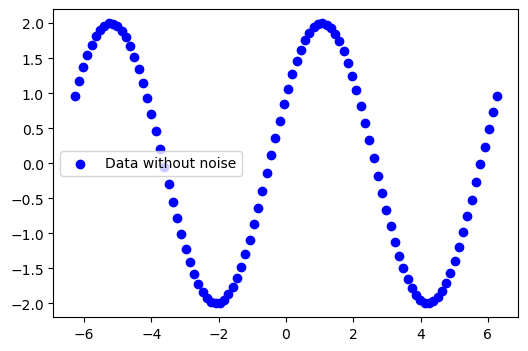
1. המודל שבחרו הוא: .

a, b ו- c הם הפרמטרים של המודל.

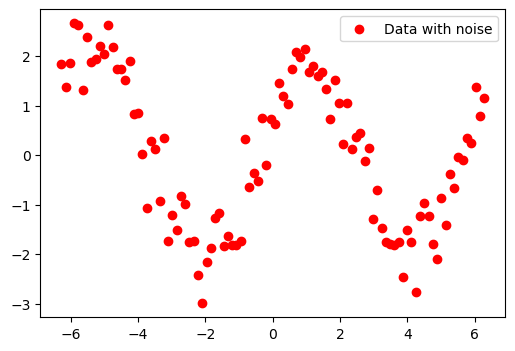
הפרמטרים ההתחלתיים שבחרנו הם: a\_val = 1, b\_val = 1, c\_val = 1

# Initial parameters

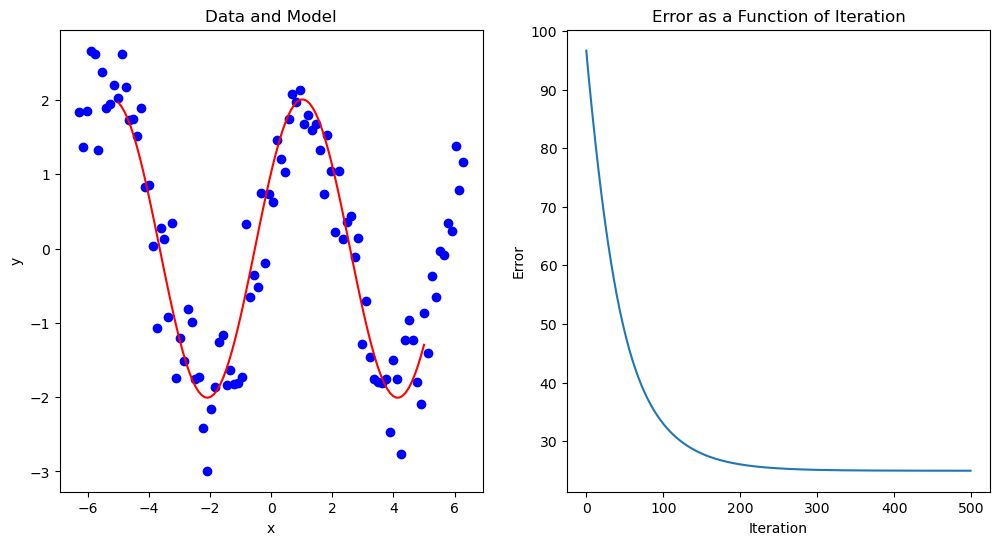
a\_val, b\_val, c\_val = 1, 1, 1

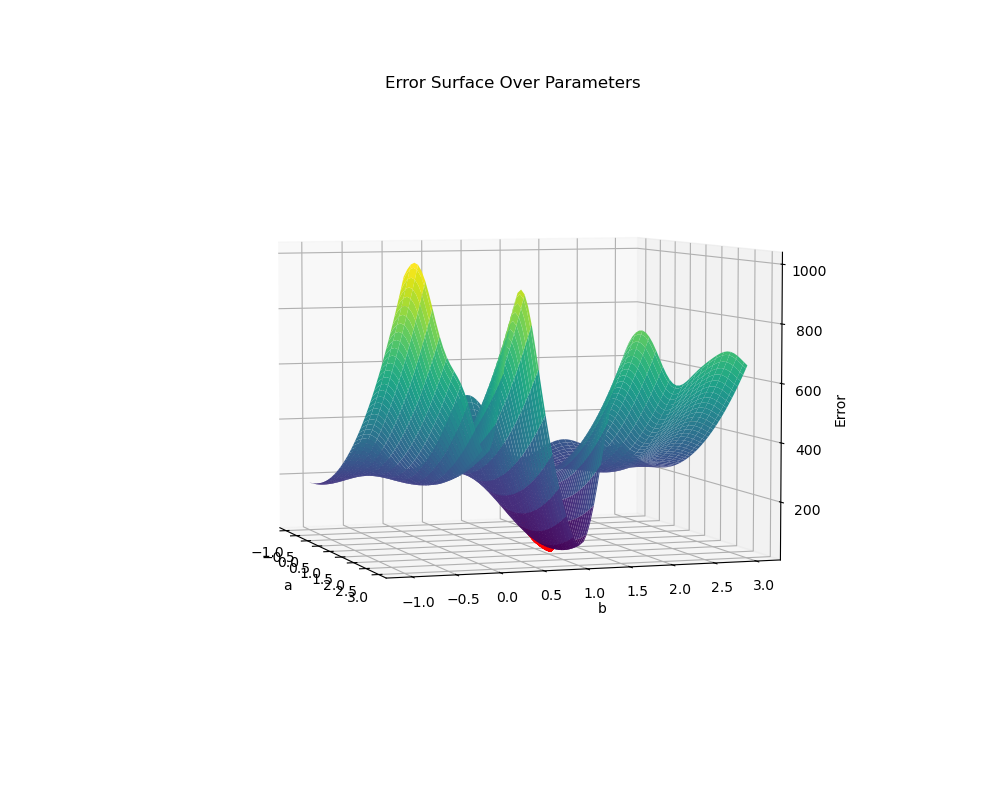
גרף לפני הוספת הרעש:  


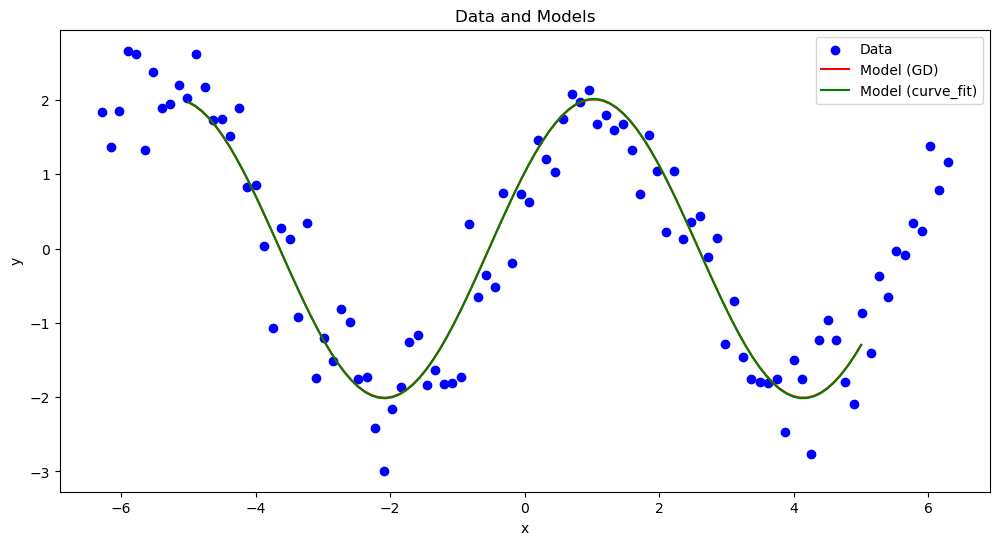
גרף אחרי הוספת הרעש:



גרפים בדומה לסעיף א:







פרמטרים שנמצאו על ידי Gradient descent:  
Final parameters: a=2.0081125397799204, b=1.0092632527092733, c=0.536024564951171

פרמטרים שנמצאו על ידי curve\_fit:  
Parameters found by curve\_fit: a=2.0156616351282866, b=1.0093224363174194, c=0.5359304117967073

הזמן שלוקח ל- Gradient descent:  
Time elapsed for gradient descent: 2.832636833190918 seconds

הזמן שלוקח ל- curve\_fit:  
Time elapsed for curve\_fit: 0.0 seconds

ההבדלים בין שתי השיטות, Gradient descent (GD) ו-curve\_fit :

* **זמן הריצה:**
  + Gradient descent: לקח כ-2.83 שניות למצוא את הפרמטרים האופטימליים.
  + :curve\_fit שיטה זו הייתה מהירה באופן משמעותי, ולקחה רק כ-0.0 שניות.

ההבדלים נובעים מהתכונות הייחודיות של כל שיטה:

* **:Gradient descent** Gradient Descent מעדכן באופן איטרטיבי את הפרמטרים. כל איטרציה כרוכה בחישוב הגרדיאנט ועדכון הפרמטרים בהתאם. תהליך זה נמשך עד שמגיעים להתכנסות.
* **:curve\_fit** fit\_curve משתמשת באלגוריתמי אופטימיזציה שתוכננו במיוחד כדי למצוא את הפרמטרים האופטימליים הממזערים את ההבדל בין הערכים הנצפים לערכים החזויים. אלגוריתמים אלו לרוב מתכנסים לפתרון בפחות איטרציות בהשוואה ל- Gradient Descent.

לסיכום, ההבדל בזמן הריצה נובע מהאופי האיטרטיבי של Gradient Descent, הדורש איטרציות רבות כדי להגיע להתכנס, במיוחד עבור פונקציות מורכבות או מערכי נתונים גדולים. וזאת לעומת curve\_fit, הנהנה מאלגוריתמי אופטימיזציה מיוחדים יותר שלעתים קרובות יעילים יותר במציאת הפרמטרים האופטימליים בפחות איטרציות.

שאלה 3

1. TRUE:

1. (p1 and not p2) or (p1 and not p4)

2. (not p3 and not p2)

3. p1

FALSE:

1. p5 or p2

2. (p2 and p8)or p5

3. (P9 and p5) or p3 and p7)) or p2 and p6))

1. נחשב את מספר הכללים שמסווגים נכונה את סט האימון:

גודל המטריצה הוא – 3 על 3. ולכן, מספר המטריצות האפשריות הוא: .

סט האימון שלנו מכיל 6 מטריצות מתוך כל המטריצות (מתוך 512 מטריצות). לכן, סט המבחן יכיל 506 מטריצות ().

יש לנו שתי תשובות אפשריות – TRUE או FALSE. לכן, כל מטריצה יכולה להיות מסווגת לאחת מן התשובות – TRUE או FALSE.

* + *מספר הכללים שמסווגים את סט האימון הוא: .*