Homework 1

Ilia Kamyshev

February 8, 2021

Contents

1 Cylindrical can

The surface of a cylinder $S(r, h)$ can be considered as a function of two dimensions $S(r, h) = 2\pi(rh + r^2)$ and the volume is following function $V(r, h) = \pi r^2 h$. Assume we have some optimal r_{opt}, h_{opt} which correspond to a minimum of the function $S(r, h)$. Note, that following also holds: $V(r_{opt}, h_{opt}) = V$, this is our constraint as well as $r \geq 0$. Thus, we can write:

$$S(r = r_{opt}, h = h_{opt}) = 2\pi r_{opt}^2 + 2\pi r_{opt} h_{opt}$$

$$S(r = r_{opt}) = 2\pi r_{opt}^2 + \frac{2V}{r_{opt}}$$

But we do not know r_{opt}, since we may rewrite the last equation in terms of r as a variable:

$$S(r) = 2\pi r^2 + \frac{2V}{r}, \ r > 0$$

To find $r = r_{opt}$ we first have to optimize our objective function $S(r)$:

$$\frac{d}{dr}S(r) = 4\pi r - \frac{2V}{r^2} = 0$$

From where:

$$r_{opt} = (\frac{V}{2\pi})^{\frac{1}{3}}$$

And the h:

$$h_{opt} = \frac{V}{\pi r_{opt}^2}$$

Now we have to be sure that r_{opt}, h_{opt} correspond to the minimum of a function. For local minimum the sufficient condition is $\frac{d^2}{dr^2}S(r) > 0$. Indeed, $\frac{d^2}{dr^2}S(r) = 4\pi + \frac{4V}{r^3} > 0$ for any $r > 0$.

Answer: $r_{opt} = (\frac{V}{2\pi})^{\frac{1}{3}}$ and $h_{opt} = \frac{V}{\pi r_{opt}^2}$.

2 Optimally conditions

$$f(x_1, x_2) = \frac{3}{2}(x_1^2 + x_2^2) + (1+a)x_1x_2 - (x_1 + x_2) + b$$

Neccessary condition of optimallity is $\nabla f = 0$:

$$\begin{cases} \frac{d}{dx_1}f(x_1, x_2) = 3x_1 + (1+a)x_1 - 1 = 0 \\ \frac{d}{dx_2}f(x_1, x_2) = 3x_2 + (1+a)x_2 - 1 = 0 \end{cases}$$

From where

$$x_1 = \frac{1}{4+a}, \ a \neq -4$$

$$x_2 = \frac{1}{4+a}, \ a \neq -4$$

Sufficient condition of optimallity and local minima is $\nabla^2 f \succ 0$. The Hessian ∇^2 of $f(x_1, x_2)$ is:

$$\begin{bmatrix} \frac{d^2}{dx_1^2}f(x_1, x_2) & \frac{d^2}{dx_1 dx_2}f(x_1, x_2) \\ \frac{d^2}{dx_2 dx_1}f(x_1, x_2) & \frac{d^2}{dx_2^2}f(x_1, x_2) \end{bmatrix}$$

$$\begin{bmatrix} 3 + (1+a) & 0 \\ 0 & 3 + (1+a) \end{bmatrix}$$

Eigenvalues $\{\lambda_i : 1 \leq i \leq 2\}$ can be found from $|\nabla^2 f - \lambda \mathbf{I}| = 0$:

$$\begin{bmatrix} 3 + (1+a) - \lambda & 0 \\ 0 & 3 + (1+a) - \lambda \end{bmatrix}$$

$$(4 + a - \lambda)^2 = 0$$

$$\lambda_{1,2} = 4 + a$$

So for $\nabla^2 f \succ 0$ we will have:

$$\lambda_{1,2} > 0$$

$$4 + a > 0 \implies a > -4$$

For $a > -4$, $b \in \mathbb{R}$ the solution is optimal and locally minimal. This is unique since there is only one solution of $\nabla f = 0$ (only one critical point).

Answer: $a > -4$ and $b \in \mathbb{R}$.

3 Nelder-Mead method

Given:

$$f(x, y) = \sin(y)^{(1-\cos(x))^2} + \cos(x)^{(1-\sin(y))^2} + (x - y)^2$$

Subjected to $(x + 5)^2 + (y + 5)^2 < 25$. The plot of this function with domain boundaries is on Figure 1.

Note. The implementation of the algorithm was taken from this paper [Lagarias, J.C., Reeds, J.A., Wright, M.H., Wright, P.]. The methodology on the constructing of initial simplex with given size $l > 0$ was taken from this book [Michael Baudin]:
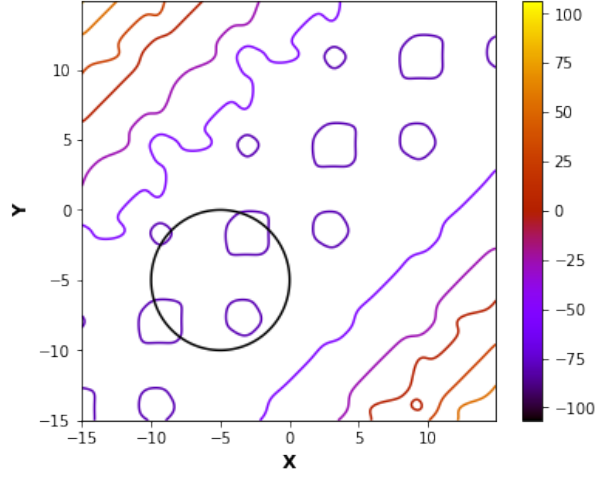
2

Figure 1: Mishar's Bird function contour plot. The circle is a domain constraint

$$p = \frac{1}{n\sqrt{2}}(n - 1 + \sqrt{n+1})$$

$$q = \frac{1}{n\sqrt{2}}(\sqrt{n+1} - 1)$$

where n is a dimensionallity of a problem ($n = 2$ in our case). Simplex is $\{v_i\}_{i=1,n+1}$. The first vertice $v_1 = x_0$, where x_0 is an initial guess. So the other vertices are defined by:

$$(v_i)_j = \begin{cases} (x_0)_j + lp \text{ if } j = i - 1 \\ (x_0)_j + lq \text{ if } j \neq i - 1 \end{cases}$$

Default parameters of an algorithm are: reflection $\rho = 1$, expansion $\chi = 2$, contraction $\gamma = 0.5$, shrinkage $\sigma = 0.5$.

3.1 The demonstration of the algorithm
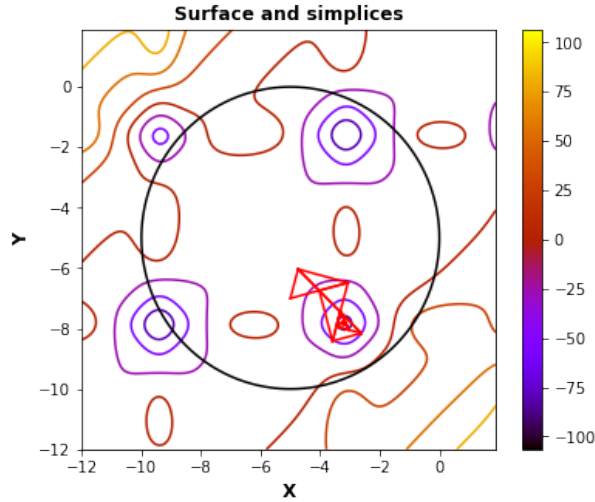

Figure 2: Nelder-Mead algorithm demonstration for Mishar's Bird function. $x0 = \{-5, -7\}$, $x^* = \{-3.18, -7.82\}$, $f(x^*) = -87.3108$, $l = 1$, tolerance $\epsilon = 10^{-4}$, Oracle calls $= 146$.
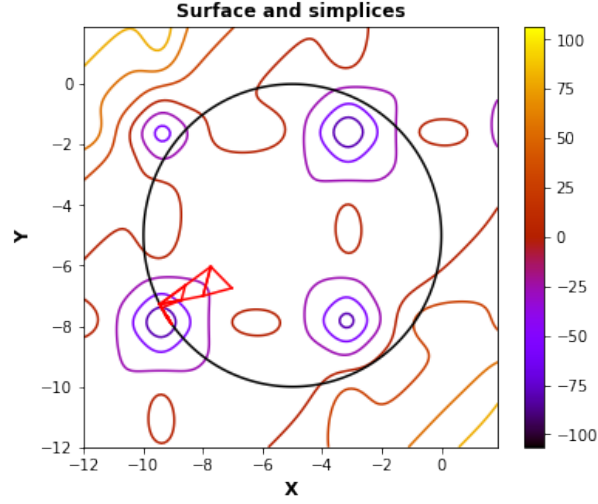
3

3.2 Different starting points

Figure 3: Nelder-Mead algorithm demonstration for Mishar's Bird function. $x0 = \{-8, -7\}$, $x^* = \{-9.19, -7.73\}$, $f(x^*) = -97.8942$, $l = 1$, tolerance $\epsilon = 10^{-4}$, Oracle calls $= 108$.
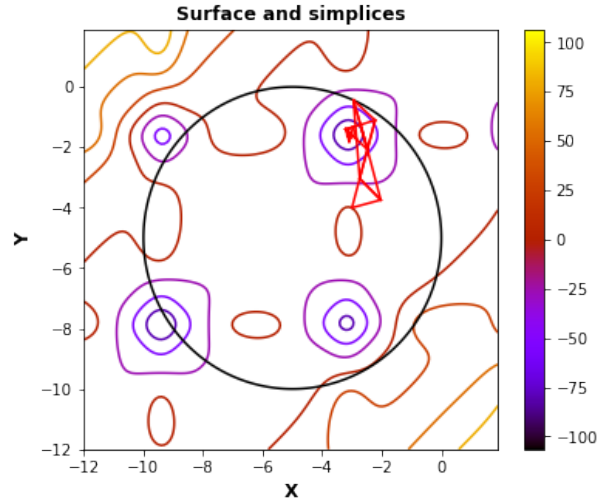

Figure 4: Nelder-Mead algorithm demonstration for Mishar's Bird function. $x0 = \{-3, -4\}$, $x^* = \{-3.13, -1.58\}$, $f(x^*) = -106.7645$, $l = 1$, tolerance $\epsilon = 10^{-4}$, Oracle calls $= 125$.

Note. These figures show that the simplices are not going over the boundary, but at some iterations they are 'sliding' along the boundary in the direction of function decrease. However, the first figure shows that the actual local minima is on the other side of the boundary, and here the final simplex touches the boundary in the point of minimal value of the function.
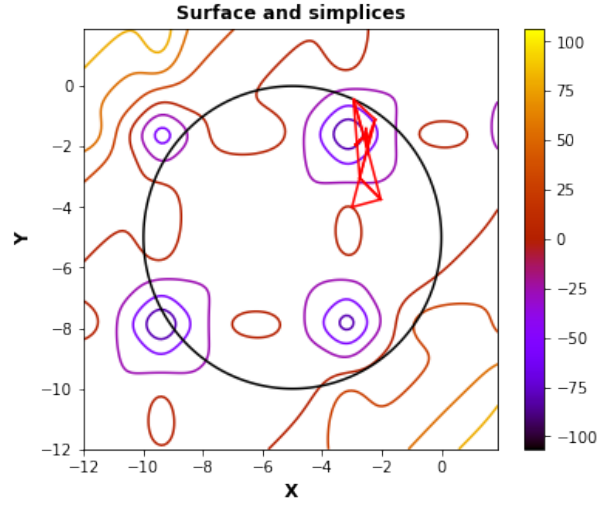
3.3 Different parameters

Figure 5: Nelder-Mead algorithm demonstration for Mishar's Bird function. $x0 = \{-3, -4\}$, $x^* = \{-2.85, -1.91\}$, $f(x^*) = -84.5684$, $l = 1$, tolerance $\epsilon = 10^{-4}$, Oracle calls $= 145$, contraction $\gamma = 0.1$.
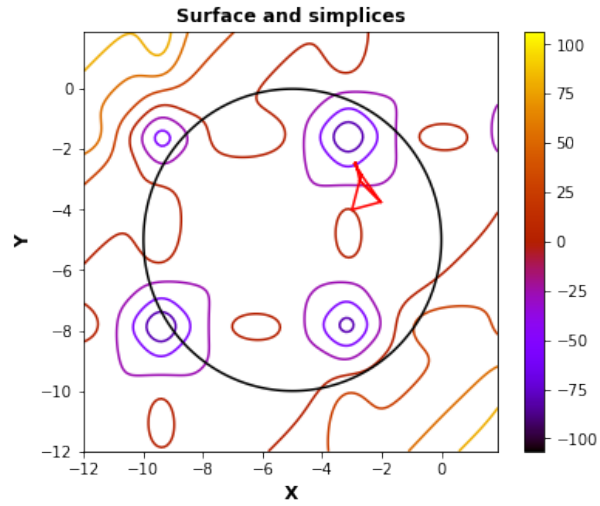

Figure 6: Nelder-Mead algorithm demonstration for Mishar's Bird function. $x0 = \{-3, -4\}$, $x^* = \{-2.89, -2.48\}$, $f(x^*) = -42.8311$, $l = 1$, tolerance $\epsilon = 10^{-4}$, Oracle calls $= 128$, reflection $\rho = 0.1$.
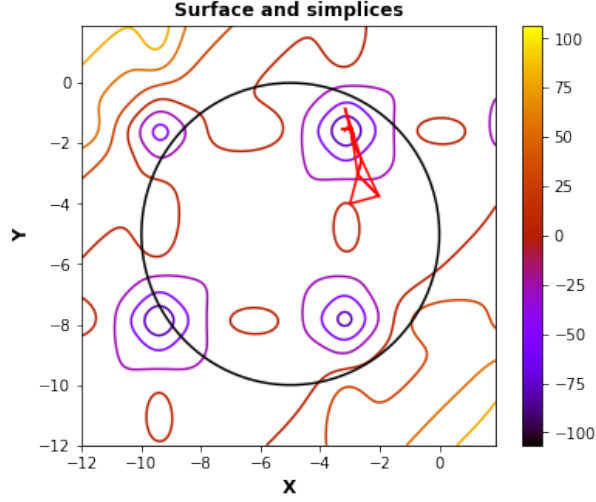
Figure 7: Nelder-Mead algorithm demonstration for Mishar's Bird function. $x0 = \{-3, -4\}$, $x^* = \{-3.13, -1.53\}$, $f(x^*) = -106.4303$, $l = 1$, tolerance $\epsilon = 10^{-4}$, Oracle calls $= 270$, reflection $\rho = 0.5$ and expansion $\chi = 10$.
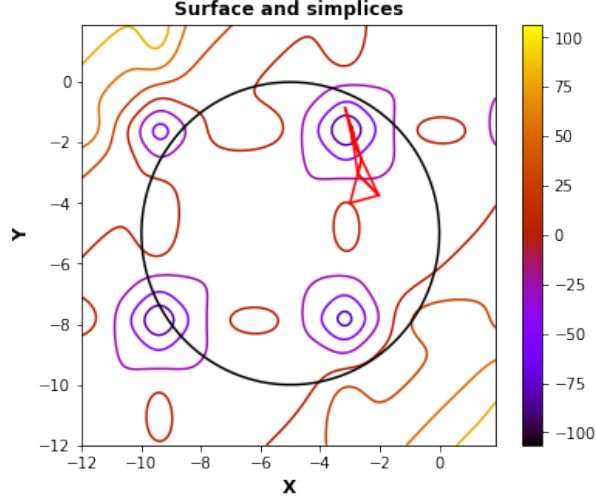

Figure 8: Nelder-Mead algorithm demonstration for Mishar's Bird function. $x0 = \{-3, -4\}$, $x^* = \{-2.95, -1.53\}$, $f(x^*) = -102.3585$, $l = 1$, tolerance $\epsilon = 10^{-4}$, Oracle calls $= 114$, reflection $\rho = 0.5$, expansion $\chi = 10$, contraction $\gamma = 0.1$ and shrinkage $\sigma = 0.9$.

4 Coordinate descend

$x_0 = \{-3, -4\}$ was chosen as initial guess. The 'update rule' at iteration k is as follows:

$$x(k+1) = x(k) - \gamma_k s(k)$$

where $s(k)$:

$$\frac{1}{2\alpha_k}(f(x(k) + \alpha_k e_j) - f(x(k) - \alpha_k e_j))e_j$$

where e_j is a direction over jth component. And j is cyclic: $j = k \mod n$, for $n = 2$.

Figure 9: Coordinate descend algorithm demonstration for Mishar's Bird function. $x0 = \{-3, -4\}$, $x^* = \{-3.13, -1.58\}$, $f(x^*) = -106.7645$, $\gamma = 0.005$, $\alpha = 10^{-10}$, tolerance $\epsilon = 10^{-4}$, Oracle calls = 47
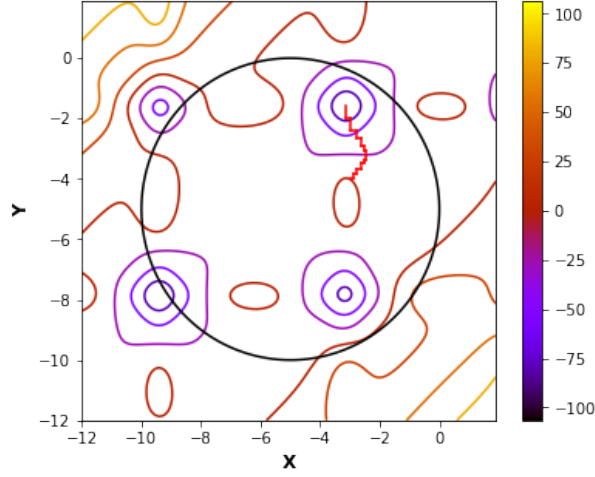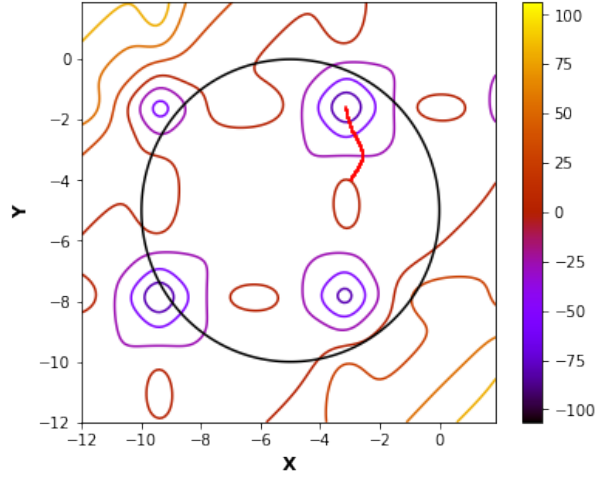

Figure 10: Coordinate descend algorithm demonstration for Mishar's Bird function. $x0 = \{-3, -4\}$, $x^* = \{-3.13, -1.58\}$, $f(x^*) = -106.7645$, $\gamma = 0.001$, $\alpha = 10^{-10}$, tolerance $\epsilon = 10^{-4}$, Oracle calls = 169

5 Conclusion

Python notebook for last two problems is available on github: https://github.com/arx7ti/optimization-course/Homework1.ipynb

5.1 Comment on the termination criterias

For Nelder-Mead method the absolute difference between best and worst vertices was used as stopping criteria. For given tolerance $\epsilon = 10^{-4}$ this difference should be less than ϵ, this also implies that we can use objective function value at the best vertex as the value of minima.

For coordinate descend the $\max-$norm of a gradient $s(k)$ was used. This implies, that in point near to the local minima the gradient over $j-$th component $(s(k))_j$ should reach 0.

5.2 Comparison of two methods

For the common starting point $x_0 = \{-3, -4\}$ in some cases Coordinate descend method requires less oracle calls than Nelder-Mead method (47 vs 125). But it depends on the parameters given. For example, if we will decrease γ for coordinate descend, this will lead to 169 oracle calls (but the path will be smoother).

Generally, coordinate descend may converge faster if the initial guess is not on a flat surface: requires from 1 to 2 Oracle calls per 1 algorithm run (depending on the approximation scheme of a gradient), where Nelder-Mead method requires at least 3 Oracle calls for 1 algorithm run (evaluate function on 3 vertices). In case is x_0 is on flat piece of surface, Nelder-Mead method with larger initial size l of a simplex can be a solution: it can cover another level set and therefore, simplex will be moved from the flat neighborhood and will be morphed accrodingly to the function decrease. For coordinate descend we will need larger γ_k in the beginning and decrease it accrodingly to the function decrease, but for constant and relatively large γ_k it may skip all the optimal points. Another advantage of Nelder-Mead method is that derivatives of an objective function may not be known, whereas for coordinate descend we should compute gradient components.