Numerical Linear Algebra Homework Project 5: Constrained Optimization

Catalano Giuseppe, Cerrato Nunzia

Exercise 1

We want to minimize the function

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

subject to the constraints

$$x_1 + x_2 \leq 2, \quad x_1 \geq 0, \quad x_2 \geq 0, \tag{2}$$

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

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

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:

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

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:

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

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:

$$\begin{cases} \text{stationarity} & (2(x_1 - 4) + \lambda_1 - \lambda_2, 2x_2 + \lambda_1 - \lambda_3) = (0, 0) \\ \text{primal feasibility} & (2 - x_1 - x_2, x_1, x_2) \geq (0, 0, 0) \\ \text{dual feasibility} & (\lambda_1, \lambda_2, \lambda_3) \geq (0, 0, 0) \\ \text{complementarity} & (\lambda_1(2 - x_1 - x_2), \lambda_2 x_1, \lambda_3 x_2) = (0, 0, 0) \end{cases} \tag{6}$$

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)	$\mathbf{x}^* \in \mathring{\mathcal{C}}_a$
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, $\boldsymbol{\lambda}^* = (4, 0, 4)$ 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

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

under the constraints

$$x_1^2 + x_2^2 \leq 1, \quad x_1 \geq 0, \quad x_2 \geq 0, \tag{8}$$

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

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

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

$$2x_1 - x_2^2 \geq -x_2^2 \geq -1, \tag{10}$$

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:

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

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

$$\begin{cases} \text{stationarity} & (2 + 2\lambda_1 x_1 - \lambda_2, -2x_2 + 2\lambda_1 x_2 - \lambda_3) = (0, 0) \\ \text{primal feasibility} & (1 - x_1^2 - x_2^2, x_1, x_2) \geq (0, 0, 0) \\ \text{dual feasibility} & (\lambda_1, \lambda_2, \lambda_3) \geq (0, 0, 0) \\ \text{complementarity} & (\lambda_1(1 - x_1^2 - x_2^2), \lambda_2 x_1, \lambda_3 x_2) = (0, 0, 0) \end{cases} \tag{12}$$

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 $\lambda_2^* = 0$, 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 $\boldsymbol{\lambda}_1^* = (1, 0, 0)$ 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

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

where $f : \mathbb{R}^n \to \mathbb{R}$, is the function one wants to find the minimum, $\mathbf{c} : \mathbb{R}^n \to \mathbb{R}^m$ is a vector defining the constraints, which are assumed to be $c_i(\mathbf{x}) \geq 0,\ \forall i = 1, \ldots, m$, $\mathbf{z} \in \mathbb{R}^m$ is called slack variable and, finally, $\mathbf{e}^T = (1, \ldots, 1) \in \mathbb{R}^m$ 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:

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

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

$$\mathbf{r}_2 \coloneqq \mathbf{z} \odot \boldsymbol{\lambda} - \mu \mathbf{e} = \mathbf{0}. \tag{16}$$

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

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

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

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

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

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

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.

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

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

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

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

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

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

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

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 defined the function ***int_point***, which can be found in the file Project_5.py at this **link** on GitHub.

In writing this function, we pass explicitly to the function the variable **method**, which allows one to choose which method to implement in order to compute the Newton step and we also gave the option to choose whether to consider or not the curvature term in the Jacobian. Moreover, for setting the starting values of $\boldsymbol{\lambda}$ and $\mathbf{z}$, the function *** int_point *** accepts both given arrays and the string 'random', in such case these arrays are randomly generated. To ensure reproducibility, we fixed the seed, still giving the possibility of changing it by passing a specific value to the corresponding keyword argument. Finally, we chose 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.

```
def int_point(func, grad_func, hess_func, constr, grad_constr, hess_constr, x0,
    method='basic', alpha=1., beta=1., gamma=1., mu=1e-12, tol=1e-12, maxit=100,
    l0='random', z0='random', curv=False, seed=1):
    '''This function implements the interior point method for constrained
       minimization problems.
    Parameters
    ----------
    func : function
        Function to be minimized. It must be :math:`f : \mathbb{R}^{n} \rightarrow
           \mathbb{R}`
    grad_func : function
        Gradient of the function. It returns a 1d-array (vector)
    hess_func : function
        Hessian of the function. It returns a 2d-array (matrix)
    constr : function
        Function of the constraints. It must be :math:`c : \mathbb{R}^{n} \rightarrow
           \mathbb{R}^{m}` and it must return a 1d-array (vector).
    grad_constr : function
        Gradient of the function of the constraints. It returns a 2d-array (matrix)
    hess_constr : function
        Hessian of the function of the constraints. It returns a 3d-array
    x_0 : ndarray
        Starting point
    method : str
        String to choose the method to perform the Newton step. Possible values are
        - 'basic': to solve the full linear system
        - 'first': to solve the first reduced linear system
        - 'full': to solve the fully reduced linear system
        Default value method='basic'.
    alpha : float
        Starting step-lenght for the parameter :math:`x`. Default value alpha=1.
    beta : float
        Starting step-lenght for the parameter :math:`\lambda`. Default value beta=1.
    gamma : float
        Starting step-lenght for the parameter :math:`z`. Default value gamma=1.
```

```
    mu : float
       Coefficient of the log barrier term
    tol : float
       Tolerance parameter for the stopping criterion. Default value tol=1e-12
    maxit : int
       Maximum number of iterations. Default value maxit=100
    l0 : str or ndarray
       Starting values of the Lagrange multipliers. Deafult value l0='random', to
        ↪ generate uniformly distributed random values in the interval [1e-16,10).
    z0 : str or ndarray
       Starting values of the slack variable. Deafult value z0='random', to generate
        ↪ uniformly distributed random values in the interval [1e-16,10).
    curv : bool
       Boolean value to choose whether to discard the curvature term in the
        ↪ expression of the Jacobian of the system. Default value curv=False.
    seed : int
       Parameter to fix the seed. Default value seed=1

    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 = l0, 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
        ↪  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
            ↪  them until necessary
141         while any(constr(x_new) < 0 ):
142            a0 = a0/2
143            x_new = x_old + a0*dx
144
145         while any(lambda_new < 0 ):
146            b0 = b0/2
147            lambda_new = lambda_old + b0*dl
148
149         while any(z_new <= 0 ):
150            g0 = g0/2
151            z_new = z_old + g0*dz
152
153         # Append the new values of the variables to the corresponding lists
154         x_interm.append(x_new)
155         lambda_interm.append(lambda_new)
156         z_interm.append(z_new)
157
158         # Compute the new values of R
159         r1 = grad_func(x_new) - lambda_new @ grad_constr(x_new)
160         r2 = constr(x_new) - z_new
161         r3 = z_new*lambda_new - mu
162         R = np.array([*r1, *r2, *r3])
163
164         x_old = x_new
165         lambda_old = lambda_new
166         z_old = z_new
167         k = k + 1
168
169         # Check if the convergence is reached
170         conv = True
171         if k == maxit:
172             conv = False
173
174         f_min = func(x_new)
```

```
175
176         results = {'convergence' : conv, 'n_iter' : k , 'x_min' : x_new, 'f_min' :
            ↪  f_min, 'x_interm' : x_interm, 'lambda_interm' : lambda_interm, 'z_interm'
            ↪  : z_interm}
177
178         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 ** with the three different methods to compute the Newton step, namely *basic* which solve the complete linear system, *first* which solve the first reduced linear system, and *full* which solve the fully reduced linear system. We report in the following the results that we obtain by choosing the component of $\boldsymbol{\lambda}_0$ and $\mathbf{z}_0$ as uniformly distributed in the interval $\left[10^{-16}, 10\right)$.

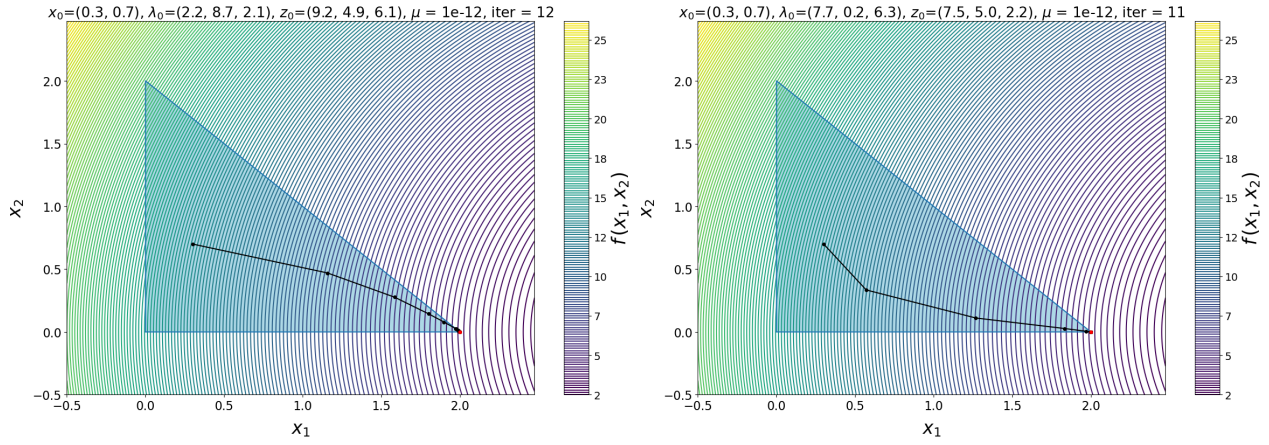
```
convergence = True, with 12 steps and method = basic
starting point = (0.5, 0.5)
mu = 1e-12, lambda_0 = (7.8, 3.1, 1.7), z_0 = (6.5, 8.2, 1.2)
min point = (1.999999999999499, 2.500564859342358e-13)
min value = 4.000000000002004

convergence = True, with 12 steps and method = first
starting point = (0.5, 0.5)
mu = 1e-12, lambda_0 = (7.8, 3.1, 1.7), z_0 = (6.5, 8.2, 1.2)
min point = (1.999999999999499, 2.500569196151048e-13)
min value = 4.000000000002004

convergence = True, with 12 steps and method = full
starting point = (0.5, 0.5)
mu = 1e-12, lambda_0 = (7.8, 3.1, 1.7), z_0 = (6.5, 8.2, 1.2)
min point = (1.999999999999499, 2.500565493176935e-13)
min value = 4.000000000002004
```

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.

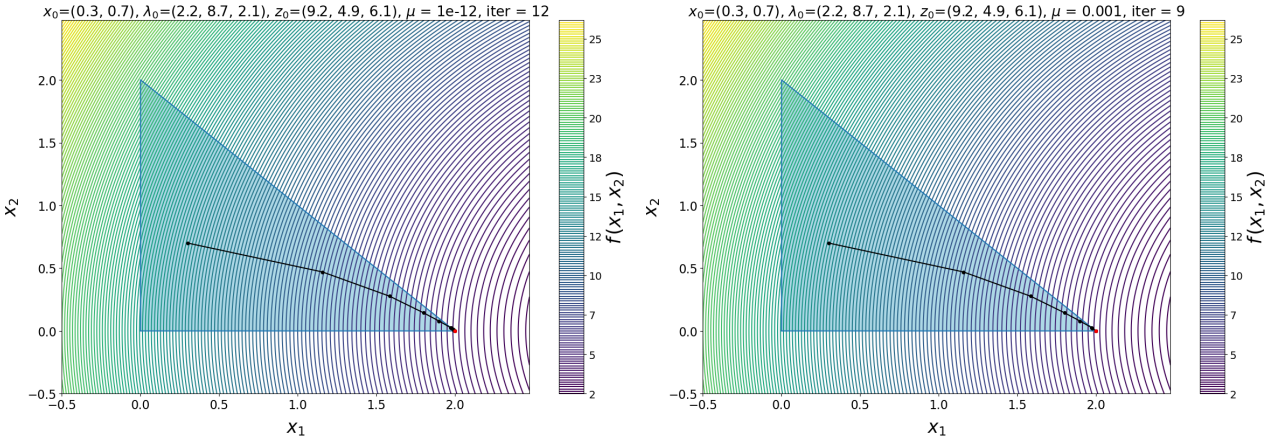
After this first analysis, we decided to use the method 'basic' as this should be the most stable one. We analyze different starting points $\mathbf{x}_0$ as well as different values of $\boldsymbol{\lambda}_0$ and $\mathbf{z}_0$. In order to present the obtained results, we first consider a fixed starting point and a fixed μ value, while varying $\boldsymbol{\lambda}_0$ and $\mathbf{z}_0$. Then, for a fixed triad of values $\mathbf{x}_0$, $\boldsymbol{\lambda}_0$, and $\mathbf{z}_0$, 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.



(a) (b)

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.



(a) (b)

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 title of the plots. In red is the constrained minimum point the interior point algorithm converges to.

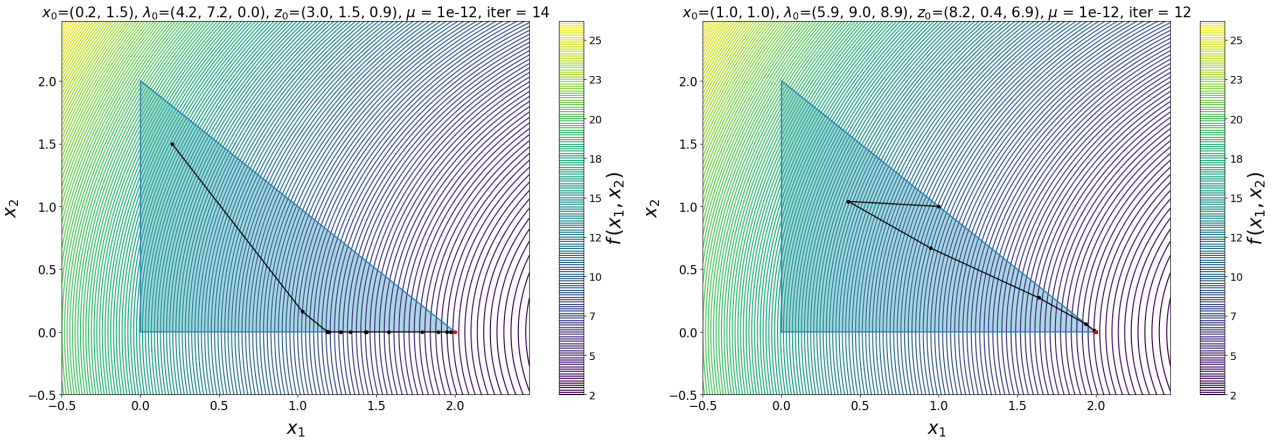
```
x_0 = (0.3, 0.7), lambda_0 = (3.3, 3.6, 0.4), z_0 = (3.0, 5.8, 3.2)

mu = 0.001
min point = (1.9995002185977582, 0.0002498750975605184)
min value = 4.001999437827982
lambda* = (4.0014996877813, 0.0005001249768169601, 4.001999437976421)
z* = (0.0002499063046813427, 1.9995002185977582, 0.0002498750975605184)
```

```
mu = 1e-12
min point = (1.9999999999994307, 2.626380016246266e-13)
min value = 4.000000000002277
lambda* = (4.000000000001556, 4.174412551738449e-13, 4.000000000002081)
z* = (3.065889841752647e-13, 1.9999999999994307, 2.626387608486303e-13)
```

As we can see from these results, for $\mu = 10^{-12}$ the algorithm returns a constrained minimum point that is much closer to the exact value with respect to the case with small μ, and the same holds for the values of $\boldsymbol{\lambda}^*$ and z^* which are, respectively, the Lagrange multiplier and the slack variable computed in correspondence with the minimum point.

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



(a) (b)

Figure 3: 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 $\mathbf{x}_1^* = (0, 1)$, which we know is the unique constrained minimum, and $\mathbf{x}_2^* = (0, 0)$. For this reason, we expect that the interior point algorithm may not always converge at the point $\mathbf{x}_1^*$ and that, sometimes, depending on the values of the input vectors $\mathbf{x}_0$, $\boldsymbol{\lambda}_0$, and $\mathbf{z}_0$, it will also converge at the point $\mathbf{x}_2^*$

We report in the following the plots that we obtained when considering different initial values of the involved vectors, fixing $\mu = 10^{-12}$.

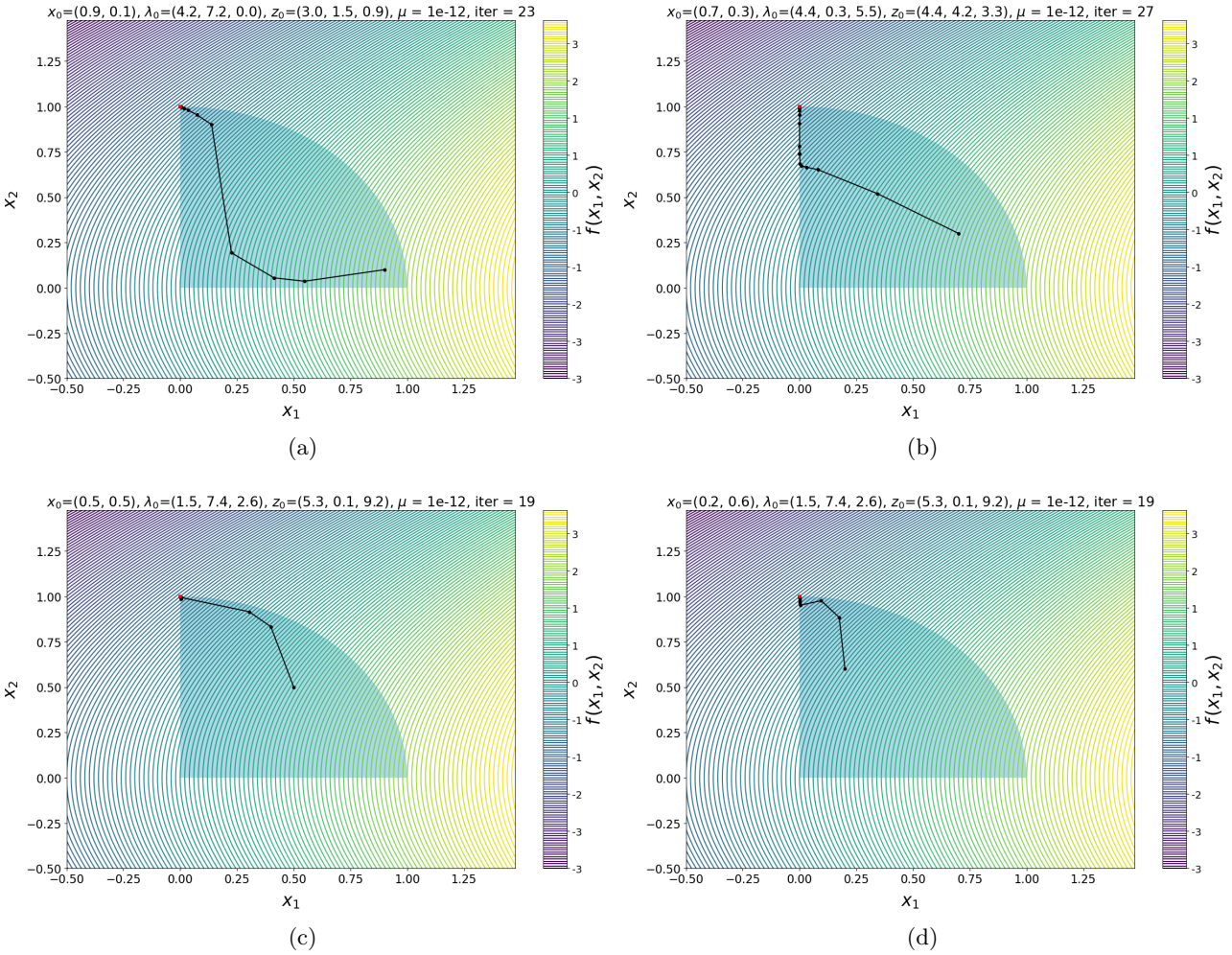
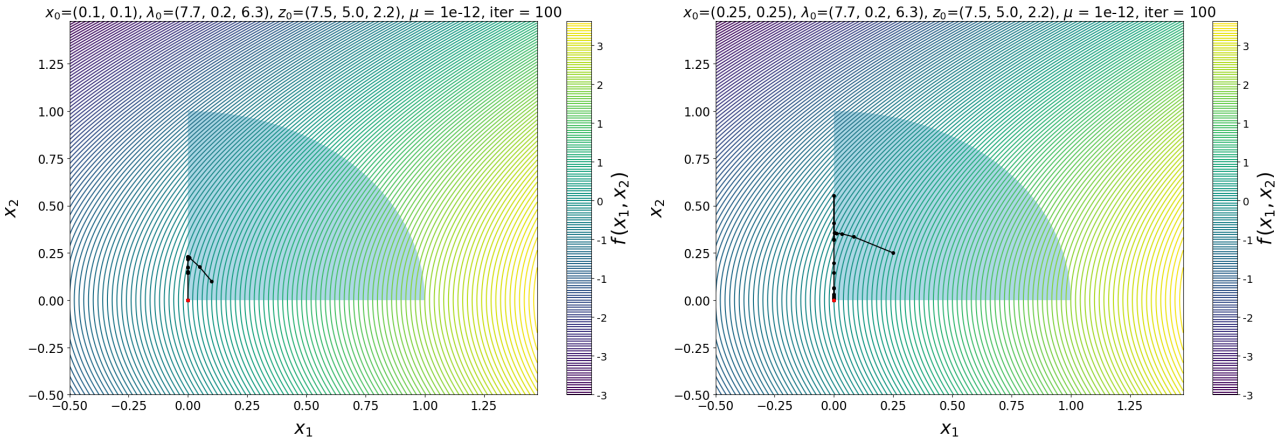


Figure 4: Contour plot of the function $f_b(x_1, x_2)$ with $\mu = 10^{-12}$ and different values of $\mathbf{x}_0$, $\boldsymbol{\lambda}_0$, and $\mathbf{z}_0$, reported in the title of the plots. The constrained set is represented in blue. In red is the constrained minimum point the interior point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

In all these cases, the algorithm converges to the real constrained minimum point $\mathbf{x}^* = (0, 1)$. However, we also observed situations in which the algorithm converges to the point $\mathbf{x} = (0, 0)$, which is the other solution of the KKT system. We report an example below.



(a) (b)

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.

2 punti x0 che convergono al punto giusto: 0.9,0.1 seed 1
1 punto che converge al punto sbagliato
efficienza convergenza