

Numerical Linear Algebra Homework Project 5: Constrained Optimization

Catalano Giuseppe, Cerrato Nunzia

Exercise 1

We want to minimize the function

$$f_a(\mathbf{x}) = (x_1 - 4)^2 + x_2^2 \quad (1)$$

subject to the constraints

$$x_1 + x_2 \leq 2, \quad x_1 \geq 0, \quad x_2 \geq 0, \quad (2)$$

which identify the set $\mathcal{C}_a = \{\mathbf{x} \in \mathbb{R}^2 : x_1 + x_2 \leq 2, x_1 \geq 0, x_2 \geq 0\}$. In other words, we want to find

$$f_a(\mathbf{x}^*) = \min_{\mathbf{x} \in \mathcal{C}_a} f_a(\mathbf{x}). \quad (3)$$

Since the function $f_a(\mathbf{x})$, reported in Eq. (1), is convex and continuous and the constraint set \mathcal{C}_a is convex, there exists a unique minimum. In particular, it can be seen that, since there are no mixed terms in $x_1 x_2$ in the expression of the function, its minimum can be found by minimizing $(x_1 - 4)^2$ and x_2^2 separately. x_2^2 is minimized when $x_2 = 0$, which is compatible with the constraint set, and $(x_1 - 4)^2$ is minimized when $x_1 = 4$, which is incompatible with the constraint set. Since it has to be $x_1 + x_2 \leq 2$, the optimal choice of x_1 is $x_1 = 2$. Therefore we have:

$$\mathbf{x}^* = (2, 0), \quad f_a(\mathbf{x}^*) = 4. \quad (4)$$

We would like to find the minimum value both analytically, by solving the KKT system, and numerically, by using the interior point method.

The Lagrangian of the problem is the following:

$$\mathcal{L}_a(\mathbf{x}, \boldsymbol{\lambda}) = f_a(\mathbf{x}) - \boldsymbol{\lambda}^T \mathbf{c}_a(\mathbf{x}), \quad \text{with } \mathbf{c}_a(\mathbf{x}) = \begin{bmatrix} 2 - x_1 - x_2 \\ x_1 \\ x_2 \end{bmatrix}, \quad (5)$$

where the constraints of the problem are satisfied when $c_i(\mathbf{x}) \geq 0 \ \forall i = 1, 2, 3$. From this definition, it is possible to write the KKT conditions, namely:

$$\left\{ \begin{array}{ll} \text{stationarity} & (2(x_1 - 4) + \lambda_1 - \lambda_2, 2x_2 + \lambda_1 - \lambda_3) = (0, 0) \\ \text{primal feasibility} & (2 - x_1 - x_2, x_1, x_2) \geq (0, 0, 0) \\ \text{dual feasibility} & (\lambda_1, \lambda_2, \lambda_3) \geq (0, 0, 0) \\ \text{complementarity} & (\lambda_1(2 - x_1 - x_2), \lambda_2 x_1, \lambda_3 x_2) = (0, 0, 0) \end{array} \right. \quad (6)$$

where the inequalities must be intended element-wise.

The complementarity conditions can be satisfied in eight different cases, depending on the fact that each constraint can be active or not. We recall that a constraint $c_i(\mathbf{x})$ is active if and only if $c_i(\mathbf{x}^*) = 0$. In the following, we identify each case with an array of m entries, where m is the number of constraints,

and we assign to the i-th entry the label A or N depending on whether the i-th constraint is active or not. Associated to each case we report the possible solutions of the constrained minimization problem which are compatible with the corresponding complementarity condition.

	A/N constraints	Possible solutions \mathbf{x}^*
1	(A, N, N)	$\mathbf{x}^* = (x_1^*, -x_1^* + 2)$, with $x_1^* \in (0, 2)$
2	(A, A, A)	$\nexists \mathbf{x}^* \in \mathbb{R}^2$
3	(N, N, N)	$\mathbf{x}^* \in \mathcal{C}_a$
4	(N, A, A)	$\mathbf{x}^* = (0, 0)$
5	(A, A, N)	$\mathbf{x}^* = (0, 2)$
6	(A, N, A)	$\mathbf{x}^* = (2, 0)$
7	(N, A, N)	$\mathbf{x}^* = (0, x_2^*)$, with $x_2^* \in (0, 2)$
8	(N, N, A)	$\mathbf{x}^* = (x_1^*, 0)$ with $x_1^* \in (0, 2)$

Table 1: Table of the cases satisfying the complementarity condition. The i-th row of the table contains the array of the active/non-active states of the constraints (second column) and the possible solutions \mathbf{x}^* of the constrained minimization problem which are compatible with that case (third column).

We want to find the solutions to the KKT system (6) by investigating if the conditions on Lagrange multipliers imposed by the constraint states and the range of possible solutions \mathbf{x}^* reported in Tab. 1, which came from the complementarity condition, are compatible with the other equations of the system. The results that we find are reported below.

1. From the case (A, N, N) we have $\lambda_2^* = \lambda_3^* = 0$. This leads to $\lambda_1^* = 2$ and $\mathbf{x}^* = (3, -1)$, which is not compatible with the constraint set.
2. For the case (A, A, A) it can be seen from the second row of the Table 1 that $\nexists \mathbf{x}^* \in \mathbb{R}^2$ which represents a solution to the constrained minimization problem.
3. From the case (N, N, N) we have $\lambda_1^* = \lambda_2^* = \lambda_3^* = 0$. In this case one finds $\mathbf{x}^* = (4, 0)$, which is outside the feasible set.
4. From the case (N, A, A) we have $\lambda_1^* = 0$. From the complementarity condition we already have $\mathbf{x}^* = (0, 0)$. From the stationarity condition, we find $\lambda_3^* = 0$, and $\lambda_2^* = -8$, which is not allowed.
5. From the case (A, A, N) we have $\lambda_3^* = 0$. From the complementarity condition we already have $\mathbf{x}^* = (0, 2)$. By inserting this value in the first equation of the KKT system, we find $\lambda_1^* = -4$, $\lambda_2^* = -12$, which are both not allowed values.
6. From the case (A, N, A) we have $\lambda_2^* = 0$. In this case, one has $\mathbf{x}^* = (2, 0)$ from the complementarity condition and $\lambda_1^* = 4$, $\lambda_3^* = 4$, which are both acceptable solutions. Note that this is the solution that we have already found heuristically.
7. From the case (N, A, N) we have $\lambda_1^* = 0$, $\lambda_3^* = 0$. In this case, we find $\lambda_2 = -8$, which is not an acceptable solution. Note that one can also find $x_2^* = 0$, which is not compatible with the state of the constraint.
8. From the case (N, N, A) we have $\lambda_1^* = 0$, $\lambda_2^* = 0$. In this case, we find $\lambda_3 = 0$ and $\mathbf{x}^* = (4, 0)$, which is outside the feasible set.

As emerges from the study of the KKT system, $\mathbf{x}^* = (4, 0, 4)$ and the unique solution to the constrained minimization problem is $\mathbf{x}^* = (2, 0)$, where $f_a(\mathbf{x}^*) = 4$.

Exercise 2

We want to minimize the function

$$f_b(\mathbf{x}) = 2x_1 - x_2^2, \quad (7)$$

under the constraints

$$x_1^2 + x_2^2 \leq 1, \quad x_1 \geq 0, \quad x_2 \geq 0, \quad (8)$$

which identify the set $\mathcal{C}_b = \{\mathbf{x} \in \mathbb{R}^2 : x_1^2 + x_2^2 \leq 1, x_1 \geq 0, x_2 \geq 0\}$. In other words, we want to find

$$f_b(\mathbf{x}^*) = \min_{\mathbf{x} \in \mathcal{C}_b} f_b(\mathbf{x}). \quad (9)$$

In this case, the function $f_b(\mathbf{x}^*)$ is concave, therefore, in principle, nothing can be said about its minimum. However, by exploiting the constraints, we can obtain the following chain of inequalities

$$2x_1 - x_2^2 \geq -x_2^2 \geq -1, \quad (10)$$

where in the first inequality we use that $x_1 \geq 0$, which is saturated when $x_1 = 0$, while in the second one we use that $x_2^2 \leq 1 - x_1^2 \leq 1$, which is saturated when $x_2 = \pm 1$. Finally, knowing that x_2 must be positive, we can conclude that the unique solution of the problem in Eq. (9) is $\mathbf{x}^* = (0, 1)$ and $f_b(\mathbf{x}^*) = -1$.

We would like to obtain now the same minimum point by solving the KKT system analytically and, subsequently, by exploiting the interior point method numerically.

The Lagrangian associated with the problem is the following:

$$\mathcal{L}_b(\mathbf{x}, \boldsymbol{\lambda}) = f_b(\mathbf{x}) - \boldsymbol{\lambda}^T \mathbf{c}_b(\mathbf{x}), \quad \text{with } \mathbf{c}_b(\mathbf{x}) = \begin{bmatrix} 1 - x_1^2 - x_2^2 \\ x_1 \\ x_2 \end{bmatrix}, \quad (11)$$

where the constraints of the problem are satisfied when $c_i(\mathbf{x}) \geq 0 \ \forall i = 1, 2, 3$. The KKT system is:

$$\left\{ \begin{array}{ll} \text{stationarity} & (2 + 2\lambda_1 x_1 - \lambda_2, -2x_2 + 2\lambda_1 x_2 - \lambda_3) = (0, 0) \\ \text{primal feasibility} & (1 - x_1^2 - x_2^2, x_1, x_2) \geq (0, 0, 0) \\ \text{dual feasibility} & (\lambda_1, \lambda_2, \lambda_3) \geq (0, 0, 0) \\ \text{complementarity} & (\lambda_1(1 - x_1^2 - x_2^2), \lambda_2 x_1, \lambda_3 x_2) = (0, 0, 0) \end{array} \right. \quad (12)$$

where the inequalities must be intended element-wise.

Analogously to the previous case, we study the eight cases that can be derived from the complementarity condition. We report in the following table the possible solutions to the constrained minimization problem which are compatible with corresponding complementarity condition.

	A/N constraints	Possible solutions \mathbf{x}^*
1	(A, A, A)	$\nexists \mathbf{x}^* \in \mathbb{R}^2$
2	(A, A, N)	$\mathbf{x}^* = (0, 1)$
3	(A, N, A)	$\mathbf{x}^* = (1, 0)$
4	(A, N, N)	$x_2^* = \sqrt{(1 - x_1^*)^2}$, with $x_1^* \in (0, 1)$
5	(N, A, A)	$\mathbf{x}^* = (0, 0)$
6	(N, A, N)	$\mathbf{x}^* = (0, x_2^*)$, with $x_2^* \in (0, 1)$
7	(N, N, A)	$\mathbf{x}^* = (x_1^*, 0)$ with $x_1^* \in (0, 1)$
8	(N, N, N)	$\mathbf{x}^* \in \mathring{\mathcal{C}}_b$

Table 2: Table of the cases satisfying the complementarity condition. The i-th row of the table contains the array of the active/non-active states of the constraints (second column) and the possible solutions \mathbf{x}^* of the constrained minimization problem which are compatible with that case (third column).

1. For the case (A, A, A) it can be seen from the second row of the Table 2 that $\nexists \mathbf{x}^* \in \mathbb{R}^2$ which represents a solution to the constrained minimization problem.
2. For the case (A, A, N) we have $\lambda_3^* = 0$. From the complementarity condition we already know that $\mathbf{x}^* = (0, 1)$. By inserting this value in the first equation of the KKT system, we find $\lambda_1^* = 1$ and $\lambda_2^* = 2$, which are allowed values.
3. For the case (A, N, A) we have $\lambda_2^* = 0$. In this case, we have already found that $\mathbf{x}^* = (1, 0)$, which leads to $\lambda_1^* = -1$, which is not allowed, and $\lambda_3^* = 0$.
4. For the case (A, N, N) we have $\lambda_2^* = \lambda_3^* = 0$. From the first equation of the KKT system, we arrive at the condition $\lambda_1 x_1 = -1$, which cannot be satisfied since both λ_1 and x_1 must be positive.
5. For the case (N, A, A) we have $\lambda_1^* = 0$. Here we already have $\mathbf{x}^* = (0, 0)$, which leads to $\lambda_2^* = 2$ and $\lambda_3^* = 0$.
6. For the case (N, A, N) we have $\lambda_1^* = \lambda_3^* = 0$. In this case, we find $\lambda_2^* = 2$ and $\mathbf{x}^* = (0, 0)$.
7. For the case (N, N, A) we have $\lambda_1^* = \lambda_2^* = 0$. By inserting these values in the stationarity condition we arrive at the expression $2 = 0$, which is absurd, therefore no solutions can be found in this case.
8. For the case (N, N, N) we have $\lambda_1^* = \lambda_2^* = \lambda_3^* = 0$. This case is analogous to the previous one, since we arrive at the same expression $2 = 0$ when considering $\lambda_1^* = 0$, therefore no solutions can be found.

From the analysis of the KKT conditions, we can conclude that there exist two solutions of the KKT system, namely $\mathbf{x}_1^* = (0, 1)$ with $\boldsymbol{\lambda}_1^* = (1, 2, 0)$ and $\mathbf{x}_2^* = (0, 0)$ with $\boldsymbol{\lambda}_2^* = (0, 2, 0)$. By evaluating the function at these points we have $f_b(\mathbf{x}_1^*) = -1$ and $f_b(\mathbf{x}_2^*) = 0$, therefore we conclude that \mathbf{x}_1^* is the solution of the constrained minimization problem. However, since there are two solutions of the KKT system, we expect there could be problems when searching numerically the minimum of this function by using the interior point method.

Exercise 3

Consider the problem

$$\min f(\mathbf{x}) - \mu \mathbf{e}^T \log(\mathbf{z}) \quad \text{s.t.} \quad \mathbf{c}(\mathbf{x}) - \mathbf{z} = \mathbf{0}, \quad (13)$$

where $f : \mathbb{R}^n \rightarrow \mathbb{R}$, is the function one wants to find the minimum, $\mathbf{c} : \mathbb{R}^n \rightarrow \mathbb{R}^m$ is a vector defining the constraints, which are assumed to be $c_i(\mathbf{x}) \geq 0$, $\forall i = 1, \dots, m$, $\mathbf{z} \in \mathbb{R}^m$ is called slack variable and, finally, $\mathbf{e}^T = (1, \dots, 1) \in \mathbb{R}^m$ and $\mu > 0$. Due to the presence of the logarithmic term, for sufficiently small values of the parameter μ this represents a slightly perturbed version of a constrained minimization problem.

In this case, from the KKT theorem it is possible to find the following necessary conditions:

$$\mathbf{r}_1 := \nabla f(\mathbf{x}) - \boldsymbol{\lambda}^T \nabla_{\mathbf{x}} \mathbf{c}(\mathbf{x}) = \mathbf{0} \quad (14)$$

$$\mathbf{r}_2 := \mathbf{c}(\mathbf{x}) - \mathbf{z} = \mathbf{0} \quad (15)$$

$$\mathbf{r}_3 := \mathbf{z} \odot \boldsymbol{\lambda} - \mu \mathbf{e} = \mathbf{0}. \quad (16)$$

In order to solve this system, it is necessary to update the variables \mathbf{x} , $\boldsymbol{\lambda}$, \mathbf{z} , which can define a new variable $\mathbf{y} = [\mathbf{x}, \boldsymbol{\lambda}, \mathbf{z}]^T$. By denoting with $R(\mathbf{x})$ the matrix with rows \mathbf{r}_1 , \mathbf{r}_2 , \mathbf{r}_3 , the updating step is

$$\mathbf{y}_{k+1} = \mathbf{y}_k - \nabla R(\mathbf{y}_k)^{-1} R(\mathbf{y}_k), \quad (17)$$

where $\nabla R(\mathbf{y})$ represents the Jacobian of the system, given by:

$$\nabla R(\mathbf{y}) = J(\mathbf{x}, \boldsymbol{\lambda}, \mathbf{z}) = \begin{pmatrix} H(\mathbf{x}) + K(\mathbf{x}, \boldsymbol{\lambda}) & -\nabla \mathbf{c}(\mathbf{x})^T & \mathbf{0}_{n \times m} \\ \nabla \mathbf{c}(\mathbf{x}) & \mathbf{0}_{m \times m} & -\mathbb{1}_m \\ \mathbf{0}_{m \times n} & \text{diag}(\mathbf{z}) & \text{diag}(\boldsymbol{\lambda}) \end{pmatrix}, \quad (18)$$

where $H(\mathbf{x}) = \nabla^2 f(\mathbf{x})$, $K(\mathbf{x}, \boldsymbol{\lambda}) = -\boldsymbol{\lambda}^T \nabla^2 \mathbf{c}(\mathbf{x})$.

We want to write down the Jacobian associated with the problem (13) considering as functions and constraints the ones of the two previous exercises.

Function $f_a(\mathbf{x})$

We recall the expression of the function $f_a(\mathbf{x}) = (x_1 - 4)^2 + x_2^2$ and that of the constraints, which are written in the form $c_i(\mathbf{x}) \geq 0$, and can be recast in the vector

$$\mathbf{c}_a(\mathbf{x}) = \begin{bmatrix} 2 - x_1 - x_2 \\ x_1 \\ x_2 \end{bmatrix}. \quad (19)$$

In order to obtain the Jacobian, we have to compute the gradient and the Hessian of the function and the constraints. We report in the following their expressions.

$$\nabla f_a(\mathbf{x}) = [2(x_1 - 4), 2x_2]^T \quad H_{f_a}(\mathbf{x}) = \begin{bmatrix} 2 & 0 \\ 0 & 2 \end{bmatrix}, \quad (20)$$

$$\nabla \mathbf{c}_a(\mathbf{x}) = \begin{bmatrix} -1 & -1 \\ 1 & 0 \\ 0 & 1 \end{bmatrix} \quad \nabla^2 \mathbf{c}_a(\mathbf{x}) = [\mathbf{0}_{2 \times 2}, \mathbf{0}_{2 \times 2}, \mathbf{0}_{2 \times 2}]^T. \quad (21)$$

Therefore, $K_a(\mathbf{x}, \boldsymbol{\lambda}) = \mathbf{0}_{2 \times 2}$ as expected, since all the constraints are linear.

Function $f_b(\mathbf{x})$

As before, we recall the expression of the function $f_b(\mathbf{x}) = 2x_1 - x_2^2$ and that of the constraints, which are written also in this case in the form $c_i(\mathbf{x}) \geq 0$, and can be recast in the vector

$$\mathbf{c}_b(\mathbf{x}) = \begin{bmatrix} 1 - x_1^2 - x_2^2 \\ x_1 \\ x_2 \end{bmatrix}. \quad (22)$$

We report in the following the gradient and the Hessian of the function and the constraints:

$$\nabla f_b(\mathbf{x}) = [2, -2x_2]^T \quad H_{f_b}(\mathbf{x}) = \begin{bmatrix} 0 & 0 \\ 0 & -2 \end{bmatrix}, \quad (23)$$

$$\nabla \mathbf{c}_b(\mathbf{x}) = \begin{bmatrix} -2x_1 & -2x_2 \\ 1 & 0 \\ 0 & 1 \end{bmatrix} \quad \nabla^2 \mathbf{c}_b(\mathbf{x}) = [\nabla^2[\mathbf{c}_b]_1, \mathbf{0}_{2 \times 2}, \mathbf{0}_{2 \times 2}]^T, \quad (24)$$

where $\nabla^2[\mathbf{c}_b]_1 = -\mathbb{1}_2$ and, therefore, $K_b(\mathbf{x}, \boldsymbol{\lambda}) = \lambda_1 \mathbb{1}_2$.

Exercise 4

We want to implement the interior point method to solve numerically the constrained minimization problems of Exercise 1 and Exercise 2. We used Python as programming language and we created a file named `Project_5.py`, which can be found at this [link](#) on GitHub, where we defined the function `int_point`.

We pass explicitly to this function the variable `method`, which allows one to choose which method to implement in order to compute the Newton step and we also give the option to choose whether to consider or not the curvature term in the Jacobian. Moreover, for setting the starting values of λ and z , the function `int_point` accepts both given arrays and the string '`random`', in such case these arrays are randomly generated. To ensure reproducibility, we fix the seed, still giving the possibility of changing it by passing a specific value to the corresponding keyword argument. Finally, we choose as default values for the parameter μ of the logarithmic barrier the value $\mu = 10^{-12}$, as tolerance parameter the value 10^{-12} , and 100 as maximum number of iterations allowed. Note that these values can be user-changed as they are keyword argument of the function. All the other details can be found in the documentation. In the following, we report the code.

```
1 def int_point(func, grad_func, hess_func, constr, grad_constr, hess_constr, x0,
2   ↵ method='basic', alpha=1., beta=1., gamma=1., mu=1e-12, tol=1e-12, maxit=100,
2   ↵ 10='random', z0='random', curv=False, seed=1):
2     '''This function implements the interior point method for constrained
2     ↵ minimization problems.
3   Parameters
4   -----
5     func : function
5       Function to be minimized.
6     grad_func : function
6       Gradient of the function. It returns a 1d-array (vector)
7     hess_func : function
7       Hessian of the function. It returns a 2d-array (matrix)
8     constr : function
8       Function of the constraints. It returns a 1d-array (vector).
9     grad_constr : function
9       Gradient of the function of the constraints. It returns a 2d-array (matrix)
10    hess_constr : function
10      Hessian of the function of the constraints. It returns a 3d-array
11    x_0 : ndarray
11      Starting point
12    method : str
12      String to choose the method to perform the Newton step. Possible values are
13      - 'basic': to solve the full linear system
13      - 'first': to solve the first reduced linear system
13      - 'full': to solve the fully reduced linear system
14      Default value method='basic'.
15    alpha : float
15      Starting step-length for the parameter x. Default value alpha=1.
16    beta : float
16      Starting step-length for the parameter lambda. Default value beta=1.
17    gamma : float
17      Starting step-length for the parameter z. Default value gamma=1.
18    mu : float
```

```

32     Coefficient of the log barrier term
33     tol : float
34         Tolerance parameter for the stopping criterion. Default value tol=1e-12
35     maxit : int
36         Maximum number of iterations. Default value maxit=100
37     l0 : str or ndarray
38         Starting value of the Lagrange multipliers. Default value l0='random', to
39             → generate uniformly distributed random values in the interval [1e-16,10).
40     z0 : str or ndarray
41         Starting value of the slack variable. Default value z0='random', to generate
42             → uniformly distributed random values in the interval [1e-16,10).
43     curv : bool
44         Boolean value to choose whether to discard (curv=False) or not (curve=True)
45             → the curvature term in the expression of the Jacobian of the system.
46             → Default value curv=False.
47     seed : int
48         Parameter to fix the seed. Default value seed=1
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74

```

Results

results : dict

Dictionary of the results given by the function. It contains the following

- items:
- 'convergence' : (bool) True if the algorithm converges, False if it doesn't converge
- 'n_iter' : (int) progressive number of final iteration
- 'x_min' : (ndarray) computed point at which the minimum of the function is reached
- 'f_min' : (float) computed minimum value of the function
- 'x_interm' : (list) list of the intermediate points
- 'lambda_interm' : (list) list of the intermediate Lagrange multipliers
- 'z_interm' : (list) list of the intermediate slack variables

'''

Check if the starting point is in the feasible set

if any(constr(x0) < 0):

print(f'Starting point x0 = {x0} is not feasible')

return False

Assign initial values to the the variables x_old, lambda_old and z_old

x_old = x0

lambda_old, z_old = 10, z0

m, n = len(constr(x_old)), len(x_old)

Fix the seed and initialize lambda_old and z_old by using the uniform random

→ distribution if required

np.random.seed(seed=seed)

if type(l0) == str and l0 == 'random':

lambda_old = np.random.uniform(low=1e-16, high= 10., size=m)

if type(z0) == str and z0 == 'random':

z_old = np.random.uniform(low=1e-16, high= 10., size=m)

```

75
76 # Compute the vector R
77 r1 = grad_func(x_old) - lambda_old @ grad_constr(x_old)
78 r2 = constr(x_old) - z_old
79 r3 = z_old*lambda_old - mu
80 R = np.array([*r1, *r2, *r3])
81
82 # Append the starting values of the variables to the corresponding lists
83 x_interm = [x0]
84 lambda_interm = [lambda_old]
85 z_interm = [z_old]
86
87 # Cycle until the stopping criterion is satisfied or k reaches the maximum number
88 # of iterations
88 k = 0
89 while(np.linalg.norm(R)>tol and k < maxit):
90     a0, b0, g0 = alpha, beta, gamma
91
92     # Compute matrices and vectors entering the expression of the linear system
93     Z = np.diag(z_old)
94     Z_inv = np.linalg.inv(Z)
95     grad_c = grad_constr(x_old)
96     Lambda = np.diag(lambda_old)
97     hess_f = hess_func(x_old)
98     K = 0
99     if curv == True:
100         hess_c = hess_constr(x_old)
101         K = - lambda_old @ hess_c
102
103     # Choose the method to perform the Newton step and compute dx, dl, dz
104     # Solve the full system
105     if method == 'basic':
106         Jacobian = np.block([[ hess_f + K, - grad_c.T, np.zeros((n,m)) ],
107                               [ grad_c , np.zeros((m,m)), - np.eye(m)],
108                               [ np.zeros((m,n)), Z, Lambda ]])
109         p = np.linalg.solve(Jacobian, -R)
110         dx = p[:n]
111         dl = p[n:n+m]
112         dz = p[n+m:]
113
114     # Solve the first reduced system
115     if method == 'first':
116         Lambda_inv = np.linalg.inv(Lambda)
117         matrix = np.block([[hess_f + K, -(grad_c).T],
118                           [grad_c, Lambda_inv @ Z]])
119         vector = np.array([*r1, *(r2 + Lambda_inv @ r3)])
120
121         p = np.linalg.solve(matrix,-vector)
122         dx = p[:n]
123         dl = p[n:]
124         dz = - Lambda_inv @ (r3 + Z @ dl)

```

```

125
126     # Solve the fully reduced system
127     if method == 'full':
128         matrix = hess_f + K + (grad_c).T @ (Z_inv @ Lambda @ grad_c )
129         vect = - r1 - (grad_c).T @ Z_inv @ (r3 + Lambda @ r2)
130
131         dx = np.linalg.solve(matrix,vect)
132         dl = - Z_inv @ Lambda @ r2 - Z_inv @ r3 -Z_inv @ Lambda @ grad_c @ dx
133         dz = - np.linalg.inv(Lambda) @ (r3 + Z @ dl)
134
135     # Update the values of the variables
136     x_new = x_old + a0*dx
137     lambda_new = lambda_old + b0*dl
138     z_new = z_old + g0*dz
139
140     # Check if the updated values satisfy the required conditions and re-update
141     # them until necessary
142     while any(constr(x_new) < 0 ):
143         a0 = a0/2
144         x_new = x_old + a0*dx
145
146         while any(lambda_new < 0 ):
147             b0 = b0/2
148             lambda_new = lambda_old + b0*dl
149
150         while any(z_new <= 0 ):
151             g0 = g0/2
152             z_new = z_old + g0*dz
153
154     # Append the new values of the variables to the corresponding lists
155     x_interm.append(x_new)
156     lambda_interm.append(lambda_new)
157     z_interm.append(z_new)
158
159     # Compute the new values of R
160     r1 = grad_func(x_new) - lambda_new @ grad_constr(x_new)
161     r2 = constr(x_new) - z_new
162     r3 = z_new*lambda_new - mu
163     R = np.array([*r1, *r2, *r3])
164
165     x_old = x_new
166     lambda_old = lambda_new
167     z_old = z_new
168     k = k + 1
169
170     # Check if the convergence is reached
171     conv = True
172     if k == maxit:
173         conv = False
174
f_min = func(x_new)

```

```

175
176     results = {'convergence' : conv, 'n_iter' : k , 'x_min' : x_new, 'f_min' :
177             ↵ f_min, 'x_interm' : x_interm, 'lambda_interm' : lambda_interm, 'z_interm'
178             ↵ : z_interm}
179
180     return results

```

Function (a)

First and foremost, after having chosen a representative starting point, i.e. $x_0 = (0.5, 0.5)$, we run the function `int_point` by using the three different methods to compute the Newton step, namely *basic* which solves the complete linear system, *first* which solves the first reduced linear system, and *full* which solves the fully reduced linear system. We report in the following the results that we obtain by choosing the components of λ_0 and z_0 uniformly distributed in the interval $[10^{-16}, 10]$. Note that we use the same vectors λ_0 and z_0 in all the three cases.

```

convergence = True, with 9 steps and method = basic
starting point = (0.5, 0.5)
mu = 1e-12, lambda_0 = (6.2, 1.4, 3.3), z_0 = (2.8, 5.9, 5.5)
min point = (1.9999999999948, 2.5768276401549883e-13)
min value = 4.00000000000208
lambda* = (4.0000000000015214, 4.818155423247372e-13, 4.0000000000020375)
z* = (2.6239080128549816e-13, 1.9999999999948, 2.576828630208037e-13)

convergence = True, with 9 steps and method = first
starting point = (0.5, 0.5)
mu = 1e-12, lambda_0 = (6.2, 1.4, 3.3), z_0 = (2.8, 5.9, 5.5)
min point = (1.9999999999948, 2.5768276401549883e-13)
min value = 4.00000000000208
lambda* = (4.0000000000015214, 4.818148918034337e-13, 4.0000000000020375)
z* = (2.623908012854933e-13, 1.999999999994804, 2.576828630208021e-13)

convergence = True, with 9 steps and method = full
starting point = (0.5, 0.5)
mu = 1e-12, lambda_0 = (6.2, 1.4, 3.3), z_0 = (2.8, 5.9, 5.5)
min point = (1.9999999999948, 2.576828630208037e-13)
min value = 4.00000000000208
lambda* = (4.000000000001522, 4.818155423247372e-13, 4.000000000002037)
z* = (2.623908012854933e-13, 1.999999999994802, 2.576828630208037e-13)

```

We can observe that the three methods converge to the solution with the same number of steps, as expected since they solve equivalent linear systems. The difference between them may be in the time they take to arrive at the solution, due to the fact that they solve systems of different dimensionalities. However, in this case, this difference has not been appreciable as the system that we solve is already of small size. Moreover, as expected, the error on the computed quantities is of $\mathcal{O}(\mu)$ and the values of the components of z^* , that is the vector of slack variables in correspondence with the computed minimum point x^* , are equal to the values of the constraints at x^* .

After this first analysis, we have decided to use the method 'basic' as we expect this to be the most stable one. We analyze different starting points x_0 as well as different values of λ_0 and z_0 . In order to present the obtained results, we first consider a fixed starting point and a fixed μ value, while varying λ_0 and z_0 . Then, for a fixed triad of values x_0 , λ_0 , and z_0 , we compare the number of iterations

towards convergence for two different μ values. In all the plots we report the intermediate points as they reach the constrained minimum point.

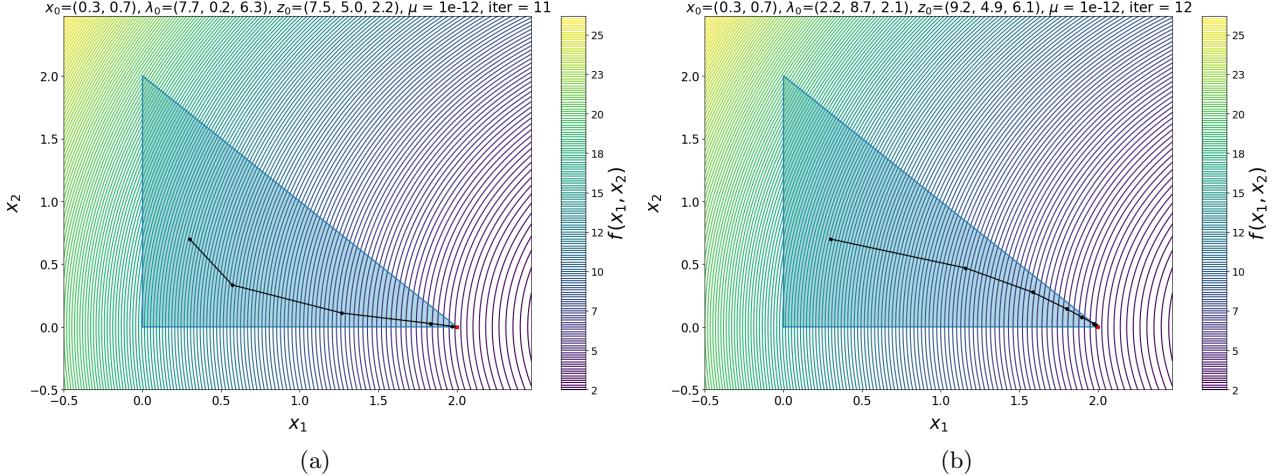


Figure 1: Contour plot of the function $f_a(x_1, x_2)$ with $\mu = 10^{-12}$, for fixed \mathbf{x}_0 and different values of $\boldsymbol{\lambda}_0$, and \mathbf{z}_0 , reported in the title of the plots. The constrained set is represented by the triangle in blue. In red is the constrained minimum point the interior point algorithm converges to.

It can be noted that different choices of the initial values of $\boldsymbol{\lambda}$ and \mathbf{z} imply, as expected, a different path towards convergence since the descent direction depend on them. On the other hand, we also expect that, for a fixed value of $\boldsymbol{\lambda}$ and \mathbf{z} , when considering different values of the parameter μ , the path remains approximatively the same while the number of iterations needed to reach convergence changes, decreasing as μ increases. Below we report the plots obtained when considering as values of $\boldsymbol{\lambda}_0$ and \mathbf{z}_0 the ones reported in Fig. 1a.

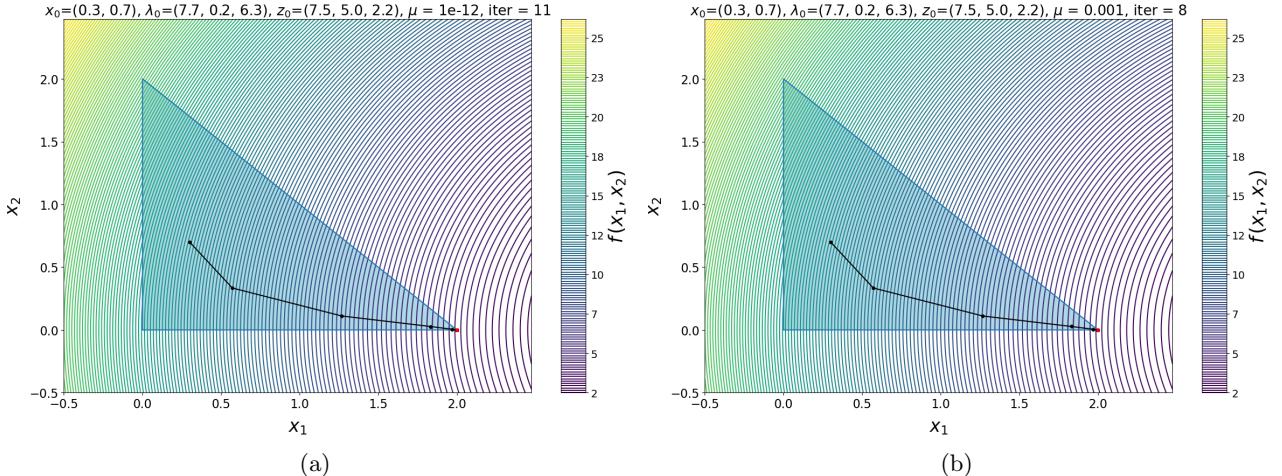


Figure 2: Contour plot of the function $f_a(x_1, x_2)$ with $\mu = 10^{-12}$ (Panel (a)) and $\mu = 10^{-3}$ (Panel (b)), together with the constrained set, in blue. Values of \mathbf{x}_0 , $\boldsymbol{\lambda}_0$, and \mathbf{z}_0 in the titles of the plots. In red is the constrained minimum point the interior point algorithm converges to.

To better compare the values of the computed minimum points and those of $\boldsymbol{\lambda}$, and \mathbf{z} at the final iteration, we report below the numerical results that we find in correspondence with the plots in Fig 2.

```

x_0 = (0.3, 0.7), lambda_0 = (3.3, 3.6, 0.4), z_0 = (3.0, 5.8, 3.2)

mu = 1e-12, iter = 11
min point = (1.999999999994307, 2.626380016246266e-13)
min value = 4.000000000002277
lambda_fin = (4.000000000001556, 4.174412551738449e-13, 4.000000000002081)
z_fin = (3.065889841752647e-13, 1.999999999994307, 2.626387608486303e-13)

mu = 0.001, iter = 8
min point = (1.9995002185977582, 0.0002498750975605184)
min value = 4.001999437827982
lambda_fin = (4.0014996877813, 0.0005001249768169601, 4.001999437976421)
z_fin = (0.0002499063046813427, 1.9995002185977582, 0.0002498750975605184)

```

As we can see from these results, in both cases the algorithm converges to the exact minimum point $\mathbf{x}^* = (2, 0)^T$ and to the vectors $\boldsymbol{\lambda}^* = (4, 0, 4)^T$ and $\mathbf{z}^* = (0, 2, 0)^T$, with an error that is of $\mathcal{O}(\mu)$ and with a number of steps which decreases as μ increases. We can also note that the computed value of the slack variable is compatible, within this error, with the value of the function $\mathbf{c}(\mathbf{x})$ at \mathbf{x}^* .

Finally, for the same triad of vectors, we report a plot with the number of iterations until convergence as a function of $\log_{10}(\mu)$.

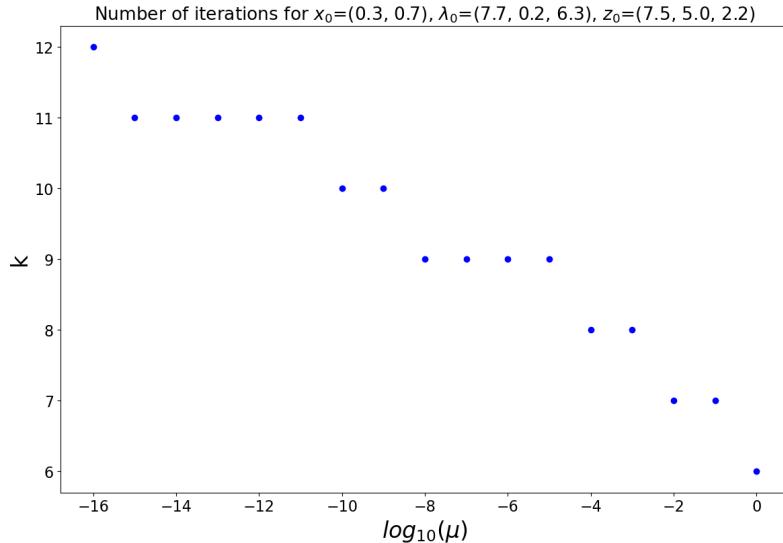


Figure 3: Scatter plot of the number of iterations until convergence as a function of $\log_{10}(\mu)$, for the triad of vectors \mathbf{x}_0 , $\boldsymbol{\lambda}_0$, and \mathbf{z}_0 reported in the title of the plot.

For demonstration purposes, we report in the following other convergence paths emerging when considering different initial points and different values of $\boldsymbol{\lambda}_0$ and \mathbf{z}_0 . In this case, we fixed $\mu = 10^{-12}$.

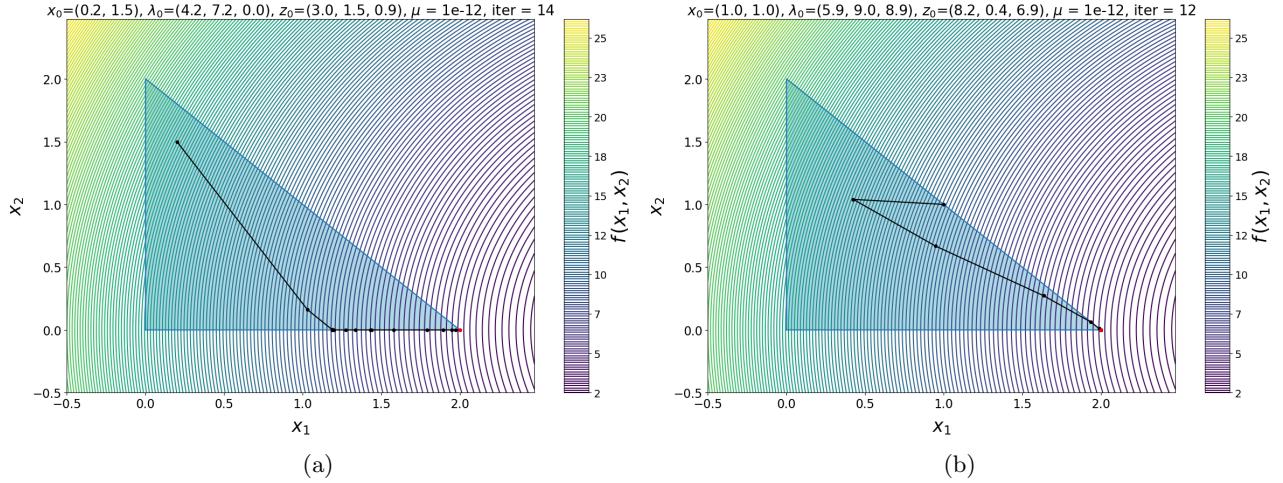


Figure 4: Contour plot of the function $f_a(x_1, x_2)$ with $\mu = 10^{-12}$ and different values of \mathbf{x}_0 , $\boldsymbol{\lambda}_0$, and \mathbf{z}_0 , reported in the title of the plots. The constrained set is represented by the triangle in blue. In red is the constrained minimum point the interior point algorithm converges to.

Function (b)

In this case, from the previous analysis, we know that the KKT system has two solutions, namely $\mathbf{x}_1^* = (0, 1)$ with $\boldsymbol{\lambda}_1^* = (1, 2, 0)$, and $\mathbf{x}_2^* = (0, 0)$ with $\boldsymbol{\lambda}_2^* = (0, 2, 0)$. Given that $f_b(\mathbf{x}_1^*) = -1$ and $f_b(\mathbf{x}_2^*) = 0$, we know that $\mathbf{x}_1^* = (0, 1)$ is the unique constrained minimum of this problem. For this reason, we expect that the interior point algorithm may not always converge to the point \mathbf{x}_1^* but, sometimes, depending on the values of the input vectors \mathbf{x}_0 , $\boldsymbol{\lambda}_0$, and \mathbf{z}_0 , it may also converge to the point \mathbf{x}_2^* .

In order to study this problem, we use the method '`'basic'`', which solves the complete linear system to compute the Newton step, and we set the variable `curv = True`, so to consider the curvature term in the expression of the Jacobian of the problem, since, in this case, the constraints are quadratic in x_1, x_2 .

We report in the Fig. 5 the plots of the convergence paths toward the point \mathbf{x}^* that we find when considering different initial values of \mathbf{x}_0 , $\boldsymbol{\lambda}_0$, and \mathbf{z}_0 , while fixing $\mu = 10^{-12}$. In the following are the numerical results that we obtain for the first one of the four cases analyzed.

```
convergence = True, with 19 steps and method = basic
starting point = (0.5, 0.5)
mu = 1e-12, lambda_0 = (1.5, 7.4, 2.6), z_0 = (5.3, 0.1, 9.2)
min point = (5.000000371825291e-13, 0.99999999999995)
min value = -0.9999999999999999
lambda_fin = (1.00000000000005, 2.00000000000001, 9.999941060547193e-13)
z_fin = (1.0000141425515907e-12, 5.000000371767508e-13, 0.99999999999995)
```

We can note that the algorithm converges to the vectors \mathbf{x}_1^* , $\boldsymbol{\lambda}_1^*$, and $\mathbf{z}_1^* := \mathbf{c}_b(\mathbf{x}_1^*)$, within an error of $\mathcal{O}(\mu)$. The same hold for the other three considered cases in Fig. 5.

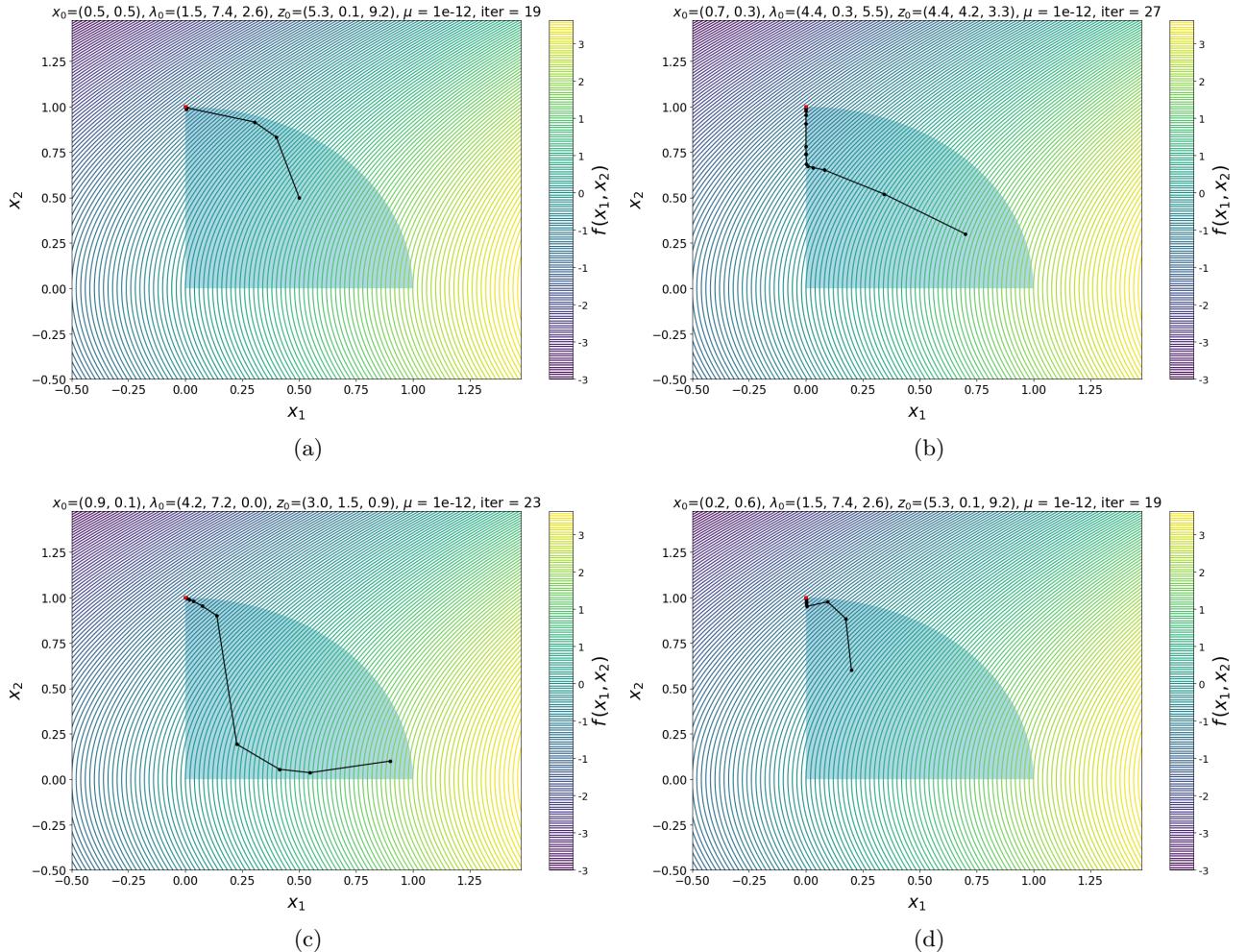


Figure 5: Contour plot of the function $f_b(x_1, x_2)$ with $\mu = 10^{-12}$ and different values of \mathbf{x}_0 , $\boldsymbol{\lambda}_0$, and \mathbf{z}_0 , reported in the title of the plots. The constrained set is represented in blue. In red is the constrained minimum point the interior point algorithm converges to.

As we can see from these plots, in all these cases, the algorithm converges to the right constrained minimum $\mathbf{x}_1^* = (0, 1)$. However, we also observe situations in which the algorithm converges to the point $\mathbf{x}_2^* = (0, 0)$ with $\boldsymbol{\lambda}_2^* = (0, 2, 0)$ and $\mathbf{z}_2^* = (1, 0, 0)$, which is the other solution of the KKT system. We report, as an example, the numerical results that we obtain for the first one of the two cases in Fig. 6, where this behavior is observed.

```
convergence = False, with 100 steps and method = basic
starting point = (0.1, 0.1)
mu = 1e-12, lambda_0 = (7.7, 0.2, 6.3), z_0 = (7.5, 5.0, 2.2)
min point = (5e-13, 5.717234610811308e-07)
min value = 6.731322840494127e-13
lambda_fin = (9.999999999948479e-13, 2.0, 1.2049499133358223e-06)
z_fin = (1.0000000000019542, 5e-13, 5.717234610811308e-07)
```

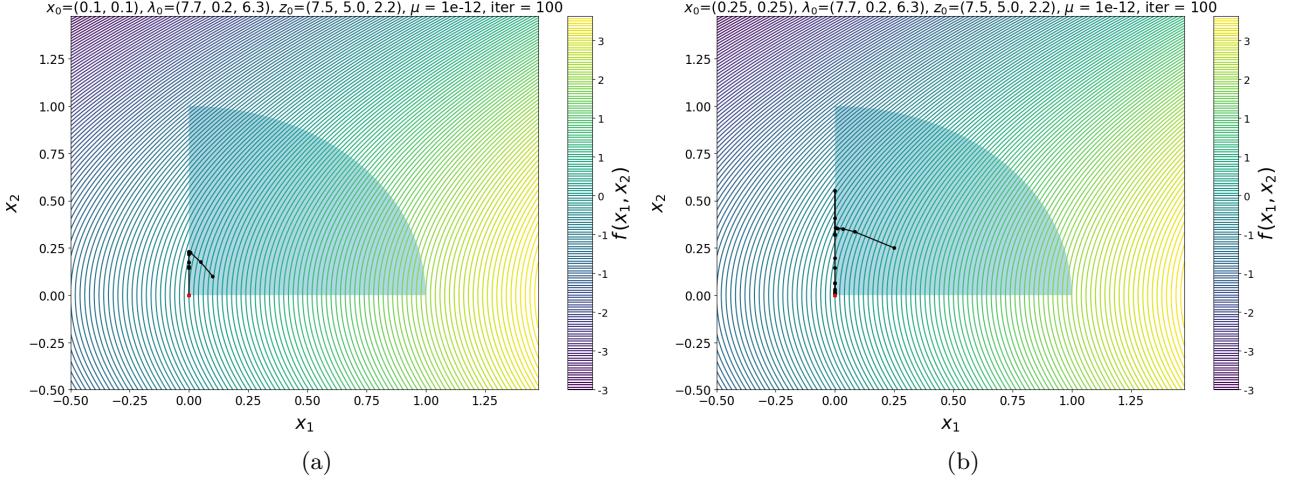


Figure 6: Contour plot of the function $f_b(x_1, x_2)$ with $\mu = 10^{-12}$ and different values of \mathbf{x}_0 , $\boldsymbol{\lambda}_0$, and \mathbf{z}_0 , reported in the title of the plots. The constrained set is represented in blue. In red is the constrained minimum point the interior point algorithm converges to.

Since in this case the KKT system has two solutions and the dependence of the convergence of the algorithm from the initial parameters is expected to be stronger than the previous case, we have decided to deepen how this convergence depends on $\boldsymbol{\lambda}_0$ and \mathbf{z}_0 . In order to do it, we have chosen two representative initial points, namely $\mathbf{x}_0 = (0.5, 0.5)$ and $\mathbf{x}_0 = (0.1, 0.1)$. For these points, we have found that, for the considered values of $\boldsymbol{\lambda}_0$ and \mathbf{z}_0 , the algorithm converges, respectively, to \mathbf{x}_1^* , which is the right minimum point, and \mathbf{x}_2^* (see Fig. 5a, 6a). However, since $\boldsymbol{\lambda}_0$ and \mathbf{z}_0 can be chosen randomly, to simplify this analysis we have decided to consider vectors of the form $\boldsymbol{\lambda}_0 = \lambda(1, 1, 1)$, $\mathbf{z}_0 = z(1, 1, 1)$ and we have chosen to vary λ and z in the range $[10^{-12}, 10]$, with a step of 0.1. For each of these values, we have run the algorithm, both considering the curvature term and not considering it, and we have stored the number of iterations until convergence. Finally, to distinguish when the algorithm arrives to the point \mathbf{x}_2^* , we have manually assigned to k the value $k = 150$ when the algorithm converges to a point $\tilde{\mathbf{x}}$ such that $\|\tilde{\mathbf{x}} - \mathbf{x}_2^*\|_2 < 10^{-2}$.

We report in Fig. 7 the obtained results.

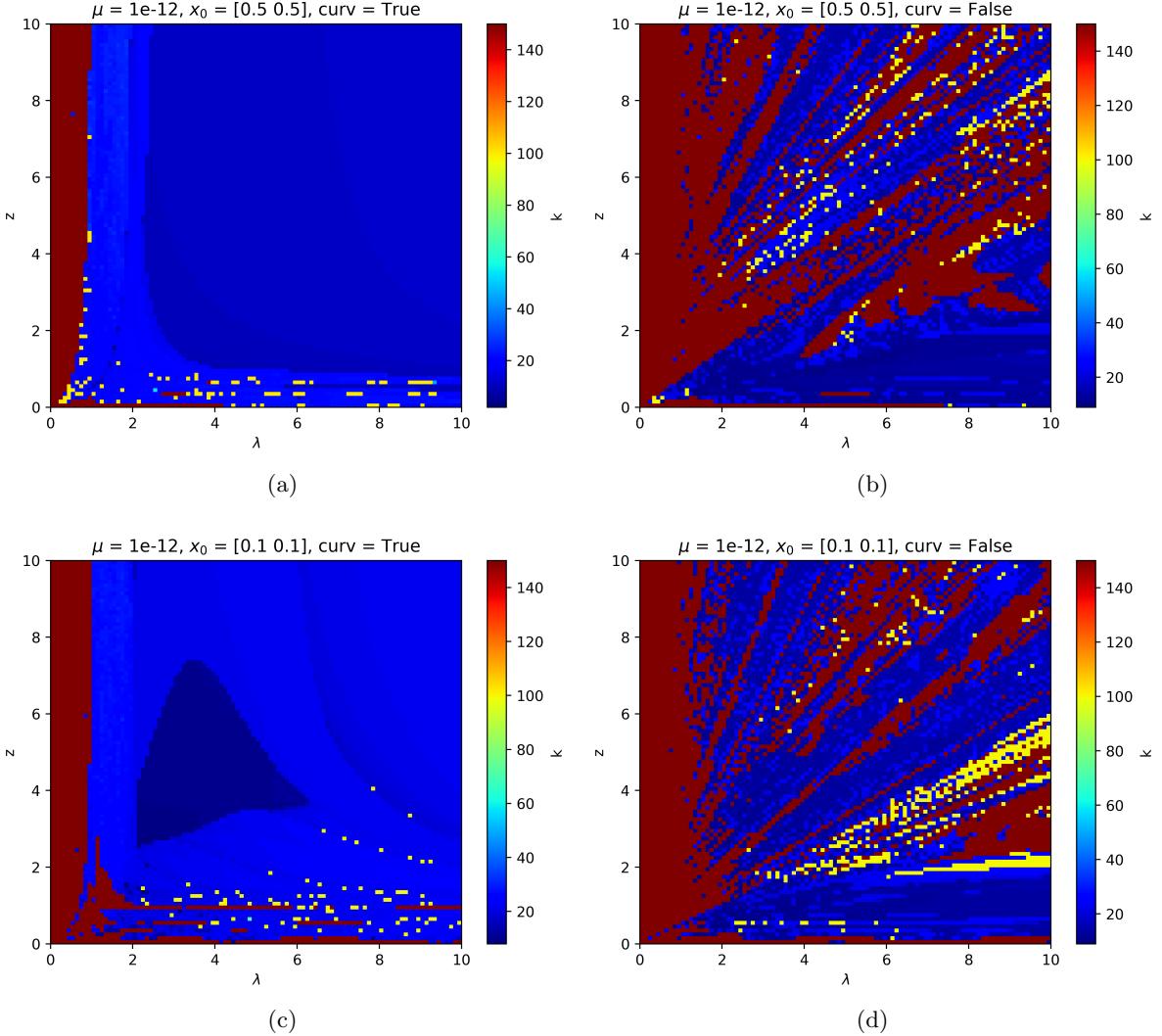


Figure 7: Colorplot of the number of iterations until convergence for the function $f_b(\mathbf{x})$, with $\mu = 10^{-12}$ and different values of \mathbf{x}_0 , reported in the title of the plots. The parameter $\text{curv} = \text{True}$ indicates the presence of the curvature term in the expression of the Jacobian of the system.

The first thing that emerges from these plots is that the addition of the curvature term strongly improves the convergence of the algorithm to the right minimum point. This is expected since the constraint functions are quadratic in x_1, x_2 . Moreover, for chosen the tolerance parameter and the maximum number of iterations allowed, we can also observe that, most of the times, when the algorithm does not converge to \mathbf{x}_1^* it stops in the neighborhood of the other solution of the KKT system, namely \mathbf{x}_2^* , while there are few points (in yellow) where the algorithm does not converge neither to \mathbf{x}_1^* nor to \mathbf{x}_2^* .