IMPORT DEPEDENCIES

In [1]:
```
from pwp.utils import Preprocessing,Visualization,\
     IdealFunction, SqliteSqlalchemy
from bokeh.plotting import figure, output_notebook, show
import pandas as pd
import numpy as np
```

Import train.csv, test.csv, and ideal.csv files

In [2]:
```
# import train.csv, test.csv, and ideal.csv files
# Create an object of the class Preprocessing
prep = Preprocessing()
train = prep.import_dataset('/home/kgyasi/Documents/iubh-pwp/train .csv')  # reads in tr
test = prep.import_dataset('/home/kgyasi/Documents/iubh-pwp/test.csv')  # reads in the t
ideal = prep.import_dataset('/home/kgyasi/Documents/iubh-pwp/ideal.csv')  # reads in the
```

In [3]:
```
# preview train
train.head(3)
```

Out[3]:

	x	y1	y2	y3	y4
0	-20.0	4.947051	9.139399	-0.025432	4.760395
1	-19.9	5.157783	9.082572	-0.103177	4.512951
2	-19.8	5.923819	9.407426	0.425233	4.669478

In [4]:
```
# preview test
test.head(3)
```

Out[4]:

	x	y
0	6.6	6.777071
1	6.7	2.097181
2	18.6	4.626871

In [5]:
```
# preview ideal
print(ideal.head(2))
print(ideal.shape)
```

```
      x        y1        y2        y3        y4        y5        y6        y7  \
0 -20.0 -0.912945  0.408082  9.087055  5.408082 -9.087055  0.912945 -0.839071
1 -19.9 -0.867644  0.497186  9.132356  5.497186 -9.132356  0.867644 -0.865213

         y8        y9  ...        y41       y42       y43       y44  \
0 -0.850919  0.816164  ... -40.456474  40.20404  2.995732 -0.008333
1  0.168518  0.994372  ... -40.233820  40.04859  2.990720 -0.008340

         y45       y46       y47       y48       y49       y50
0  12.995732  5.298317 -5.298317 -0.186278  0.912945  0.396850
1  12.990720  5.293305 -5.293305 -0.215690  0.867644  0.476954

[2 rows x 51 columns]
(400, 51)
```

Get dependent and independent variables

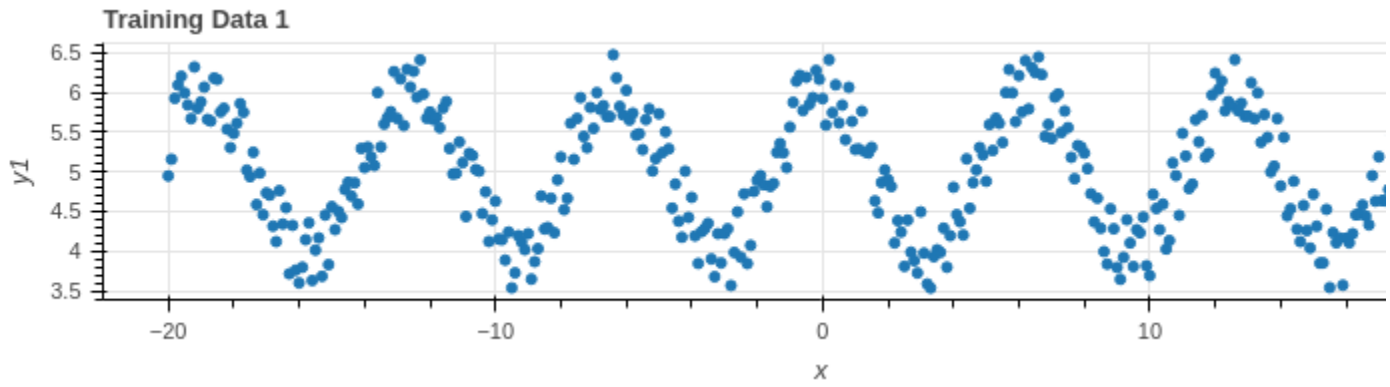
In [6]:
```
# Split dataframe into dependent(X) and independent(Y) values
X_train, Y_train = Preprocessing.values(train)
X_test, Y_test = Preprocessing.values(test)
X_ideal, Y_ideal = Preprocessing.values(ideal)
```

Visualise the Training data and test data

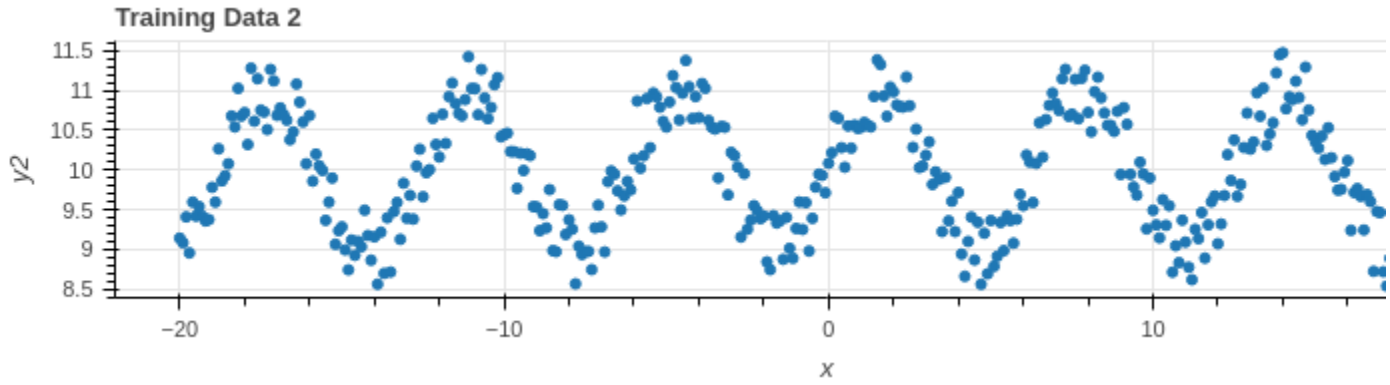
In [7]:
```
visual = Visualization()
```

In [8]:
```
# Visualise train data
number = 1
for i in train.iloc[:, 1:]:
    visual.plot_bokeh("x", i, f"Training Data {number}", train)
    number +=1
```

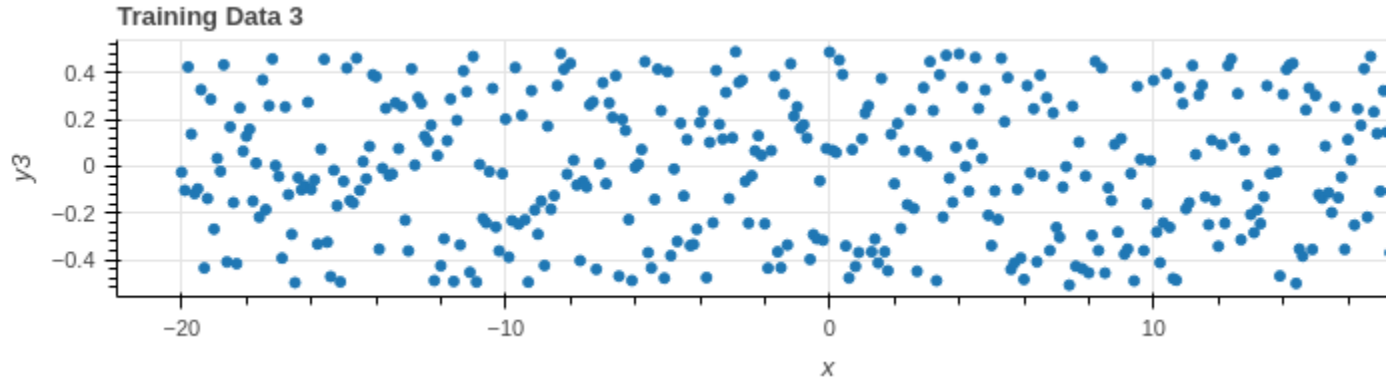
Loading BokehJS ...



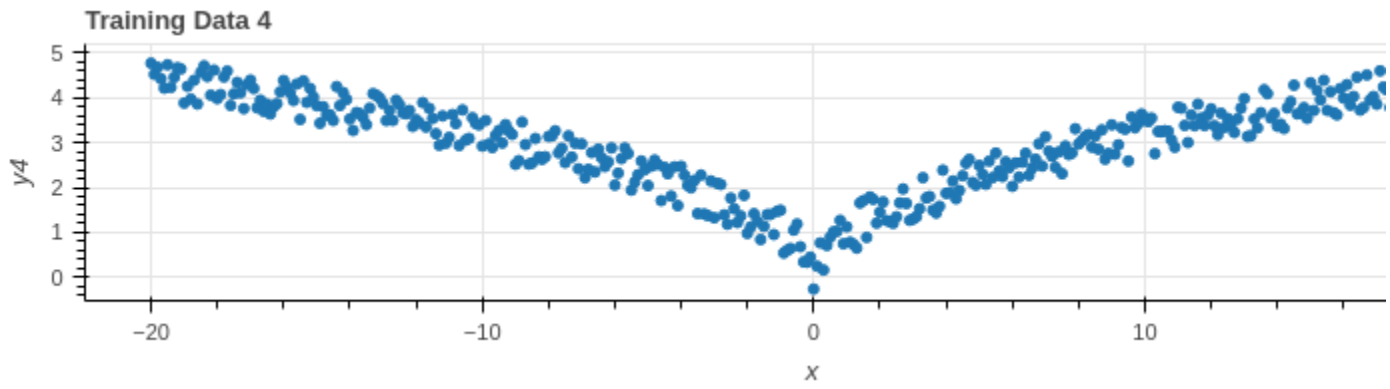
Loading BokehJS ...



Loading BokehJS ...

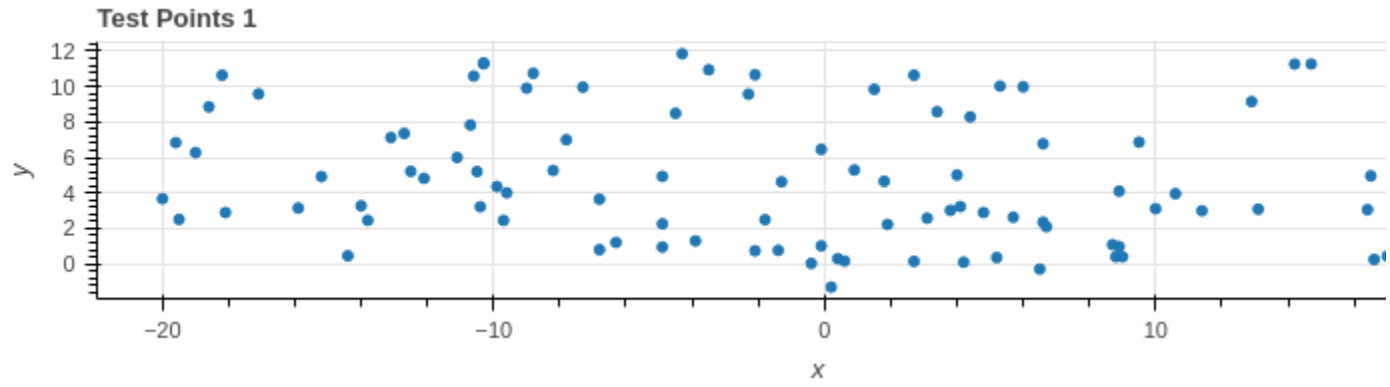


Loading BokehJS ...



In [9]:
```
# Visualise test data
number = 1
for i in test.iloc[:, 1:]:
    visual.plot_bokeh("x", i, f"Test Points {number}", test)
    number +=1
```

Loading BokehJS ...



IDEAL FUNCTIONS FOR TRAIN DATA

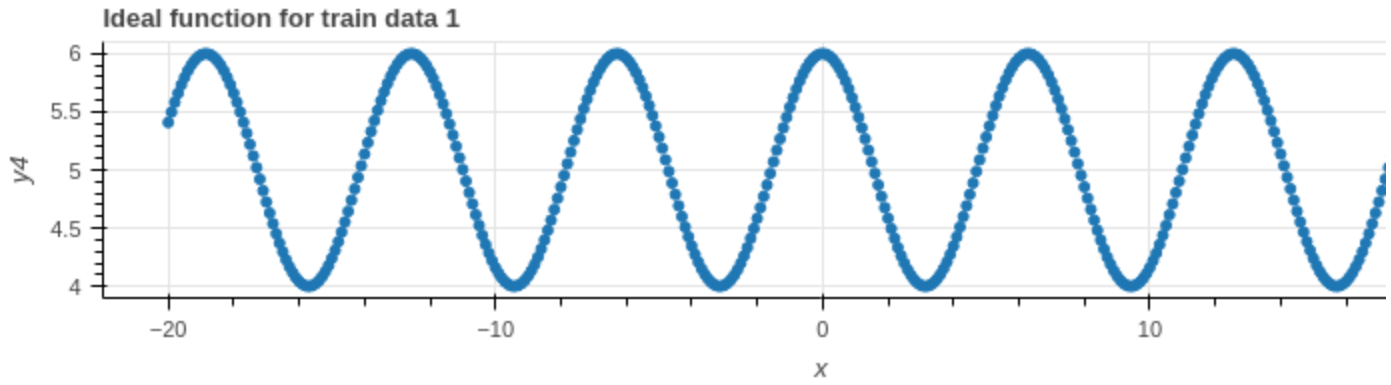
In [10]:
```
# Extract the Y values from the ideal data
Y_ideal_col = Y_ideal.iloc[:].values
# print(Y_ideal_col

# Determine the ideal function for each training data

id_fun = IdealFunction()  # Object of the class ideal function
ideal_fn = list()
train_col = list()
number = 1
for i in Y_train.iloc[:]:
    train_col.append(i)
    lse = id_fun.locate_ideal_fun(Y_train[i], Y_ideal_col, Y_ideal)
    ideal_fn.append(lse)
    print(f'The ideal function of training data {number} is the ideal data column :{lse}
    # Visualize the ideal functions
    visual.plot_bokeh("x", lse, f"Ideal function for train data {number}", ideal)
    number +=1
```

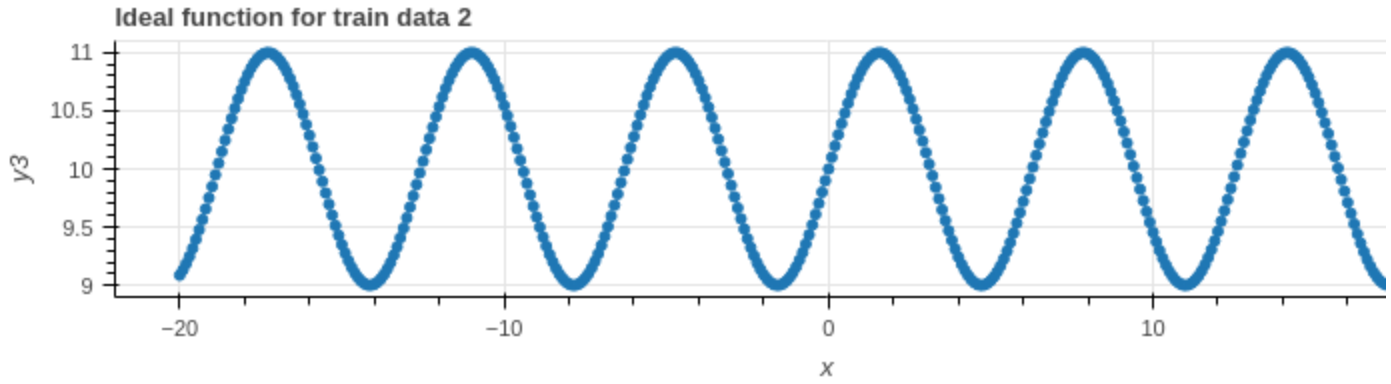
```
The ideal function of training data 1 is the ideal data column :y4
```

Loading BokehJS ...



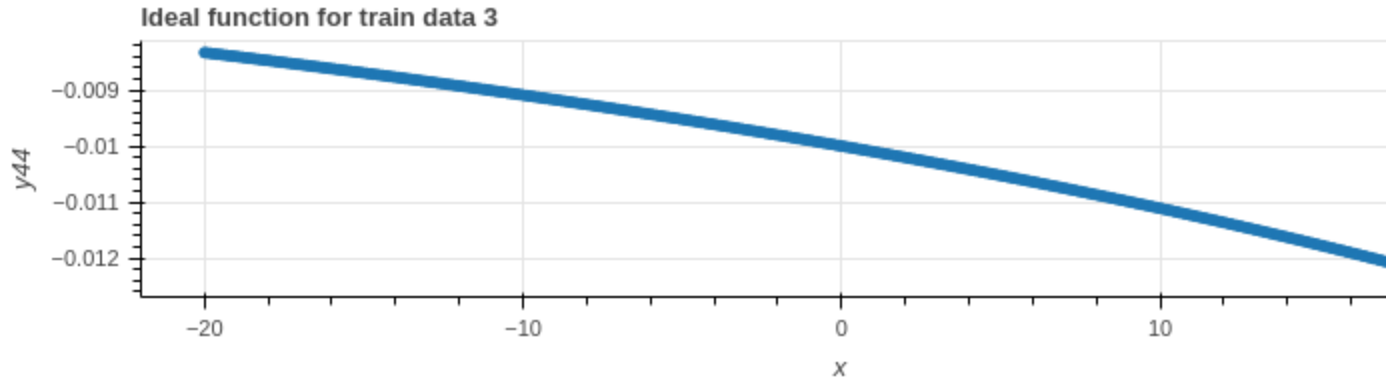

```
The ideal function of training data 2 is the ideal data column :y3
```

Loading BokehJS ...



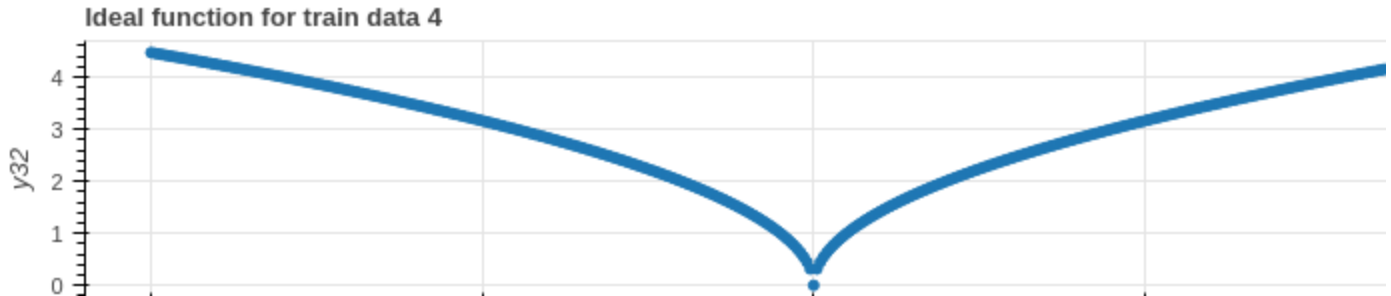

```
The ideal function of training data 3 is the ideal data column :y44
```

Loading BokehJS ...




```
The ideal function of training data 4 is the ideal data column :y32
```

Loading BokehJS ...



Determine the maximum deviation between each training data and its corresponding ideal function

In [11]:
```
max_deviation = list()
for i in range(0, len(ideal_fn)):
    ideal_id = ideal_fn[i]
    train_id = train_col[i]
    max_dev = IdealFunction.max_deviation(Y_train[train_id], Y_ideal[ideal_id])
    max_deviation.append(max_dev)
    print(
        f"The maximum deviation between the train data {i+1} and it's ideal function is:
```

```
The maximum deviation between the train data 1 and it's ideal function is:0.49743
The maximum deviation between the train data 2 and it's ideal function is:0.49632
The maximum deviation between the train data 3 and it's ideal function is:0.49999
The maximum deviation between the train data 4 and it's ideal function is:0.49621
```

Determine the 100 ideal points from the 400 ideal function coordinates for the four ideal functions and calculate the deviation from the test points

Ceate an ideal function table

In [12]:
```
ideal_table = (ideal['x'], ideal['y4'], ideal['y3'], ideal['y44'],  ideal['y32'])
ideal_table = pd.DataFrame(ideal_table).transpose()
ideal_table.head(3)
```

Out[12]:

	x	y4	y3	y44	y32
0	-20.0	5.408082	9.087055	-0.008333	4.472136
1	-19.9	5.497186	9.132356	-0.008340	4.460942
2	-19.8	5.581322	9.186326	-0.008347	4.449719

In [13]:
```
dev, id_table = id_fun.deviations(X_test, X_ideal, ideal_table, Y_test.values)
#print(dev)
#id_table[:4]
```

In [14]:
```
# create a table of the test points and the ideal coordinates
id_table = np.array(id_table).transpose()
id_table = {'x': id_table[0].flatten(),
            'y4': id_table[1].flatten(),
            'y3': id_table[2].flatten(),
            'y44': id_table[3].flatten(),
            'y32': id_table[4].flatten()
            }

#id_table
```

Compare the deviation to the conditional statement

In [15]:
```
label = 1
dif = dict()
for i in range(0, len(max_deviation)):

    d1 = id_fun.dev_mapper(dev[i], max_deviation[i])
```

```
    dif[label]=d1
    label +=1
# # Create a table of the deviations outcome and output the minimum of each row
diff = pd.DataFrame(dif)
# The function with the least deviation will be the most ideal for a test point

diff['min'] = diff.min(axis=1)
dev_min = np.array(diff['min'])  # convert to NumPy array
```

Map the deviations to the ideal functions

In [16]:
```
func = {}
label = 1
for i in range(1, len(ideal_fn)+1):
    f1 = id_fun.func_map(dev_min, dif[i], i)
    func[label] = f1
    label +=1

func = pd.DataFrame(func)
# get the mean
func = func.mean(axis=1)
#print(func)
```

Create a table of the test-data, with mapping and y-deviation

In [17]:
```
df = test
df['delta'] = diff['min']
df['No. of Ideal Function'] = func
df.style.set_table_attributes("style='display:inline'").set_caption('Table of test-data,
```

Out[17]:

Table of test-data, with mapping and y-deviations

	x	y	delta	No. of Ideal Function
0	6.600000	6.777071	nan	nan
1	6.700000	2.097181	0.491255	4.000000
2	18.600000	4.626871	0.314099	4.000000
3	3.400000	8.583755	nan	nan
4	8.800000	0.401559	0.412524	3.000000
5	-12.700000	7.358367	nan	nan
6	-1.300000	4.631897	0.635602	1.000000
7	10.000000	3.112941	0.049337	4.000000
8	-9.900000	4.367376	0.256567	1.000000
9	5.300000	10.026350	nan	nan
10	0.400000	0.296513	0.306553	3.000000
11	6.500000	-0.286687	0.275991	3.000000
12	-19.500000	2.512208	nan	nan
13	16.500000	4.964210	0.666607	1.000000
14	0.600000	0.164798	0.174858	3.000000
15	3.100000	2.580569	nan	nan
16	-8.200000	5.270977	0.610132	1.000000
17	17.400000	9.113753	0.106413	2.000000
18	-4.300000	11.849324	nan	nan

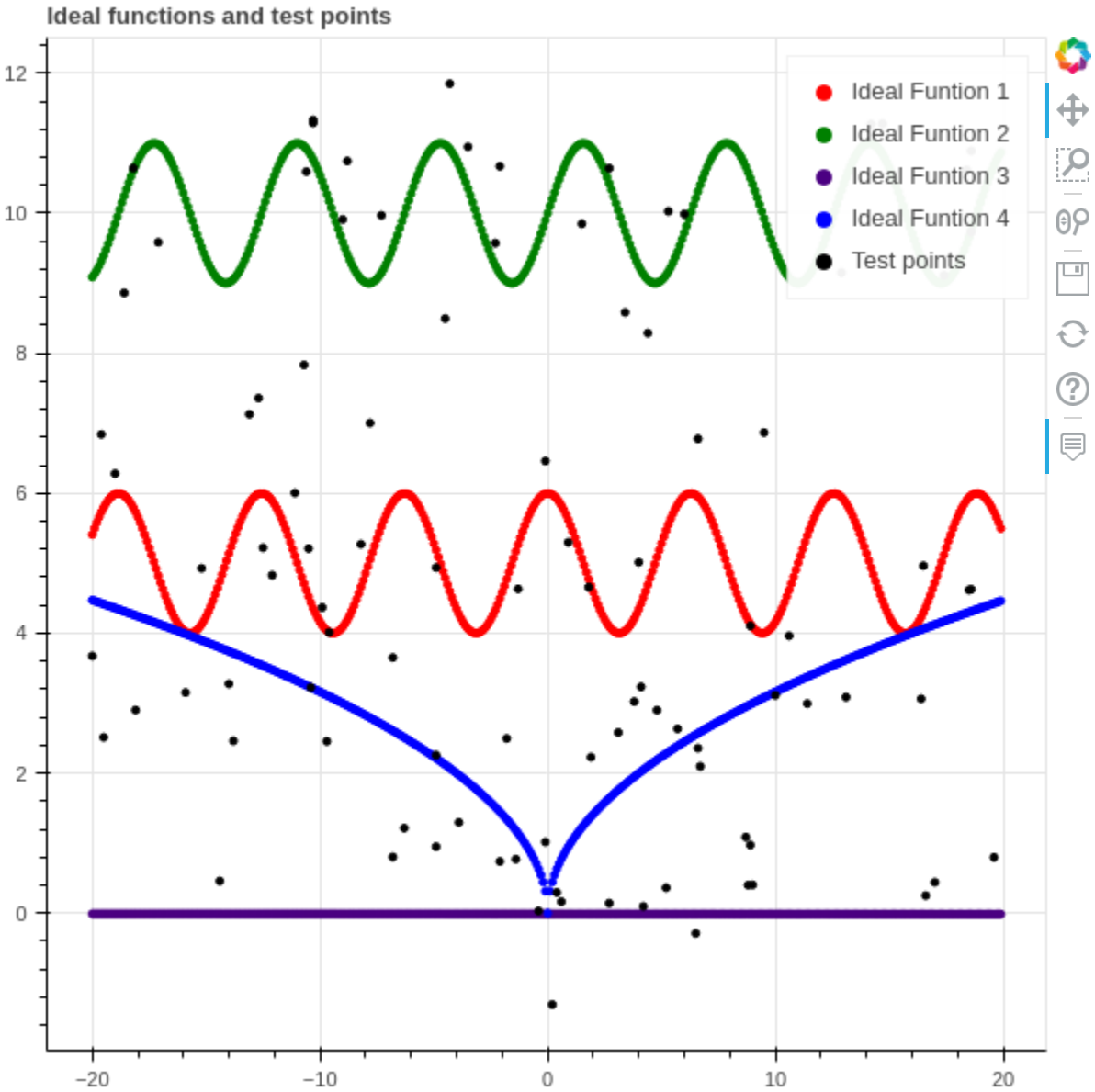
19	6.000000	9.987526	0.266941	2.000000
20	13.100000	3.085916	0.533476	4.000000
21	12.900000	9.152636	nan	nan
22	9.500000	6.865649	nan	nan
23	5.700000	2.633575	0.246108	4.000000
24	16.600000	0.251652	0.263643	3.000000
25	-14.400000	0.459847	0.468588	3.000000
26	4.400000	8.287042	nan	nan
27	-0.400000	0.032540	0.042500	3.000000
28	-4.900000	4.936894	0.249618	1.000000
29	4.800000	2.898996	nan	nan
30	-6.300000	1.215231	nan	nan
31	-10.400000	3.224202	0.000701	4.000000
32	-0.100000	6.460672	0.465668	1.000000
33	18.600000	9.734438	0.018588	2.000000
34	-1.800000	2.495732	nan	nan
35	-3.500000	10.947888	0.597105	2.000000
36	0.900000	5.297898	0.323712	1.000000
37	2.700000	10.641709	0.214329	2.000000
38	4.000000	5.014447	0.668091	1.000000
39	-19.000000	6.279578	0.290873	1.000000
40	-2.300000	9.572986	0.318692	2.000000
41	-8.800000	10.744899	nan	nan
42	-7.300000	9.968316	nan	nan
43	1.500000	9.848586	nan	nan
44	-7.800000	7.002672	nan	nan
45	-10.600000	10.592483	0.330292	2.000000
46	3.800000	3.023217	nan	nan
47	18.400000	10.616735	nan	nan
48	-12.100000	4.828303	nan	nan
49	-6.800000	3.651482	nan	nan
50	-1.400000	0.771451	0.411766	4.000000
51	-12.500000	5.220581	nan	nan
52	-10.700000	7.832296	nan	nan
53	-13.800000	2.461168	nan	nan
54	-18.100000	2.901198	nan	nan
55	-14.000000	3.275950	0.465708	4.000000
56	8.700000	1.087246	nan	nan
57	19.600000	0.798555	nan	nan
58	4.100000	3.233759	nan	nan

59	-15.200000	4.927066	nan	nan
60	14.200000	11.271268	0.273241	2.000000
61	8.900000	0.974458	nan	nan
62	-2.100000	0.739598	nan	nan
63	-18.600000	8.860240	nan	nan
64	18.500000	4.616625	0.315462	4.000000
65	-9.600000	4.011017	0.004295	1.000000
66	1.800000	4.657360	0.115438	1.000000
67	-10.500000	5.209053	0.684590	1.000000
68	-10.300000	11.330760	0.563074	2.000000
69	-3.900000	1.296653	0.678188	4.000000
70	0.200000	-1.305003	nan	nan
71	-18.200000	10.643221	0.038388	2.000000
72	10.600000	3.961990	0.652672	1.000000
73	-2.100000	10.669963	nan	nan
74	-9.000000	9.910162	0.322281	2.000000
75	1.900000	2.228912	nan	nan
76	14.700000	11.277069	0.431322	2.000000
77	-13.100000	7.127266	nan	nan
78	-11.100000	6.003875	nan	nan
79	-4.900000	0.950870	nan	nan
80	8.900000	4.103986	0.030579	1.000000
81	18.600000	10.889785	nan	nan
82	-17.100000	9.584914	nan	nan
83	-0.100000	1.017679	0.701451	4.000000
84	-15.900000	3.152485	nan	nan
85	5.200000	0.363506	0.374055	3.000000
86	-6.800000	0.803535	nan	nan
87	-4.900000	2.259100	0.045505	4.000000
88	9.000000	0.404932	0.415921	3.000000
89	17.000000	0.442203	0.454251	3.000000
90	-9.700000	2.455544	0.658938	4.000000
91	-10.300000	11.288804	0.521118	2.000000
92	-4.500000	8.494228	nan	nan
93	4.200000	0.097929	0.108367	3.000000
94	-20.000000	3.673985	nan	nan
95	6.600000	2.356898	0.212149	4.000000
96	11.400000	2.995157	0.381232	4.000000
97	2.700000	0.142764	0.153041	3.000000
98	-19.600000	6.840677	nan	nan
99	16.400000	3.061969	nan	nan

visualize the test points, the ideal functions

In [18]:
```
# define tooltips for bokeh
TOOLTIPS = [
("index", "$index"),
("(x,y)", "($x, $y)")]
# plot
p = figure(title='Ideal functions and test points', tooltips=TOOLTIPS)
p.circle(ideal['x'], ideal['y4'], color='red', legend_label='Ideal Funtion 1')
p.circle(ideal['x'], ideal['y3'], color='green', legend_label='Ideal Funtion 2')
p.circle(ideal['x'], ideal['y44'], color='indigo', legend_label='Ideal Funtion 3')
p.circle(ideal['x'], ideal['y32'], color='blue', legend_label='Ideal Funtion 4')
p.circle(test['x'], test['y'], color='black', legend_label='Test points')
p.legend.location = "top_right"
output_notebook()
show(p)
```

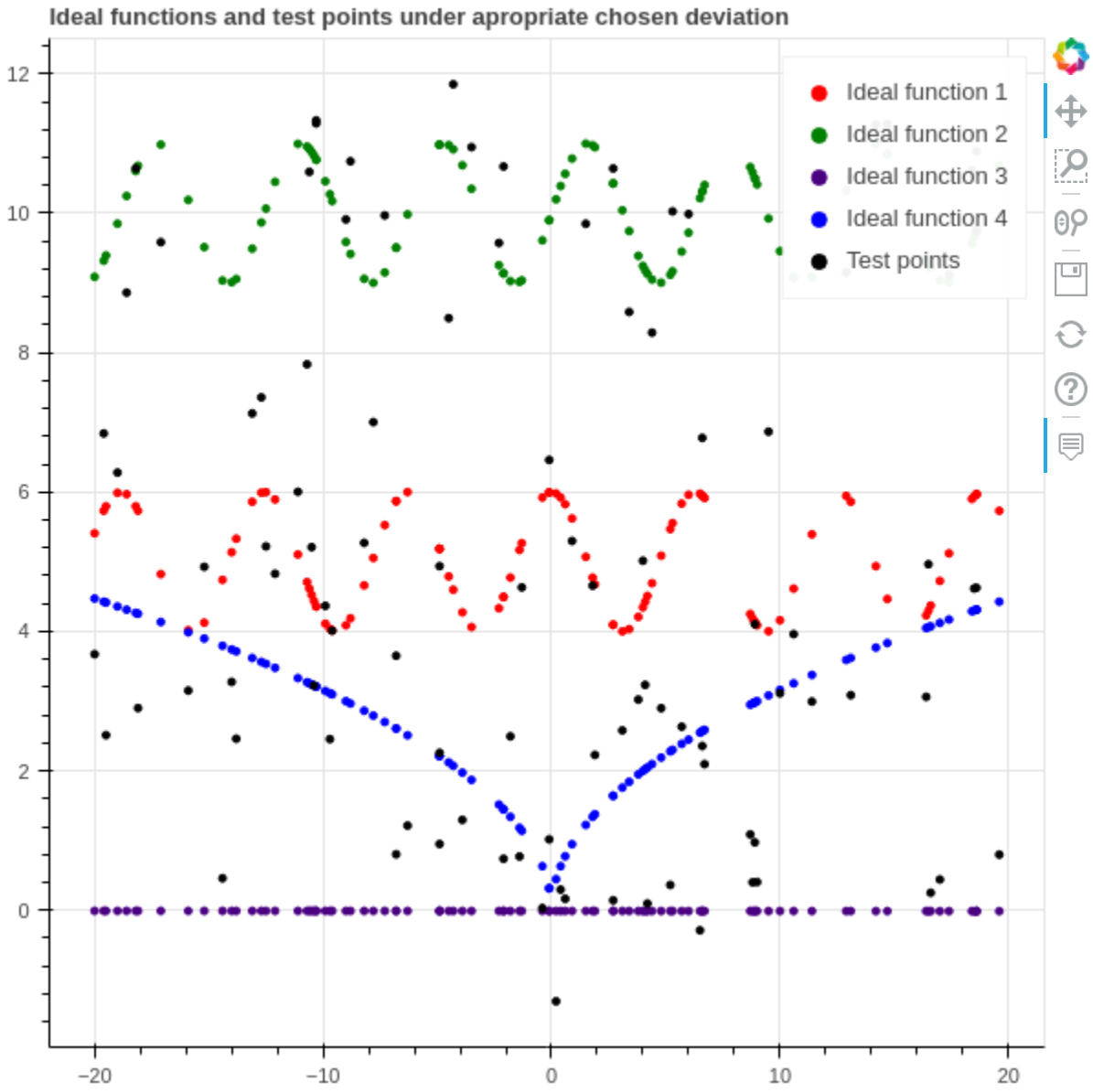
Loading BokehJS ...



In [19]:
```
p = figure(title='Ideal functions and test points under apropriate chosen deviation',
tooltips=TOOLTIPS)
p.circle(id_table['x'], id_table['y4'], color='red', legend_label='Ideal function 1')
p.circle(id_table['x'], id_table['y3'], color='green', legend_label='Ideal function 2')
```

```
p.circle(id_table['x'], id_table['y44'], color='indigo', legend_label='Ideal function 3')
p.circle(id_table['x'], id_table['y32'], color='blue', legend_label='Ideal function 4')
p.circle(test['x'], test['y'], color='black', legend_label='Test points')
p.legend.location = "top_right"
output_notebook()
show(p)
```

BokehJS 2.4.1 successfully loaded.



Save final table to the current directory

In [20]:
```
df.to_csv(r'./final_table.csv', index=False, header=True)
```

Importing CSV files into SQLite database assignment.db

In [21]:
```
sqlite_sqlalchemy = SqliteSqlalchemy() # object of the class SqliteSqlalchemy
# Import the train.csv
sqlite_sqlalchemy.import_csv_table("train .csv", "Table 1: The training data's database
"assignment")
# Import the idealfunction table
sqlite_sqlalchemy.import_csv_table("ideal.csv", "Table 2: The ideal functions' database
"assignment")
```

```
# Import the database table of the test-data, with mapping and y-deviation
sqlite_sqlalchemy.import_csv_table("final_table.csv", "Table 3:The database table of the
```

```
2023-01-21 19:05:02,226 INFO sqlalchemy.engine.Engine PRAGMA main.table_info("Table 1: T
he training data's database table")
2023-01-21 19:05:02,227 INFO sqlalchemy.engine.Engine [raw sql] ()
2023-01-21 19:05:02,238 INFO sqlalchemy.engine.Engine BEGIN (implicit)
2023-01-21 19:05:02,249 INFO sqlalchemy.engine.Engine INSERT INTO "Table 1: The training
data's database table" ("index", x, y1, y2, y3, y4) VALUES (?, ?, ?, ?, ?, ?)
2023-01-21 19:05:02,251 INFO sqlalchemy.engine.Engine [generated in 0.00791s] ((0, -20.
0, 4.9470515, 9.139399, -0.025432454, 4.7603946), (1, -19.9, 5.1577826, 9.082572, -0.103
176646, 4.512951), (2, -19.8, 5.923819, 9.407426, 0.42523292, 4.6694784), (3, -19.7, 6.0
915623, 8.953618, 0.13769506, 4.414474), (4, -19.6, 6.2020335, 9.593519, -0.11682482, 4.
2050943), (5, -19.5, 5.9927864, 9.417254, -0.096975856, 4.724585), (6, -19.4, 5.8405395,
9.537546, 0.32761803, 4.2226596), (7, -19.3, 5.668819, 9.415276, -0.43552196, 4.448568)
... displaying 10 of 400 total bound parameter sets ...  (398, 19.8, 5.598388, 10.54493
4, -0.162114, 4.492192), (399, 19.9, 5.7651916, 10.942644, 0.36237967, 4.016596))
2023-01-21 19:05:02,305 INFO sqlalchemy.engine.Engine COMMIT
2023-01-21 19:05:02,657 INFO sqlalchemy.engine.Engine SELECT name FROM sqlite_master WHE
RE type='table' ORDER BY name
2023-01-21 19:05:02,658 INFO sqlalchemy.engine.Engine [raw sql] ()
2023-01-21 19:05:02,696 INFO sqlalchemy.engine.Engine PRAGMA main.table_info("Table 2: T
he ideal functions' database table")
2023-01-21 19:05:02,698 INFO sqlalchemy.engine.Engine [raw sql] ()
2023-01-21 19:05:02,721 INFO sqlalchemy.engine.Engine BEGIN (implicit)
2023-01-21 19:05:02,769 INFO sqlalchemy.engine.Engine INSERT INTO "Table 2: The ideal fu
nctions' database table" ("index", x, y1, y2, y3, y4, y5, y6, y7, y8, y9, y10, y11, y12,
y13, y14, y15, y16, y17, y18, y19, y20, y21, y22, y23, y24, y25, y26, y27, y28, y29, y3
0, y31, y32, y33, y34, y35, y36, y37, y38, y39, y40, y41, y42, y43, y44, y45, y46, y47,
y48, y49, y50) VALUES (?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?,
?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?,
?)
2023-01-21 19:05:02,770 INFO sqlalchemy.engine.Engine [generated in 0.03224s] ((0, -20.
0, -0.9129453, 0.40808207, 9.087055, 5.408082, -9.087055, 0.9129453, -0.8390715, -0.8509
1937, 0.81616414, 18.258905, -20.0, -58.0, -45.0, 20.0, 13.0, 400.0, -400.0, 800.0, 410.
0, 289.0, -8000.0, 8000.0, 8000.0, -16000.0, -23995.0, -5832.0, 10648.0, -8020.0, -7600.
0, -8795.0, 20.0, 4.472136, 20.12461, -0.7464143, 10.0, 100.0, -20.0, -1.3210273, 399.08
707, 899.5919, -40.456474, 40.20404, 2.9957323, -0.008333334, 12.995732, 5.2983174, -5.2
983174, -0.18627828, 0.9129453, 0.3968496), (1, -19.9, -0.8676441, 0.4971858, 9.132356,
5.4971857, -9.132356, 0.8676441, -0.8652126, 0.16851768, 0.9943716, 17.266117, -19.9, -5
7.7, -44.8, 19.9, 12.95, 396.01, -396.01, 792.02, 406.01, 285.61, -7880.599, 7880.599, 7
880.599, -15761.198, -23636.797, -5735.339, 10503.459, -7900.499, -7484.589, -8667.619,
19.9, 4.460942, 20.025234, -0.6204504, 9.9, 99.5, -19.9, -1.3648299, 395.14236, 893.512
8, -40.23382, 40.04859, 2.9907198, -0.008340283, 12.99072, 5.293305, -5.293305, -0.21569
017, 0.8676441, 0.47695395), (2, -19.8, -0.81367373, 0.58132184, 9.186326, 5.5813217, -
9.186326, 0.81367373, -0.88919115, 0.6123911, 1.1626437, 16.11074, -19.8, -57.4, -44.6,
19.8, 12.9, 392.04, -392.04, 784.08, 402.04, 282.24, -7762.392, 7762.392, 7762.392, -155
24.784, -23282.176, -5639.752, 10360.232, -7782.192, -7370.352, -8541.472, 19.8, 4.44971
9, 19.925863, -0.47573867, 9.8, 99.0, -19.8, -1.3949956, 391.22632, 887.4587, -40.00683
6, 39.89066, 2.985682, -0.008347246, 12.985682, 5.288267, -5.288267, -0.23650314, 0.8136
7373, 0.5491291), (3, -19.7, -0.75157344, 0.65964943, 9.248426, 5.6596494, -9.248426, 0.
75157344, -0.91094714, -0.99466854, 1.3192989, 14.805996, -19.7, -57.1, -44.4, 19.7, 12.
85, 388.09, -388.09, 776.18, 398.09, 278.89, -7645.373, 7645.373, 7645.373, -15290.746,
-22931.12, -5545.233, 10218.313, -7665.073, -7257.283, -8416.553, 19.7, 4.438468, 19.826
498, -0.31643602, 9.7, 98.5, -19.7, -1.4112228, 387.33844, 881.43036, -39.775787, 39.729
824, 2.9806187, -0.008354219, 12.9806185, 5.2832036, -5.2832036, -0.24788749, 0.7515734
4, 0.6128399), (4, -19.6, -0.6819636, 0.7313861, 9.318036, 5.731386, -9.318036, 0.681963
6, -0.9304263, 0.7743557, 1.4627723, 13.366487, -19.6, -56.8, -44.2, 19.6, 12.8, 384.16,
-384.16, 768.32, 394.16, 275.56, -7529.536, 7529.536, 7529.536, -15059.072, -22583.607,
-5451.776, 10077.696, -7549.136, -7145.376, -8292.856, 19.6, 4.427189, 19.727139, -0.147
038, 9.6, 98.0, -19.6, -1.4133497, 383.47803, 875.4286, -39.54098, 39.565693, 2.9755297,
-0.008361204, 12.97553, 5.278115, -5.278115, -0.24938935, 0.6819636, 0.6679019), (5, -1
9.5, -0.60553986, 0.795815, 9.39446, 5.795815, -9.39446, 0.60553986, -0.9475798, -0.1170
2018, 1.59163, 11.808027, -19.5, -56.5, -44.0, 19.5, 12.75, 380.25, -380.25, 760.5, 390.
25, 272.25, -7414.875, 7414.875, 7414.875, -14829.75, -22239.625, -5359.375, 9938.375, -
7434.375, -7034.625, -8170.375, 19.5, 4.41588, 19.627787, 0.027781596, 9.5, 97.5, -19.5,
```

```
-1.4013548, 379.64447, 869.45416, -39.30277, 39.397907, 2.9704144, -0.008368201, 12.9704
14, 5.273, -5.273, -0.24094884, 0.60553986, 0.7144341), (6, -19.4, -0.52306575, 0.852292
3, 9.476934, 5.8522925, -9.476934, 0.52306575, -0.96236485, -0.59004813, 1.7045846, 10.1
47476, -19.4, -56.2, -43.8, 19.4, 12.7, 376.36, -376.36, 752.72, 386.36, 268.96, -7301.3
84, 7301.384, 7301.384, -14602.768, -21899.152, -5268.024, 9800.344, -7320.784, -6925.02
4, -8049.104, 19.4, 4.404543, 19.52844, 0.20333645, 9.4, 97.0, -19.4, -1.3753581, 375.83
694, 863.5077, -39.06153, 39.226147, 2.9652731, -0.00837521, 12.965273, 5.267858, -5.267
858, -0.22290246, 0.52306575, 0.75279135), (7, -19.3, -0.43536535, 0.90025383, 9.564634,
5.900254, -9.564634, 0.43536535, -0.97474456, 0.97776526, 1.8005077, 8.402552, -19.3, -5
5.9, -43.6, 19.3, 12.65, 372.49, -372.49, 744.98, 382.49, 265.69, -7189.057, 7189.057, 7
189.057, -14378.114, -21562.172, -5177.717, 9663.597, -7208.357, -6816.567, -7929.037, 1
9.3, 4.3931766, 19.429102, 0.37509164, 9.3, 96.5, -19.3, -1.3356192, 372.05463, 857.589
7, -38.817684, 39.050125, 2.9601052, -0.00838223, 12.960105, 5.26269, -5.26269, -0.19596
967, 0.43536535, 0.7834847)  ... displaying 10 of 400 total bound parameter sets ...  (3
98, 19.8, 0.81367373, 0.58132184, 10.813674, 5.5813217, -10.813674, -0.81367373, -0.8891
9115, 0.6123911, 1.1626437, 16.11074, 19.8, 61.4, 34.6, -19.8, -6.9, 392.04, -392.04, 78
4.08, 402.04, 519.84, 7762.392, 7762.392, -7762.392, 15524.784, 23292.176, 10360.232, -5
639.752, 7782.192, 8154.432, 6983.312, 19.8, 4.449719, 19.925863, 1.1516088, 9.8, 99.0,
-19.8, 0.23235193, 392.85367, 95.45868, 40.006836, -39.309338, 2.985682, -0.012468828, 1
2.985682, 5.288267, -5.288267, 0.23650314, 0.81367373, 0.5491291), (399, 19.9, 0.867644
1, 0.4971858, 10.867644, 5.4971857, -10.867644, -0.8676441, -0.8652126, 0.16851768, 0.99
43716, 17.266117, 19.9, 61.7, 34.8, -19.9, -6.95, 396.01, -396.01, 792.02, 406.01, 524.4
1, 7880.599, 7880.599, -7880.599, 15761.198, 23646.797, 10503.459, -5735.339, 7900.499,
8276.609, 7093.579, 19.9, 4.460942, 20.025234, 1.1148378, 9.9, 99.5, -19.9, 0.3704583, 3
96.87766, 97.51282, 40.23382, -39.551407, 2.9907198, -0.012484395, 12.99072, 5.293305, -
5.293305, 0.21569017, 0.8676441, 0.47695395))
2023-01-21 19:05:02,817 INFO sqlalchemy.engine.Engine COMMIT
2023-01-21 19:05:03,073 INFO sqlalchemy.engine.Engine SELECT name FROM sqlite_master WHE
RE type='table' ORDER BY name
2023-01-21 19:05:03,076 INFO sqlalchemy.engine.Engine [raw sql] ()
2023-01-21 19:05:03,090 INFO sqlalchemy.engine.Engine PRAGMA main.table_info("Table 3:Th
e database table of the test-data,with mapping and y-deviation")
2023-01-21 19:05:03,092 INFO sqlalchemy.engine.Engine [raw sql] ()
2023-01-21 19:05:03,098 INFO sqlalchemy.engine.Engine BEGIN (implicit)
2023-01-21 19:05:03,102 INFO sqlalchemy.engine.Engine INSERT INTO "Table 3:The database
table of the test-data,with mapping and y-deviation" ("index", x, y, delta, "No. of Idea
l Function") VALUES (?, ?, ?, ?, ?)
2023-01-21 19:05:03,104 INFO sqlalchemy.engine.Engine [generated in 0.00319s] ((0, 6.6,
6.7770715, None, None), (1, 6.7, 2.0971808, 0.4912552000000003, 4.0), (2, 18.6, 4.62687
1, 0.3140990000000005, 4.0), (3, 3.4, 8.583755, None, None), (4, 8.8, 0.4015592, 0.41252
4112, 3.0), (5, -12.7, 7.3583674, None, None), (6, -1.3, 4.6318974, 0.6356016000000002,
1.0), (7, 10.0, 3.112941, 0.0493367, 4.0)  ... displaying 10 of 100 total bound paramete
r sets ...  (98, -19.6, 6.840677, None, None), (99, 16.4, 3.0619688, None, None))
2023-01-21 19:05:03,110 INFO sqlalchemy.engine.Engine COMMIT
2023-01-21 19:05:03,340 INFO sqlalchemy.engine.Engine SELECT name FROM sqlite_master WHE
RE type='table' ORDER BY name
2023-01-21 19:05:03,341 INFO sqlalchemy.engine.Engine [raw sql] ()
```

In [ ]: