TÉCNICO LISBOA

Aircraft Optimal Design

Aerospace Engineering

José Miguel Fonseca Santos

83704

1st Assignment

08/10/2019

Contents

1 Problem 1: Design Space, Objective Function and Constraints

The aim of this first exercise is to visualise graphically the contours of the Rosenbrock function in order to estimate its minimum. Te Rosenbrock function is a non-convex function that is useful to evaluate optimization algorithms. There is one global minimum inside a parabolic valley. The major problem when optimizing this function is that it's really easy to get stuck in the valley (not in the global minimum).

The Rosenbrock function is defined by:

$$f(x_1, x_2) = b(x_2 - x_1^2)^2 + (a - x_1)^2 \tag{1}$$

During this report the values for a and b will be 1 and 100, respectively.

1.1 Contours of the function

In this first part we are only interested in visualizing the contours of the function in the domain $-2 \leq x_1, x_2 \leq 3$. Therefore it was implemented with *meshgrid* and *contour* functions. The result is expressed in figure 1.

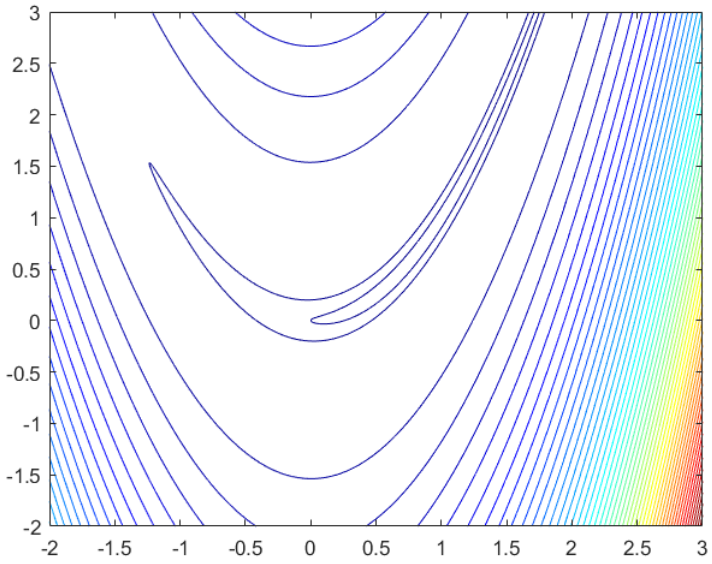

Figure 1: Plot of the contours of the Rosenbrock function

1.2 Minimum graphically estimation

According to the analyses of the first plot contour it's already possible to estimate the location of the minimum. It will be located inside the contour that is in the middle of all the others. This contour as the value of *f* equal to 1, so the next step is to get contours for levels in the interval $0 \leq f(x_1, x_2) \leq 1$ since we know the function will never be negative. The result is shown in the figure above where we can already estimate the location and value for the global minimum. The image shows the contour for a function value of 0.001 and it is clearly centered around the point of coordinates $(x_1, x_2) = (1, 1)$.

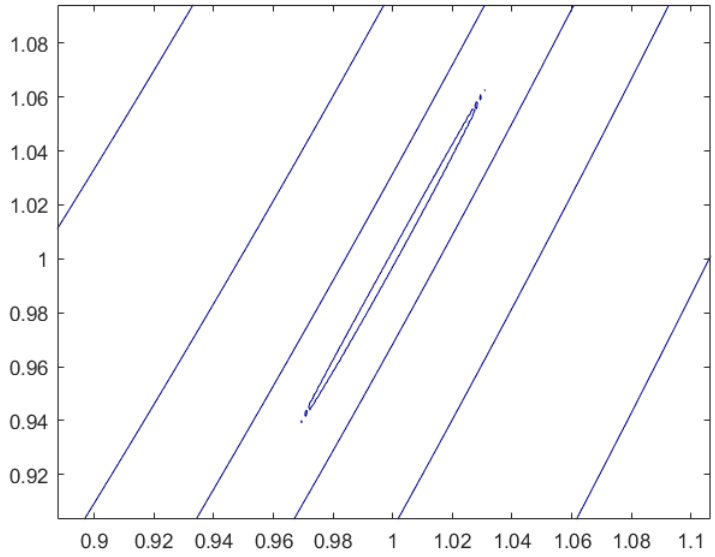

Figure 2: Graphic estimation of the minimum using MatLab function *contour*

1.3 Plot with inequality constrain and graphic estimation of new minimum

In this part the inequality given by $g(\mathbf{x} = x_1 + x_2 < 0$ was added to the plot. The not feasible area is colored in red.

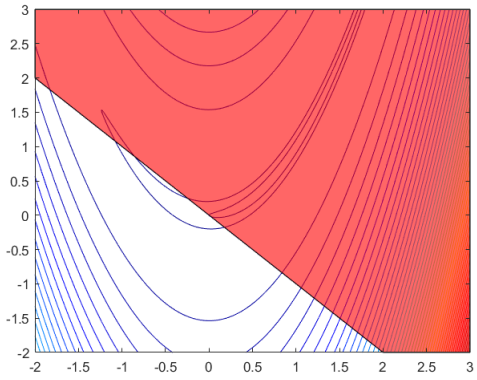

Figure 3: Inequality constrain

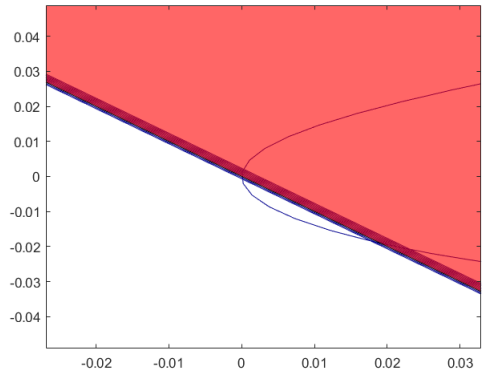

Figure 4: Contour and inequality constrain

In figure 5 it is possible to estimate the new minimum. The contour shown has the function value of 1. The graphic estimation makes possible to say that the minimum is located in the coordinates $(x_1, x_2) = (0, 0)$ having the value $f(\mathbf{x}) = 1$.

2 Minimization of the Rosenbrock function using *fminunc*

2.1 *fminunc* with default parameters

Using MatLab function *fminunc* and given for the input only the Rosenbrock function and the starting point $\mathbf{x}_0 = (2, 2)$ it was possible to obtain, as expected, a minimum of 4.63×10^{-12} for the point $(x_1, x_2) = (1, 1)$ wich confirmed the initial graphic estimation.

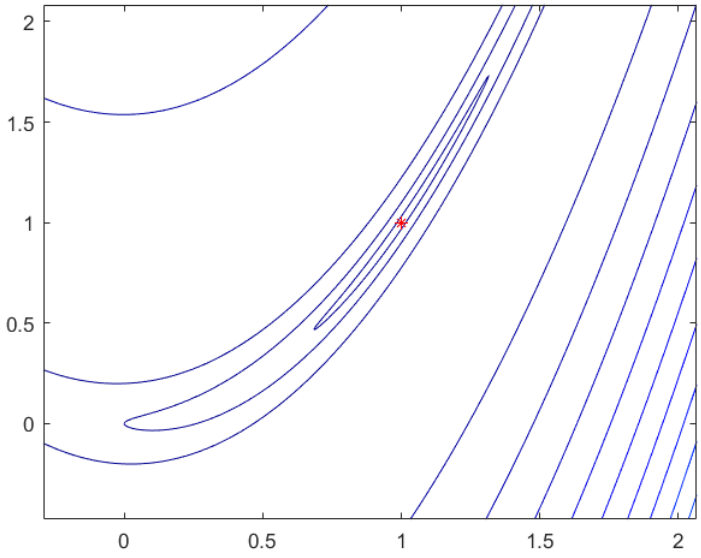

Figure 5: Plot of the minimum obtained with *fminunc*

The optimization stopped because the size of the gradient was less than the default value of the optimality tolerance (which has a default value of $1 \times^{-6}$). At the minimizing point, the gradient should be zero.

2.2 Results varying maximum number of iterations and absolute tolerance

Setting up different values for the maximum number of iterations, the results obtained are shown in table 1. It is possible to observe that only 22 iterations are needed in order to obtain the best result possible for the default optimality tolerance.

MaxIter	(x_1*, x_2*)	f(x_1*, x_2*)
1	$1.236, 2.147$	38.478
2	$1.440, 2.076$	0.194
10	$1.343, 1.787$	0.145
12	$1.236, 1.525$	0.057
20	$1.001, 1.003$	3.20×10^{-6}
22	$1.0001.000$	4.63×10^{-12}

Table 1: Results obtained for different number of maximum iterations

The relative tolerance, which correspond to the parameter *OptimalityTolerance*, was varied and the results are summarized in the next table.

Relative Tolerance	(x_1*, x_2*)	f(x_1*, x_2*)
1×10^{-1}	$1.440, 2.076$	0.194
1×10^{-2}	$1.440, 2.076$	0.194
1×10^{-3}	$1.440, 2.076$	0.194
1×10^{-4}	$1.007, 1.014$	5.4×10^{-5}
1×10^{-5}	$1.002, 1.004$	3.20×10^{-6}
1×10^{-6}	$1.000, 1.000$	4.63×10^{-12}

Table 2: Results obtained for different reltive tolerances

The results show that the solutionis the same while the *OptimalityTolerance* is greater than $1times10^{-3}$. Once the value gets to 1×10^{-4} the solution gets closer to the exact value.

3 Minimization of the Rosenbrock function in a constrain domain using *fmincon*

The objective of this section is to constrain the optimization problem with $g(\mathbf{x}) = x_1 + x_2 < 0$ and get an optimal result for the global minimum. After getting this result, it will be possible to compare it with the initial graphical analyses.

3.1 *fmincon* with default parameters

Using default parameters the solution obtained was $f(x_1^*, x_2^*) = 0.9903$ in $(x_1^*, x_2^*) = (0.0096, -0.0096)$.

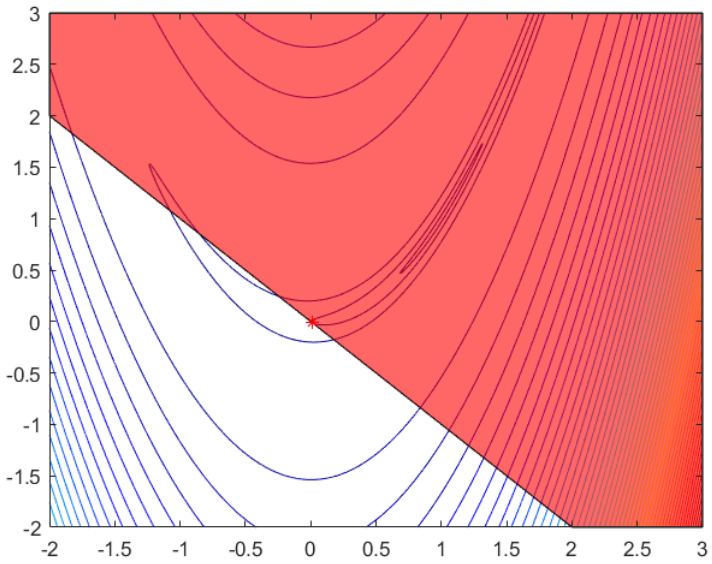

Figure 6: Verification using the contour and inequality from the first section

3.2 Results for different initial guesses

x_0	(x_1*, x_2*)	f(x_1*, x_2*)	Iterations number	Function evaluations
$(2, 2)$	$(0.0096, -0.0096)$	0.9903	17	58
$(2, -2)$	$(0.0096, -0.0096)$	0.9903	19	64
$(-2, -2)$	$(0.0096, 0.0096)$	0.9903	26	86
$(0, 0)$	$(0.0096, -0.0096)$	0.9903	9	42

It can be noticed that the number of iterations is related to how close the initial guess is to the solution. The minimum number of iterations is obtained when the initial guess is $(0, 0)$. COMENTAR

4 Heuristic Optimization using a Generic Algorithm

4.1 Solving using MatLab function *ga* with its default parameters

The heuristic method to be studied in this section is the genetic algorithm. In the first case, it will be used with default parameters, which means that the size of the population will be 50, maximum number of generations will be 200. The stopping criteria is defined with a lot of parameters within which are the maximum number of stall generations and the function tolerance. If the mean fitness value doesn't chance more than the function tolerance over the number of maximum stall generations, the algorithm stops.

(x_1*, x_2*)	(x_1*, x_2*)	Iterations number	Function evaluations
$(0.8113, 0.8668)$	4.3852	51	2600
$(0.3834, 0.1457)$	0.3804	73	3700
$(0.6524, 0.2766)$	2.399	62	3150
$(2.5793, 6.9087)$	9.0483	51	2600
$(2.2050, 4.8721)$	1.4623	95	4800
$(0.7707, 0.6718)$	0.6600	66	3350
$(0.1267, -0.0133)$	0.8486	53	2700

In order to have a better graphical perception of what the genetic algorithm is doing and able to reach with default parameters, two plots were made. This plots are shown in the next figure for the last generation of result obtained last in the previous table.

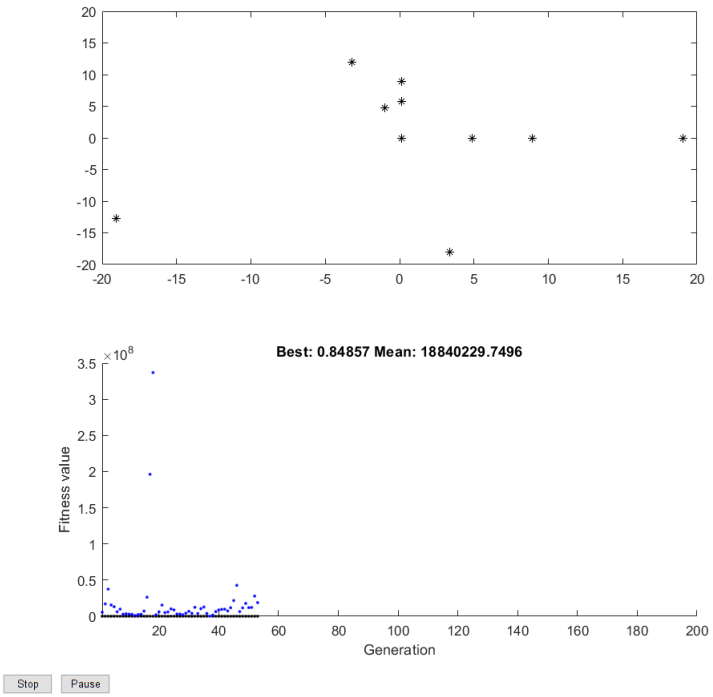

Figure 7: First plot: Each individual in generation 53. Second plot: mean fitness value and best fitness value over generations

Opposite to what was expected, the individual were closer to each other in the first generation (not shown here because of lack of number of pages in the report). This happens cause over time, selection, mutation and cross-over are responsible for giving diversity to the elements. Since the stopping criteria reached was that the average change in the fitness value was less than the default function tolerance and with the information provided by the graphs above, we can conclude that the algorithm was not able to converge completely. At this point, it is necessary to change some of the genetic algorithm options in order to understand ways of making it converge for the Rosenbrock function.

Table 3: Number of individuals

20 individuals			**100 individuals**			**500 individuals**		
$\boldsymbol{x}$	$f(\boldsymbol{x})$	Gen.	$\boldsymbol{x}$	$f(\boldsymbol{x})$	Gen.	$\boldsymbol{x}$	$f(\boldsymbol{x})$	Gen.
$0.231, 0.1468$	1.56	99	$2.665, 7.110$	2.77	51	$1.003, 1.005$	1.5×10^{-4}	184
$-1.638, 2.687$	6.96	58	$0.467, 0.254$	0.414	107	$1.4184, 2.0176$	0.18	127
$0.558, 0.402$	1.03	77	$0.927, 0.832$	0.0806	89	$0.910, 0.822$	0.01	120
$2.549, 4.678$	2.45	87	$1.264, 1.600$	0.07	200	$1.090, 1.194$	0.01	81

The results show that for a larger number of individuals the output keeps getting closer to the exact solution. On the downside, the number of generations and function evaluation increases a lot. For a population of 500 individual, the number of function evaluation is more less 100.000, increasing the memory required and the time taken to perform the optimization.

In order to determine the influence of the value for the function tolerance a series of simulations were made with different values for it. The solution obtained was similar when decreasing the tolerance. This says that the default value of 1×10^{-6} is already really low and that it doesn't make much difference to decrease. When increasing this value the solution starts getting away from the exact value. The maximum number of iterations also doesn't affect the results since the default value for the maximum stall generations is 50 and it is always the reason why the method stops. Bearing in mind what was just said, in order to have a better understanding on the number of iterations effect, the maximum stall generations was set to infinite.

Table 4: Number of generations

200 generations			**350 generations**			**500 generations**		
$\boldsymbol{x}$	$f(\boldsymbol{x})$	fcount	$\boldsymbol{x}$	$f(\boldsymbol{x})$	fcount	$\boldsymbol{x}$	$f(\boldsymbol{x})$	fcount
$1.767, 3.117$	0.590	10050	$1.269, 1.624$	0.086	17550	$0.987, 0.974$	1.8×10^{-4}	25050
$0.680, 0.465$	0.103	10050	$0.967, 0.933$	1.7×10^{-3}	17550	$1.056, 1.112$	3.4×10^{-3}	25050
$1.447, 2.090$	0.2008	10050	$1.143, 1.300$	0.025	17550	$1.078, 1.654$	6.8×10^{-4}	25050
$1.791, 3.2120$	0.628	10050	$1.080, 1.162$	6.9×10^{-3}	89	$0.848, 0.724$	0.0253	25050

It is possible to conclude that the solution gets closer to the exact minimum while increasing the number of generations. On the downside, the number of function evaluations and time taken increases. For 500 generations and a population of 50 individuals, it's necessary to evaluate the fitness function

25050 times.

To conclude, an extreme simulation was made in which the number of individuals and generations was increased a lot. In the next figure is shown the results obtained. The solution was $(x_1, x_2) = (0.9999, 0.9997)$ and $f(\boldsymbol{x}) = 9.1 \times 10^{-7}$. The fitness function was evaluated over 500.000 times. It can be visualized that all the individuals converge to the exact solution.

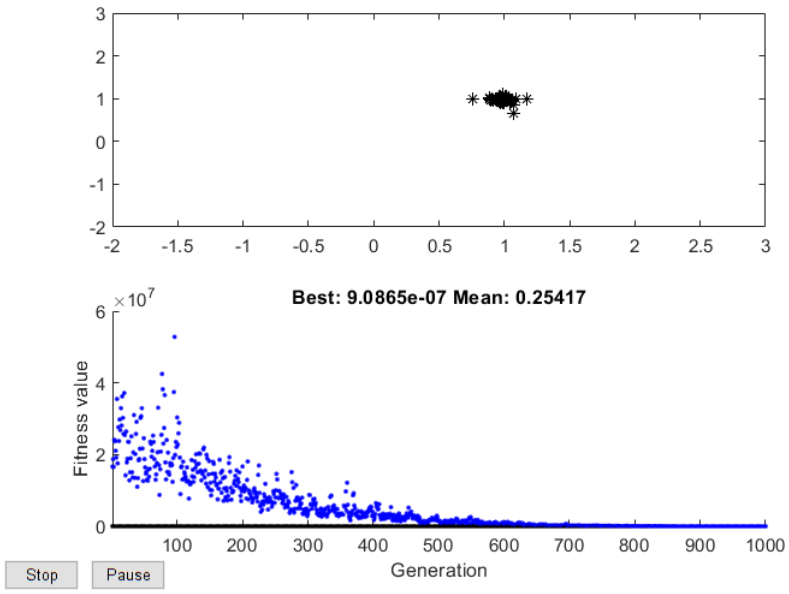

Figure 8: Solution for an optimization with 500 initial individuals and 1000 generations

It is now possible to conclude that the genetic algorithm applied to the *Rosenbrock* function is not the best option to minimize it. It requires a lot of memory, time and iterations to obtain a really close to the exact solution. In order to understand if the genetic algorithm takes a lot of time with other functions as well it was experimented with the *Rastrigin* function and the results were a lot more positive. The solution converged a lot quicker and with few generations and individuals it is possible to get a solution really close to the exact minimum. In the *Rosenbrock* function once the solution converges to the valley, it's difficult to to get to the minimum.