Dawid Jantosz

Signal processing and identification in control of mechatronic devices

Topic: Basic optimization algorithms

Introduction:

Optimization is the process of finding the maximum or minimum values of a quantity, or finding when these maximum or minimum values occurs. This process is used in many areas of life such as path planning, scheduling , modeling, pharmacy and many, many more.

Purpose of the exercise:

In my report I will present basic optimization algorithms using the following functions provided by the LA: nof_2D_fewminima_4 and nof_2D_manyminima_4. Both functions are two-dimensional optimization problems, with the main difference being the number of local minima. The location and value of the global minimum also varies. The solutions are based on the script op_f_RandomSampling.m (A function that optimizes a 2-parameter function using a random algorithm, with visualization of its operation and convergence curve), also provided by LA.

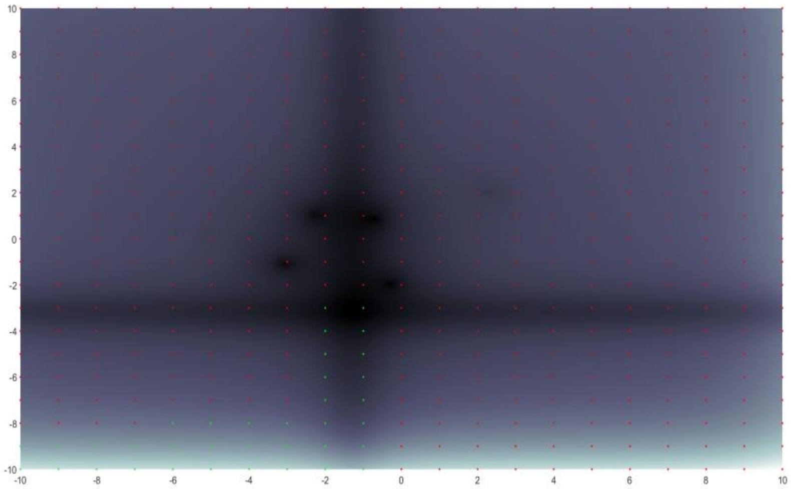
Ex.1 Grid search algorithm

Grid search alghoritm is the first algorithm that we will investigate. The grid search method is the easiest to implement and understand, but sadly not efficient when the number of parameters is large. Grid search involves taking n equally spaced points in desired. This creates a finite number of possible grid points to check. Finally, once each pair of points is calculated, the maximum of these values is chosen.To obtain a grid search algorithm I need to check all the points in nodes of the grid defined by values equally spaced on both parameter axes. The best approach to do that would be to replace while loop with two nested for loops:

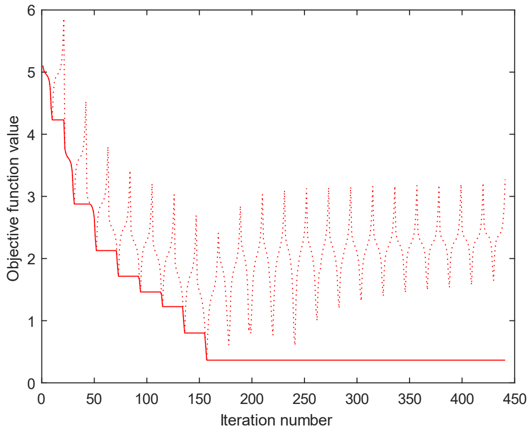
```
step_x = 1;

 for NewX = MaxRangeX(1):step_x:MaxRangeX(2)
   for NewY = MaxRangeY(1):step_x:MaxRangeY(2)
    iter = iter + 1;
```

step_x in the code refers to distance between grid nodes.

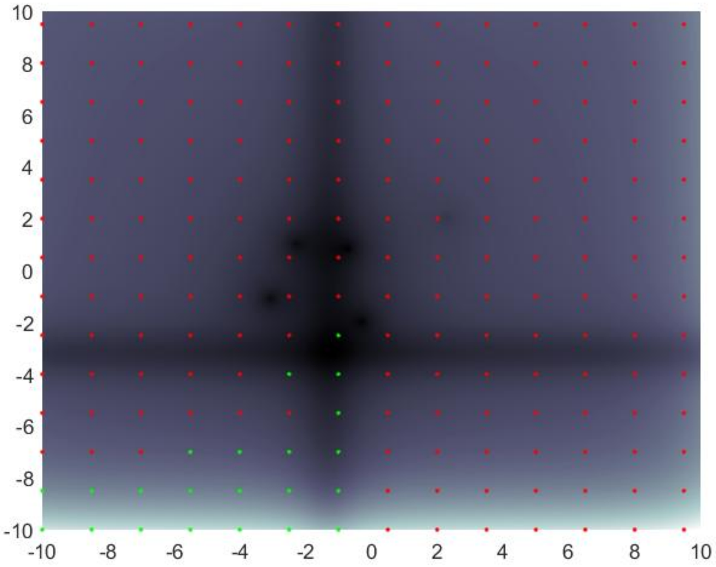
For a full representation of the complexity of grid search alghoritm I will present 3 cases with different value of step. Corresponding results are placed in figures below. Arbitrary values of step that I have chosen are egual to: 0.5, 1 and 1.5. Interpreting the received convergence curves, we can conclude that the smaller value of step is the reguired number of iterations is bigger and solution is more precise. This examples present the characteristics of grid search. As we can see with more complex problems (more dimensional problems), we may be forced to use different aproaches than grid search alghoritm.



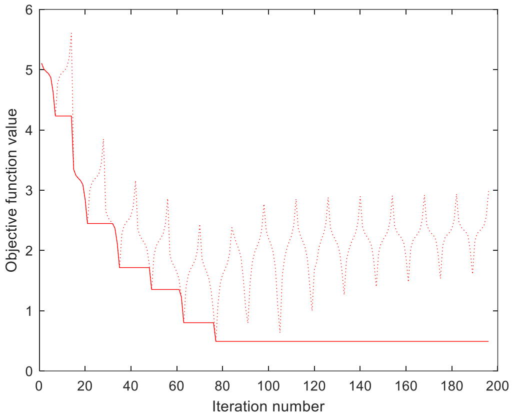
(Fig 1 - example of operation of a Grid Search with step equal to 1)



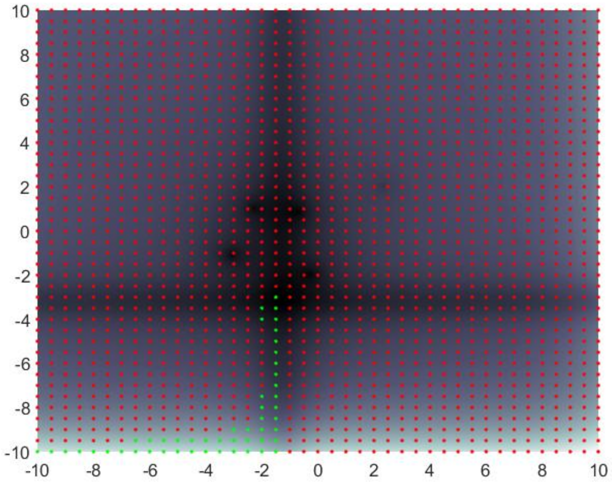
(Convergence curve of a Grid Search with step equal to 1)



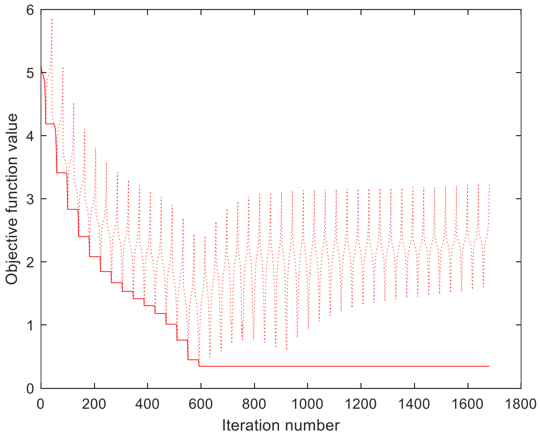
(Fig 2 - example of operation of a Grid Search with step equal to 1.5)



(Convergence curve of a Grid Search with step equal to 1.5)



(Fig 3 - example of operation of a Grid Search with step equal to 0.5)

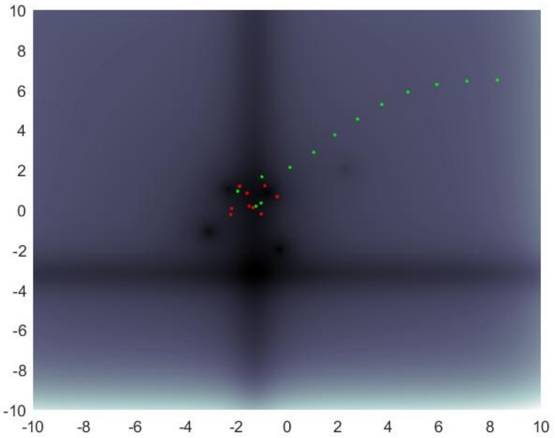


(Convergence curve of a Grid Search with step equal to 1.5)

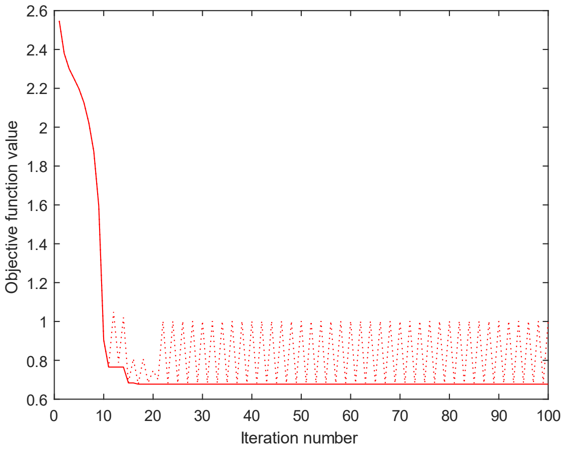
Ex.2: Gradient algorithm

Second algorithm that will be investigated in my report is gradient algorithm. Procedure that describes this algorithm is as follows: compute the gradient (a differential operator applied to a three-dimensional vector-valued function to yield a vector whose three components are the partial derivatives of the function with respect to its three variables), the first order derivative of the function at that point and make a step in the direction opposite to the gradient, opposite direction of slope increase from the current at that point.

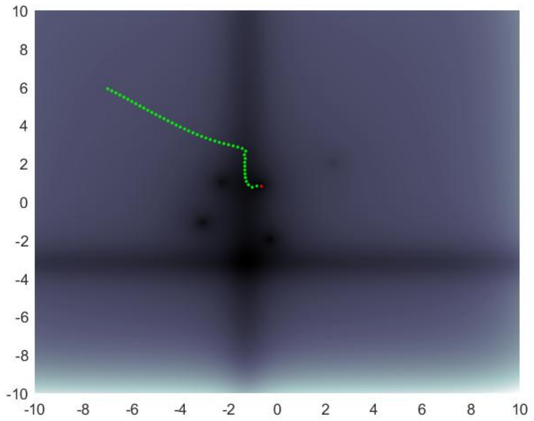
To obtain a gradient algorithm I need to change the code so that it chooses a starting point randomly, and then takes the next step in the direction of steepest gradient descent.Implementation of this code causes the calculation of gradient of the objective function around a testing point. After that it picks point for next loop passing by multiplying previous coordinates by a normalized gradient and Step constant.The characteristic feature of this method is poor exploration that couses in many cases to fall into local minima instead of global minima. For a full representation of the complexity of gradient algorithm I will choose two arbitrary values of step and compare the obtained results.



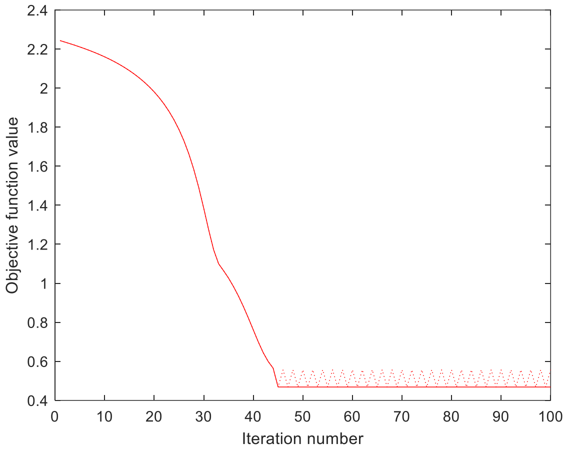
(Fig 3 - example of operation of a gradient algorithm with step equal to 1.2)



(Convergence curve of a gradient algorithm with step equal to 1.2)



(Fig 4 - example of operation of a gradient algorithm with step equal to 0.3)

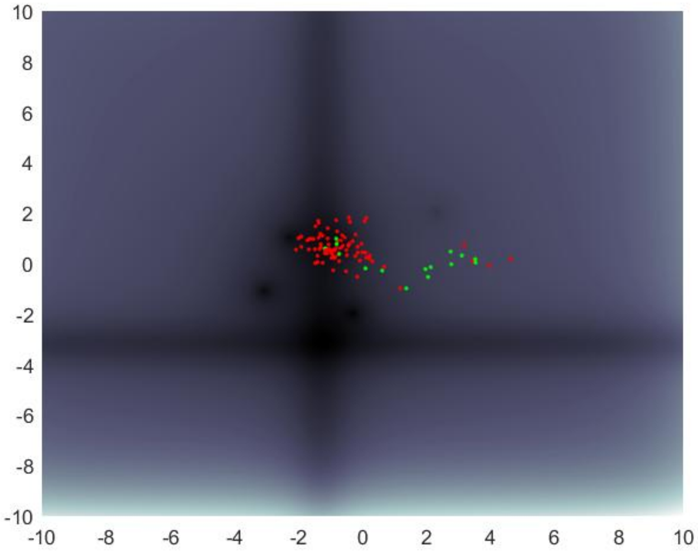


(Convergence curve of a gradient algorithm with step equal to 0.3)

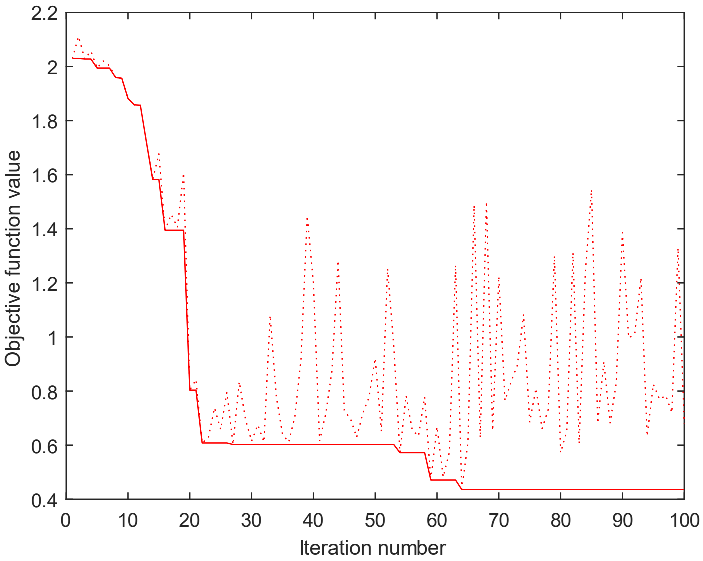
After carrying out many starts of the algorithm with different step parameters, we can see that the repeatability of the results is very low. The algorithm is susceptible to local minima. Contrary to the first algorithm, the change of parameters did not significantly affect the duration of the simulation. In addition to this method we have to remember, that if our objective function is flat, our solution will not improve.

Ex.3: 1+1 algorithm

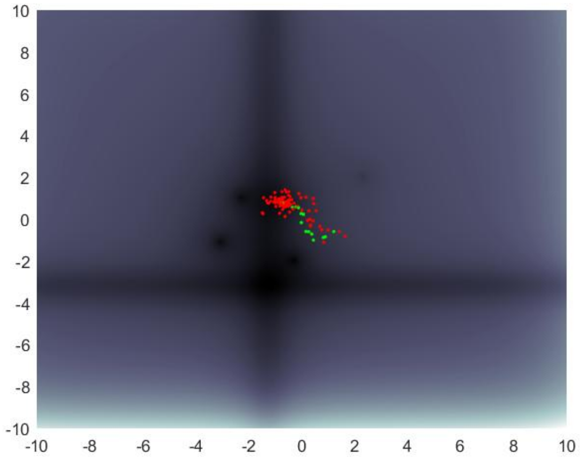
Third algorithm that will be investigated in my report is gradient algorithm. As in case with gradient algorithm our previous solution influences our future solutions. The difference with gradient algorithm the difference is that we don't calculate the gradient, we just compute the values from a certain space and save the best result. To obtain algorithm 1+1 I have to generate a random starting point. Next step will be checking a value in this point, then generating a new point by adding to him a small random value. After that, the new point is being either preserved or discarded, depending on whether it yields a value lower than its predecessor.



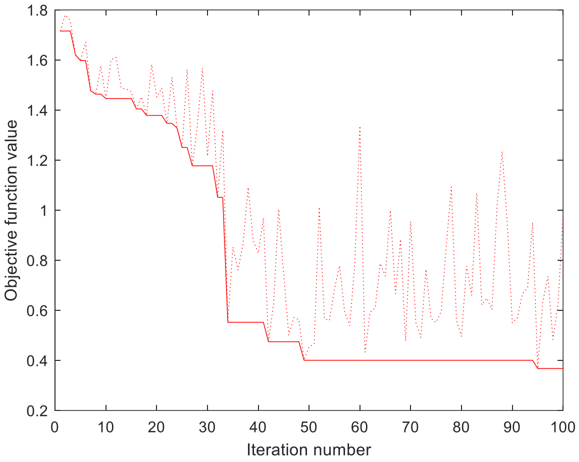
(Fig 5 - example of operation of a 1+1 algorithm with step equal to 0.5)



(Convergence curve of 1+1 algorithm with step equal to 0.5)



(Fig 6 - example of operation of a 1+1 algorithm with step equal to 0.3)

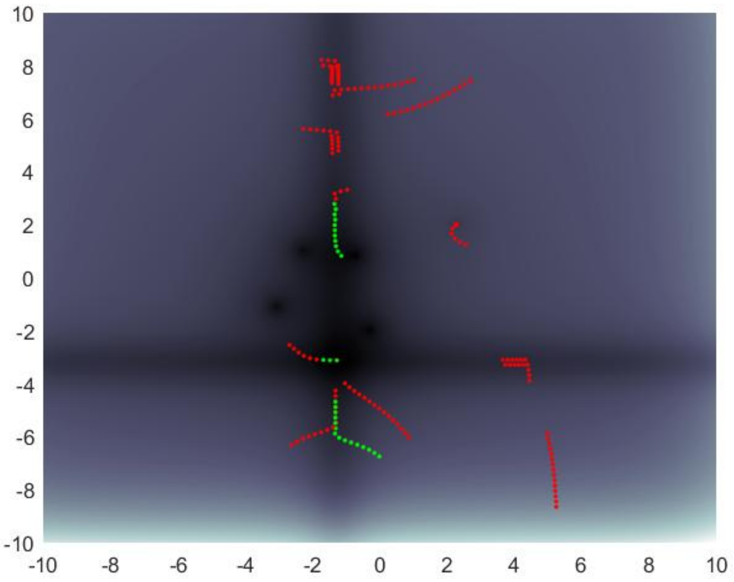


(Convergence curve of 1+1 algorithm with step equal to 0.3)

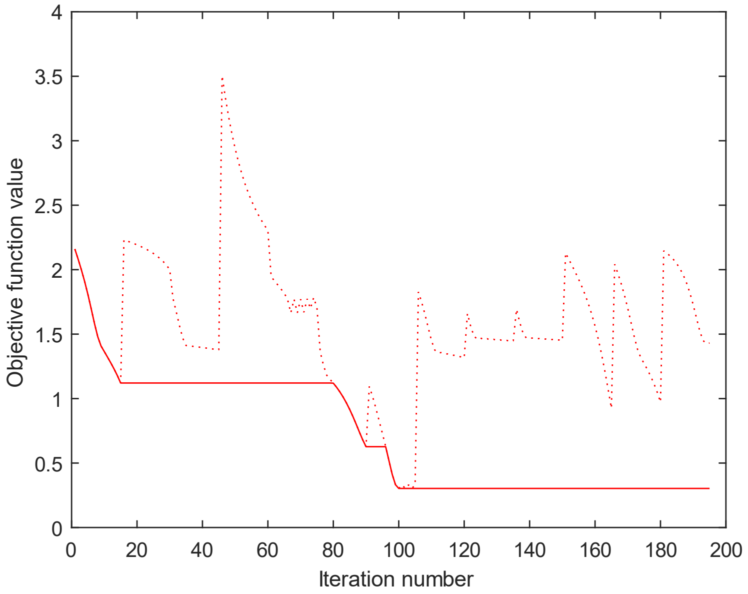
From this 2 examples, we can conclude that size of our step is cricual regarding to obtaining satisfying solution. If we increase our step value we may be able to search more parameter space, but improving value of minima may be challenging. Tradeoff between exploration of parameter space and expoitation is made by determination of step values. Our algorithm is really sensitive to the local minimum, despite a slightly better exploration than the gradient algorithm.

Ex.4: Multi-start gradient algorithm

In practical applications we often have many local minima, what causes that we are never sure if we have the best possible solution. A way of increasing the chance of hitting the global one is a multi-start algorithm. The best results we will obtain when we will base on gradient algorithm. By the use of the metaparameter “Starts” we will be able to obtain multi-start algorithm.



(Fig 7- example of operation of a multistart gradient)



(Convergence curve of a multistart gradient)

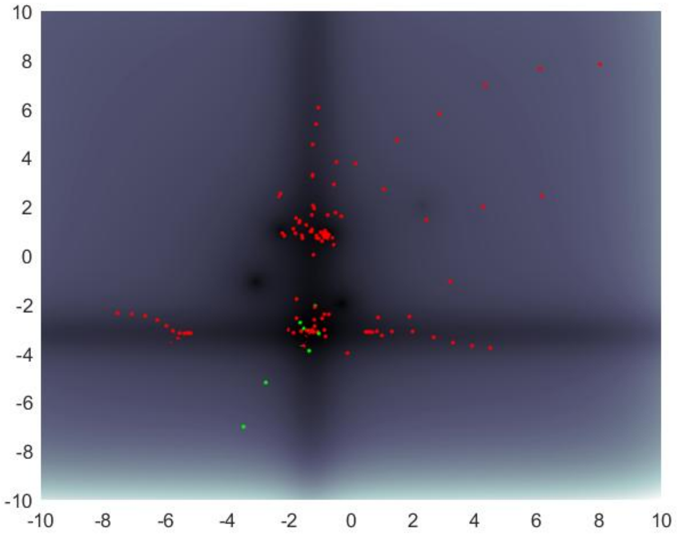
We still need to remember about characteristics of gradient algorithm. Even with many starts results may differ significantly, due to sensitivity to local minima. If our obejctive function is complex, results will be less reapatable.

Ex.5: Exploration and exploitation

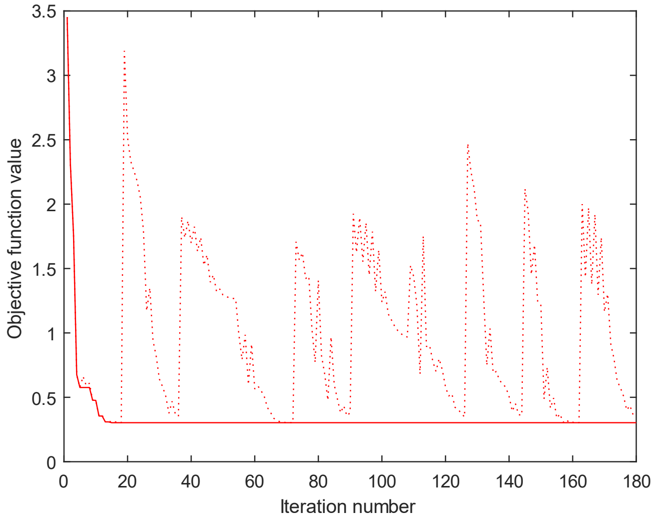
When we are testing the objective function the crucial thing to determine is initial step. It should be large enough to cover significant region of objective function, but later small enough to improve obtained result. We can obtain such phenomenon by using a sigmoid function, balancing the exploration/exploitation parameters.

How to compare methods?

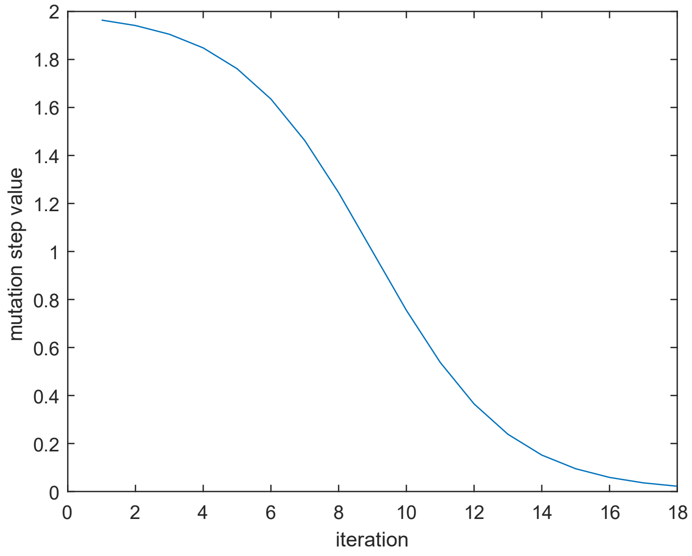
Comparison of methods have to be done properly to obtain sensible informations and conclusions. The most important thing is that comparions should not favour any of the solutions. In our exercises the most important determinant will be number of objective function checks. Note that number of checks depends on the dimensionality of the problem, number of iterations and the method. For example to execute comparison of 2D 1+1 algorithm and gradient algorithm number of iterations for the first one should be 3 times bigger.



(Fig 8- example of operation of a multistart gradient algorithm)



(Convergence curve of a multistart gradient)



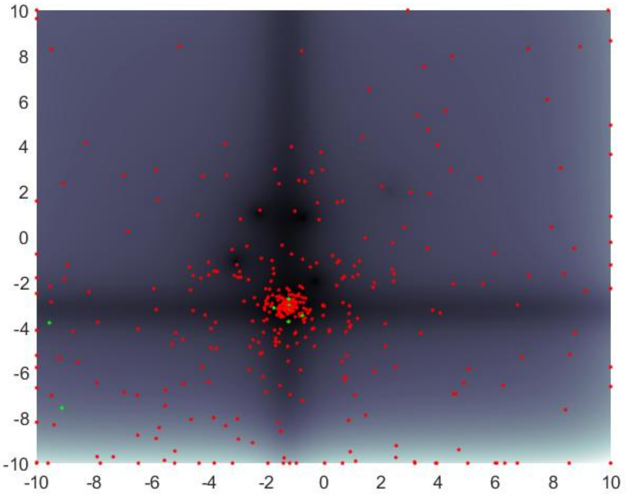
(Function that determines how value of step changes per iteration)

The problem with tradeoff of exploration and exploitation is made by changing the mutation step value per iteration. To determine if our function is well-optimized we have to interpret convergence curve and mutation step curve. Mutation step curve should be like sigmoid or the figure simillar to it.

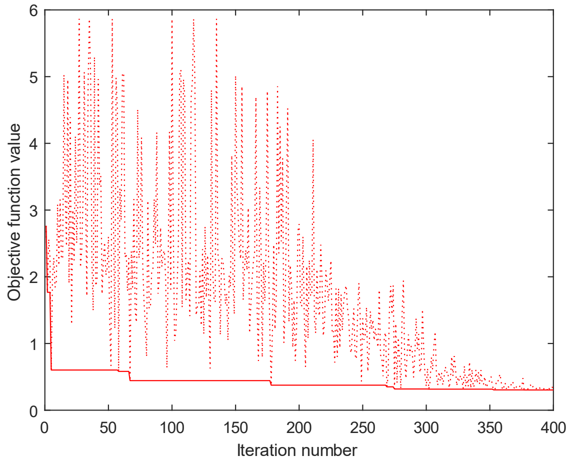
As we can see well-optimized multistart gradient algorithm gives us the most satysfying solutions. It requires more knowledge end experience, but the obtained results make up for it.

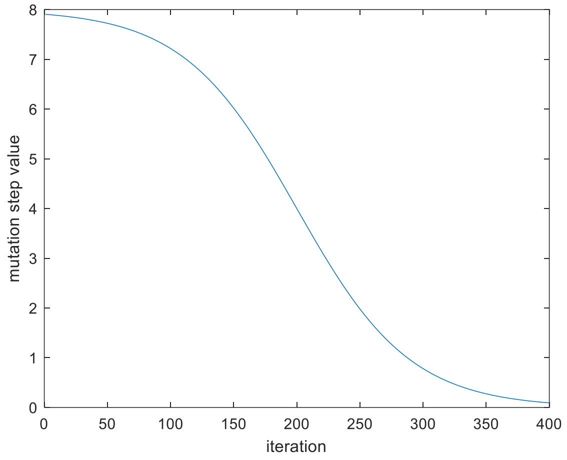
Ex.6: Implementation of 1+1 algorithm

As in the solution presented above, we have to take into account the step size change per iteration and optimize the algorithm, the only difference is change of algorithm to from gradient to 1+1.In this exercise is crucial to determine the parameters to obtain satisfying solution. We should remember to avoid dead iterations, rapid changes in mutation step values, all while still probing the whole parameter space.



(Fig 9- example of operation of a 1+1 algorithm)

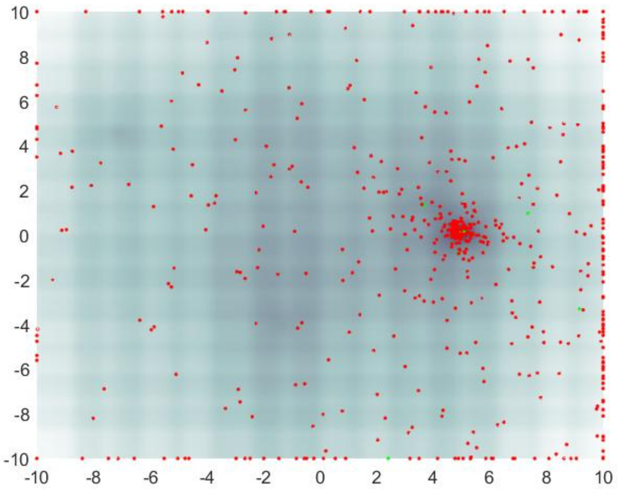




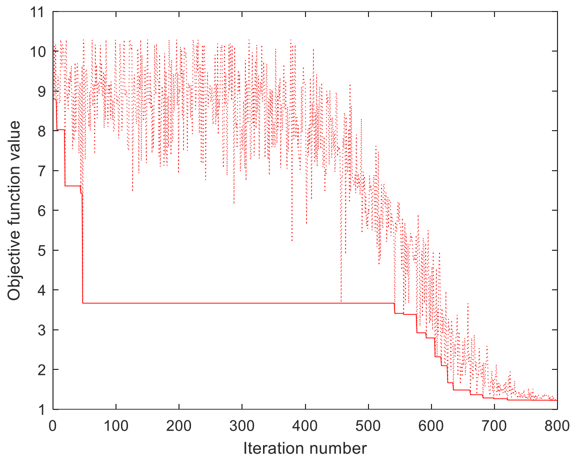
Our exploration is well done, because our samples cover significant procent of the space. Due to well-edjusted parameters P1 and P2 we obtained sigmoid function of mutation step value and we are able to improve expoilation in the end to finally detect our global minimum.

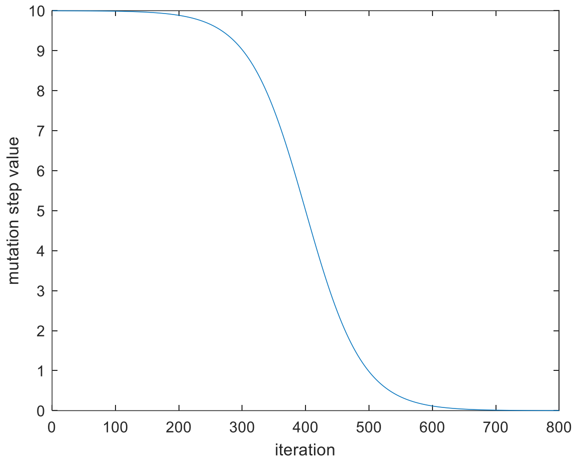
Ex. 7: 1+1 algorithm - manyminima

In this exercise, we implement the code as in previous exercise, but this time we change the type of the function to manyminima. This forces us to use more iterations to better explore the area of the function. As in previous case adjusting the parameters P1,P2 and mutation step value is crucial.



(Fig 10- example of operation of a 1+1 algorithm)





Ex 8: Repeatability check

To analyze the efficiency and consistency, we can use statistical tools. We use them, because our algorithm is non-deterministic. For this reason statistical approach to their assessment is required in order to check repeatability of a solution and likelihood of hitting a local minima.

Results:

0.30348	0.303	1.5362e-04	0.30302	0.3030	2.1617e-05	1.22025	1.2120	0.002
0.30348	mean	std	0.30299	mean	std	1.21179	mean	std
0.30303			0.30298			1.21179		
0.30303			0.30295			1.21179		
0.30303			0.30295			1.21179		
0.30303			0.30295			1.21179		
0.30303			0.30295			1.21143		
0.30303			0.30294			1.21143		
0.30297			0.30294			1.21143		
0.30297			0.30294			1.21143		
0.30297			0.30294			1.21143		
0.30297			0.30294			1.21143		
0.30297			0.30294			1.21143		
0.30297			0.30294			1.21143		
0.30297			0.30294			1.21143		
0.30297			0.30294			1.21143		
0.30297			0.30294			1.21143		
0.30297			0.30294			1.21143		

0.30297			0.30294			1.21143		
0.30297			0.30294			1.21143		

First column correspond to results from task 1.5, second column corresponds to results from task 1.6, and the third column corresponds to results from task 1.7.

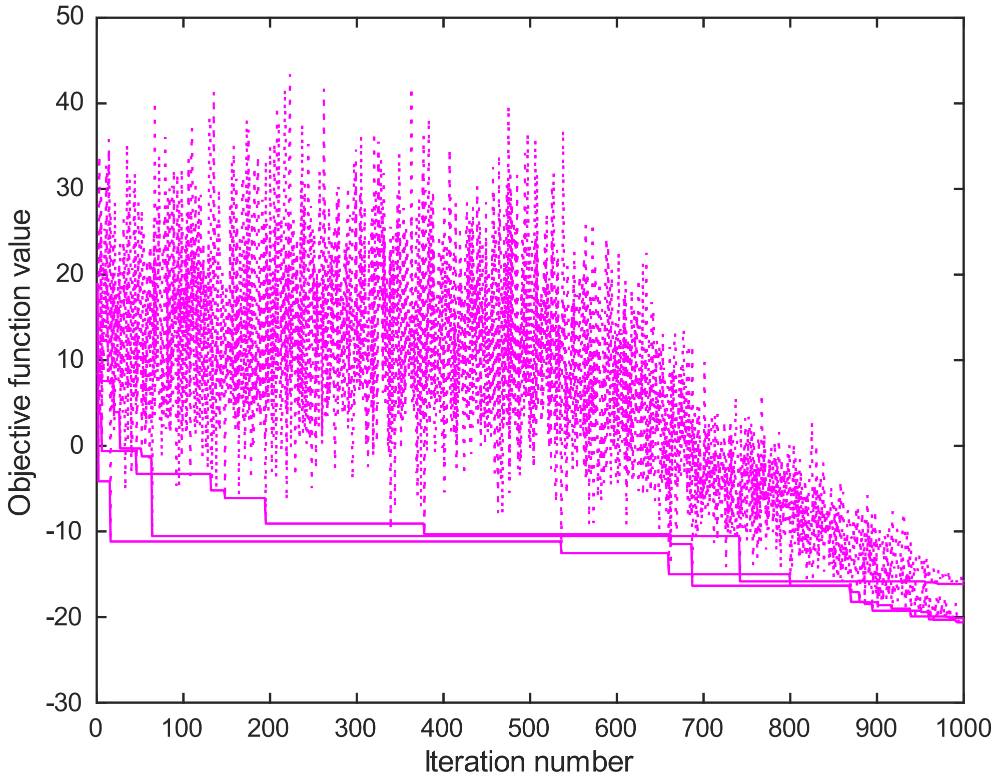
As we can notice, the values from the first and second parts are similar. Only difference is std value, which is smaller for many-minima 1+1 algorithm. Worth noticing is case with the 1+1 algorithm, obtained values are significantly larger and std value is smaller than in the first and second case. It is caused solely because a completely different function was optimized.

Ex. 9: Modify a 1+1 algorithm to work on 4-parameter (4D) functions.

In this exercise, we cannot use visualization, because we are working on a 4-dimensional problem. To obtain well-adjusted metaparameters, we have to rely only on convergence and step size curves.

Procedure with our result will be obtained:

1. Start with random values with initial step P1 and P2.
2. Max step value was set to be 1000, due to complexity of the problem.
3. Running the code three times each, small adjustments were made to assure the lowest possible values of minima and consistent results.
4. The most optimal parameters that were found:
   InitialStep = 6
   P1 = 1.7
   P2 = 45


Objective function value
Iteration number
50
40
30
20
10
0
-10
-20
-30
0
100
200
300
400
500
600
700
800
900
1000

As we can see our results are somehow consistent, repeatable, so we can conclude that our mataparameters are adjusted properly. Procedure used for this optimization was as follows: