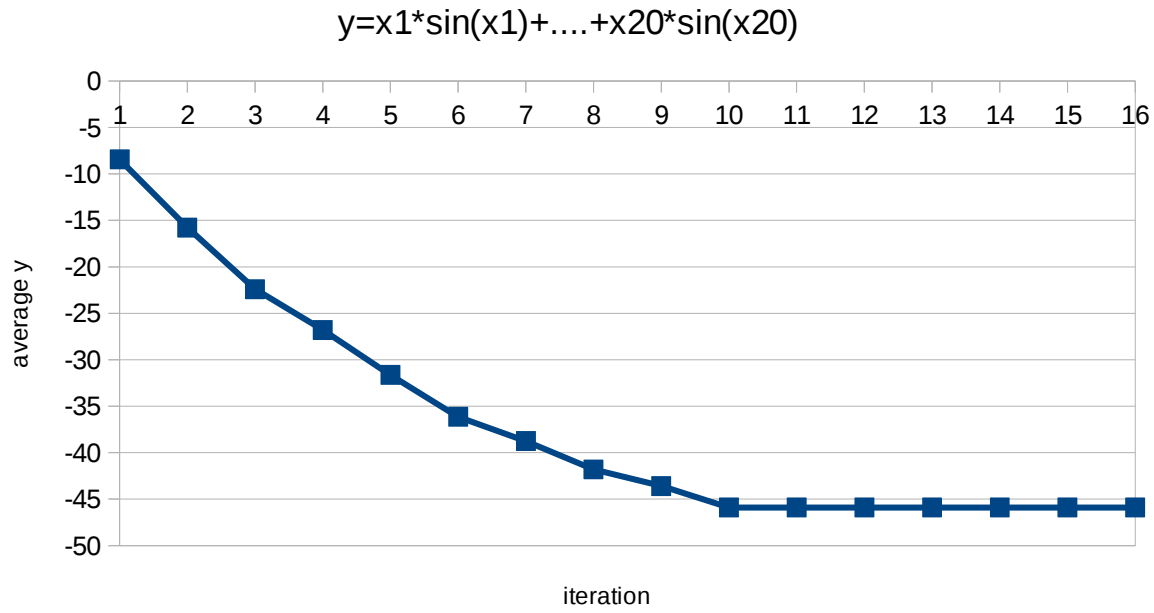
TASK 1

Diagram 1. Average value of function by iteration

in x line we have current iteration, in y line – average value of function

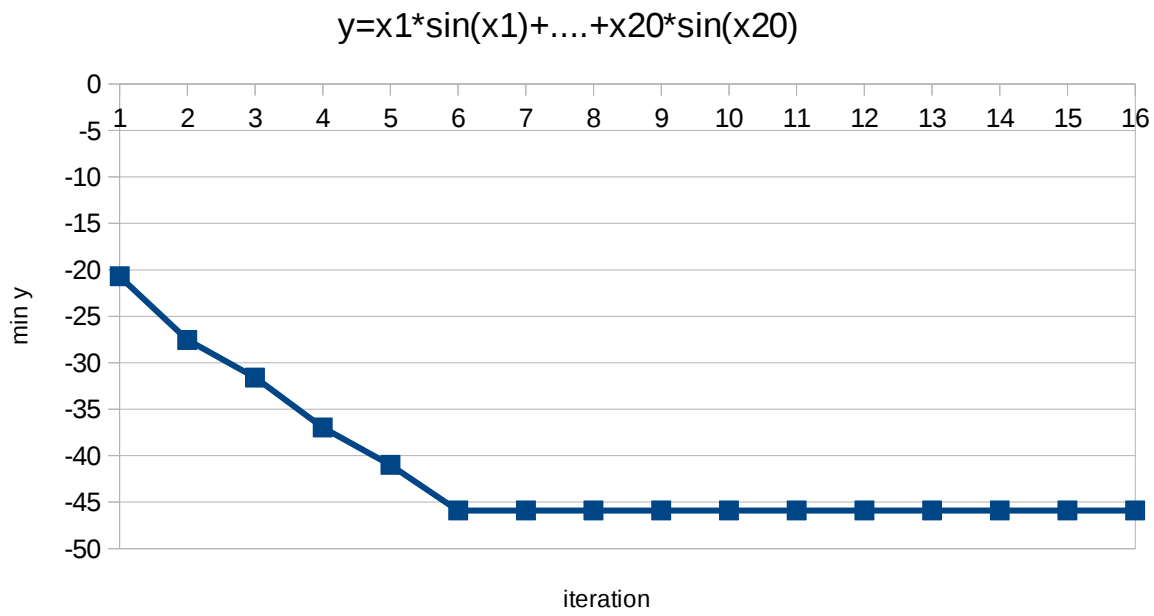


data for diagram:

1	-8.430500144
2	-15.76774386
3	-22.39793034
4	-26.79711635
5	-31.63246185
6	-36.13916085
7	-38.75688102
8	-41.80212619
9	-43.59322746
10	-45.90531033
11	-45.90531033
12	-45.90531033
13	-45.90531033
14	-45.90531033
15	-45.90531033
16	-45.90531033

Diagram 2. Min value of function by iteration

in x line we have current iteration, in y line – average value of function



data for diagram:

1	-20.69901443
2	-27.54180112
3	-31.59034381
4	-36.98150207
5	-40.99472658
6	-45.90531033
7	-45.90531033
8	-45.90531033
9	-45.90531033
10	-45.90531033
11	-45.90531033
12	-45.90531033
13	-45.90531033
14	-45.90531033
15	-45.90531033
16	-45.90531033

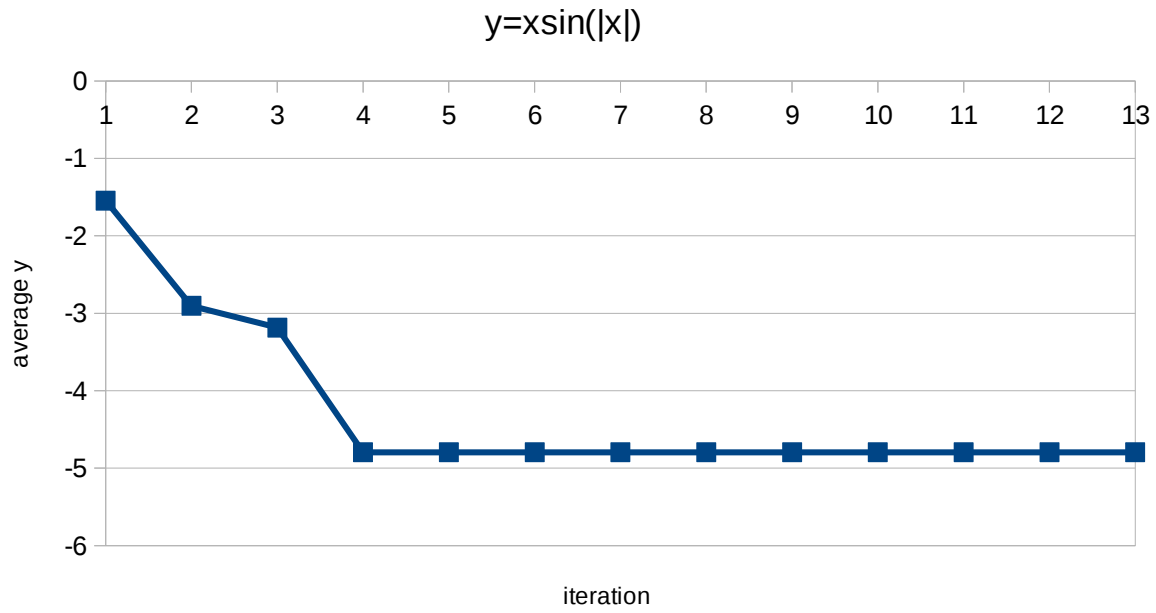
Source code:

```
import random
from math import sin
def get_average_value(generation):
    av_list = []
    for i in generation:
        sum = fitness_function(i)
        av_list.append(sum)
    average = reduce(lambda x, y: x + y, av_list) / len(av_list)
    min_value = min(av_list)
    return (average, min_value)
def fitness_function(i):
    sum = 0
    for j in i:
        sum += j * sin(abs(j))
    return sum
```

```
def main():
    generation = []
    for i in range(0, 20):
        hromosome = []
        for j in range(0, 20):
            hromosome.append(random.uniform(-5, 5))
        generation.append(hromosome)
    min_value = get_average_value(generation)[1]
    print(min_value)
    count = 0
    while count < 10:
        generation.sort(key=lambda x: fitness_function(x), reverse=False)
        fathers = generation[0:10]
        new_generation = []
        for i in range(0, 10):
            mother = random.randrange(0, 10)
            father = random.randrange(0, 10)
            rand_index = random.randrange(0, 20)
            first = fathers[mother][0:rand_index]
            first.extend(fathers[father][rand_index:20])
            second = fathers[father][0:rand_index]
            second.extend(fathers[mother][rand_index:20])
            new_generation.append(first)
            new_generation.append(second)
        new_generation.extend(fathers)
        new_max = get_average_value(new_generation)
        print(new_max)
        if new_max[1] < min_value:
            min_value = new_max[1]
            count = 0
        else:
            count += 1
        generation = new_generation
if __name__ == '__main__':
    main()
```

TASK 2

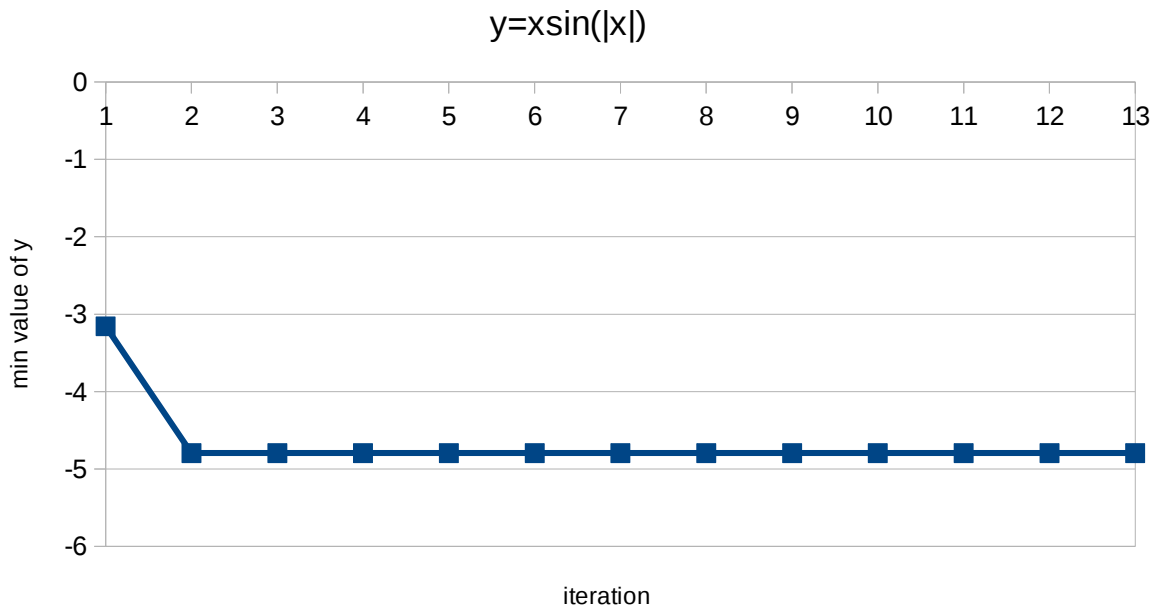
Diagram 2. Average value of function by iteration



data for diagram

1	-1.545168867
2	-2.903980635
3	-3.184621344
4	-4.794621373
5	-4.794621373
6	-4.794621373
7	-4.794621373
8	-4.794621373
9	-4.794621373
10	-4.794621373
11	-4.794621373
12	-4.794621373
13	-4.794621373

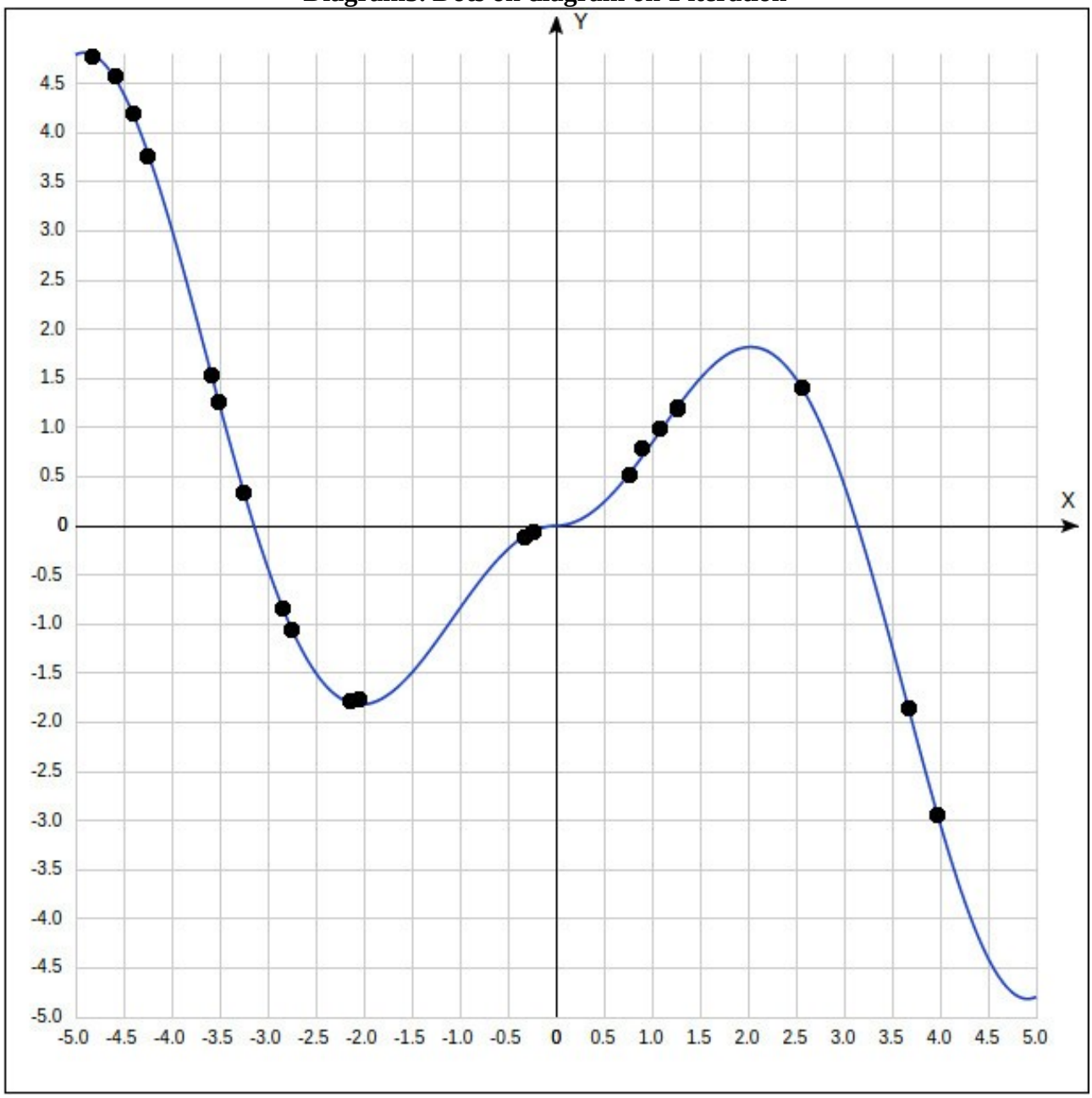
Diagram 2. Min value of function by iteration



data for diagram:

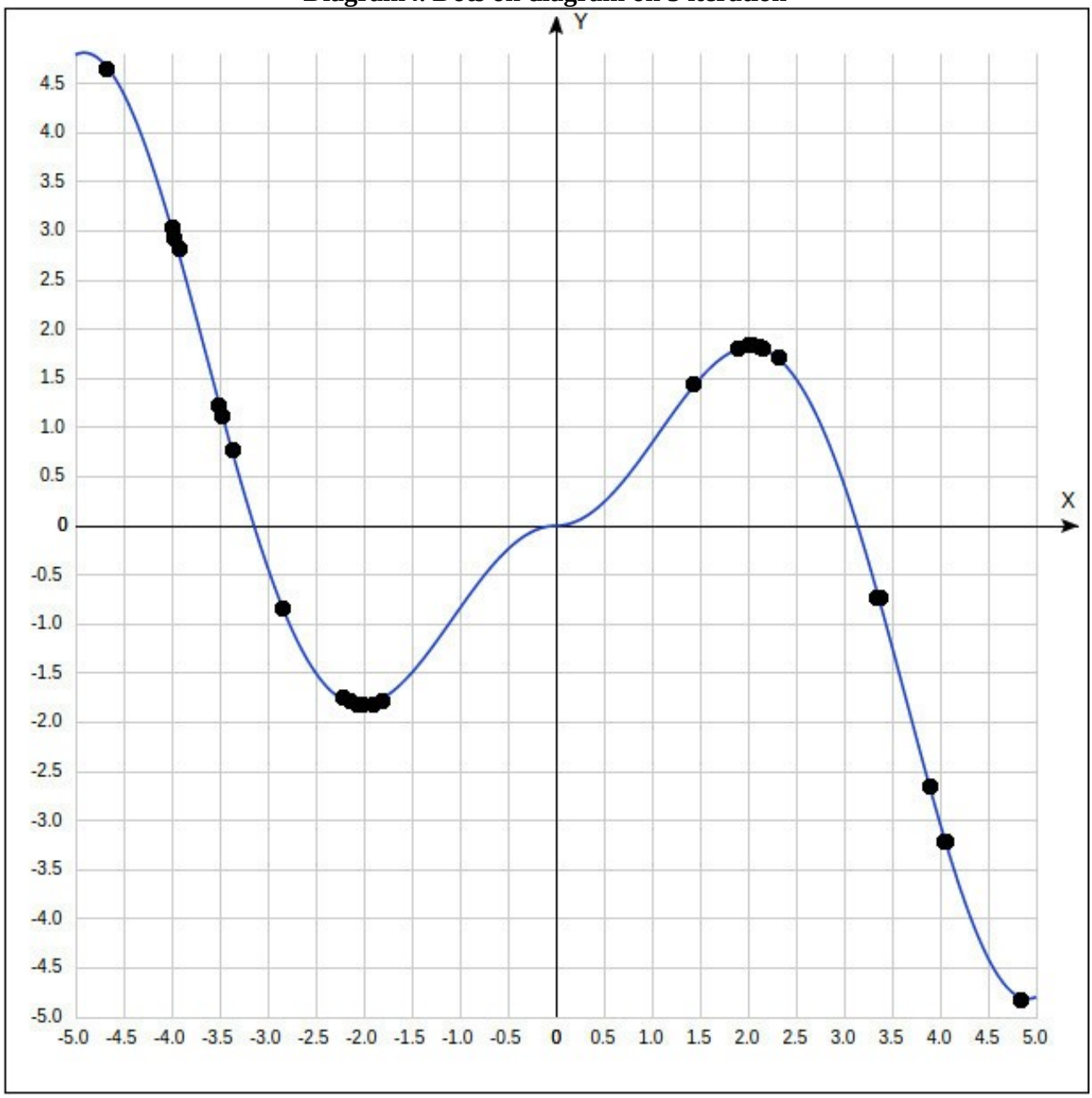
1	-3.156383444
2	-4.794621373
3	-4.794621373
4	-4.794621373
5	-4.794621373
6	-4.794621373
7	-4.794621373
8	-4.794621373
9	-4.794621373
10	-4.794621373
11	-4.794621373
12	-4.794621373
13	-4.794621373

Diagram3. Dots on diagram on 1 iteration



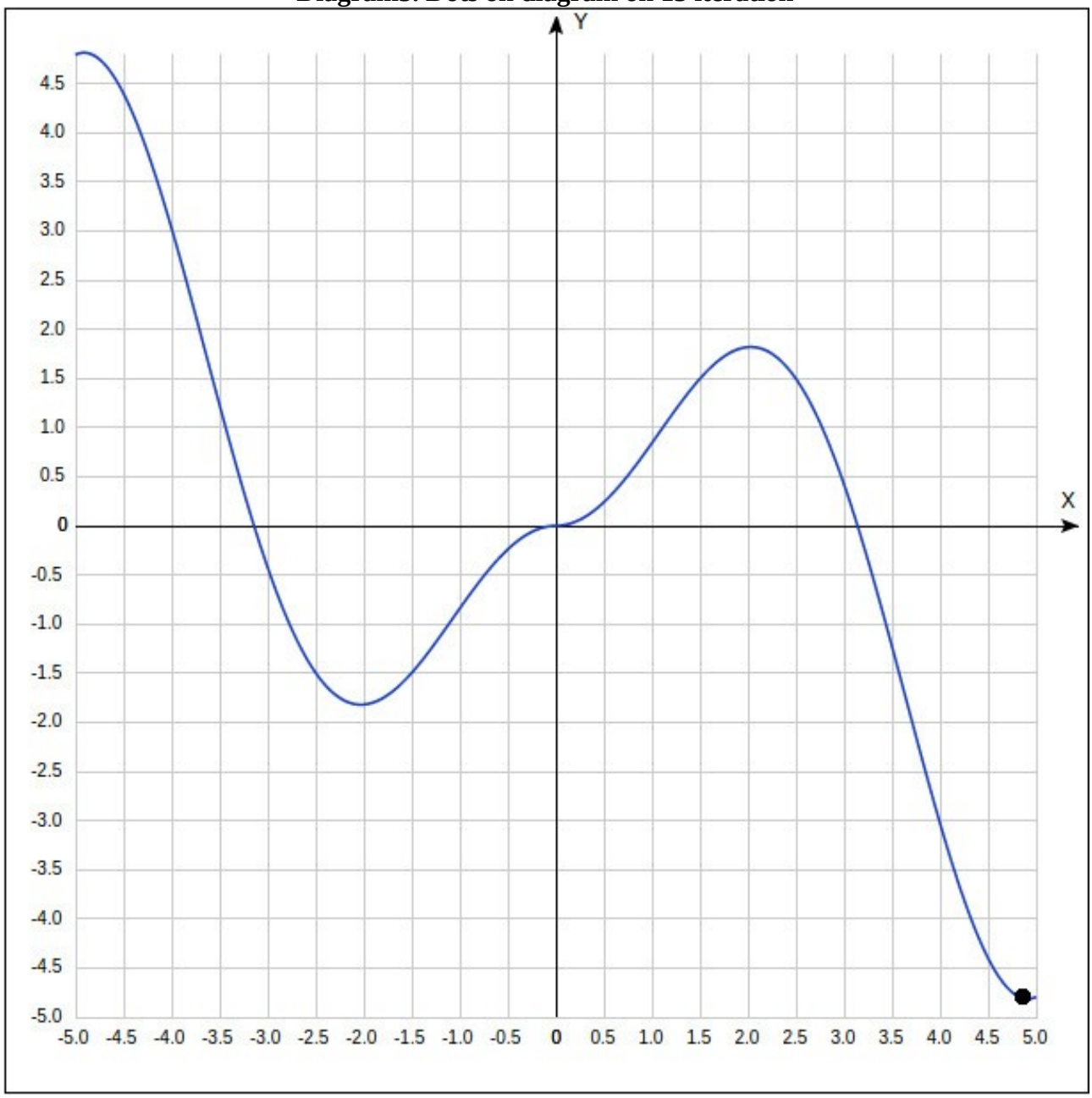
As you can see dots are situated randomly

Diagram4. Dots on diagram on 3 iteration



As you can see a lot of dots are situated in local min (-2.0,-1.78) and only tree dots are situated near by min (4.92435,-4.811223640288648)

Diagram5. Dots on diagram on 13 iteration



As you can see all dots are situated in min of this function by range(-5;5)

Source code:

```
import random
from math import sin
def get_average_value(generation):
    av_list = []
    for i in generation:
        sum = fitness_function(i)
        av_list.append(sum)
    average = reduce(lambda x, y: x + y, av_list) / len(av_list)
    min_value = min(av_list)
    return average, min_value
def fitness_function(i):
    return getResX(i)*sin(abs(getResX(i)))
def getResX(i):
    sum = 0
    for count in range(0,10):
        if i[count] == 1:
            sum += 2**count
```

```
    if i[0] == 1:
        sum *=-1
    return float(sum)/1023*5
def main():
    generation = []
    for i in range(0, 20):
        hromosome = []
        for j in range(0, 10):
            hromosome.append(random.getrandbits(1))
        generation.append(hromosome)
    min_value = get_average_value(generation)[1]
    print(min_value)
    count = 0
    Xs = []
    MAXs = []
    while count < 10:
        Xs.append((map(lambda x: getResX(x), generation)))
        generation.sort(key=lambda x: fitness_function(x), reverse=False)
        fathers = generation[0:10]
        new_generation = []
        for i in range(0, 10):
            mother = random.randrange(0, 10)
            father = random.randrange(0, 10)
            rand_index = random.randrange(0, 10)
            first = fathers[mother][0:rand_index]
            first.extend(fathers[father][rand_index:10])
            second = fathers[father][0:rand_index]
            second.extend(fathers[mother][rand_index:20])
            new_generation.append(first)
            new_generation.append(second)
        new_generation.extend(fathers)
        new_max = get_average_value(new_generation)
        MAXs.append(new_max)
        if new_max[0] < min_value:
            min_value = new_max[0]
            count = 0
        else:
            count += 1
        generation = new_generation
    print(Xs[0])
    print(Xs[2])
    print(Xs[len(Xs)-1])
    print(MAXs)
    print("------------------")
    print(generation[0])
if __name__ == '__main__':
    main()
```