Mathematical Institute
Faculty of Mathematics and Physics
Albert-Ludwigs-University Freiburg

Quadratic Penalty Method

Numerical Optimization Project

in winter semester 2022/2023

by

Seval Özkurt and Simon Ebert

Abstract

Constrained optimization is important for a grand variety of applications. In the following report we present a native Matlab implementation for solving 2-dimensional constrained optimization problems with the Quadratic Penalty method. We rely on the algorithmic differentiation tools provided by CasADi and compare computation times of our algorithm to solution times of the IpOpt-solver. We show that our algorithm, although in general being slower than the IpOpt-solver has computation times in the same magnitude as the latter. In the end we give an outlook for possible extensions and adjustments for our algorithm. The code to our project can be found at `https://github.com/SEbert171/Optimization_Project`.

Contents

1 Introduction

Solving constrained optimization problems are of great interest in many different fields [6], for example in economics [3] or in machine learning domains [2]. While interior-point-solvers as IpOpt are widely used, there is also the so-called Quadratic Penalty method for solving constrained optimization problems. In the following, we will explain the underlying theory of this method (section 2) and give graphical representations of the working of the method. In a second part, we will present our native Matlab-implementation of the method, which can be used to solve 2-dimensional constrained optimization problems (section 3). Furthermore, in section 4, we will show theoretical and empirical results, concerning both the general Quadratic Penalty method and our concrete implementation. In the end, we will give a short outlook on possible extensions of our implementation.

2 Quadratic Penalty method

In the following we will present the theoretical approach of the Quadratic Penalty method. This approach can be used to solve a constrained optimization problem, i.e.

$$\begin{aligned} &\min_{x \in \mathbb{R}^n} f(x) \\ &\text{s.t. } \; g(x) = 0, \end{aligned} \tag{1}$$

with $f, g \in \mathbb{R}$. For simplicity we suppose that all constraints are equality constraints. The main idea of the method is quite simple: Instead of solving the constrained problem (1) we solve a series of unconstrained problems which consist of the objective function and a penalty function

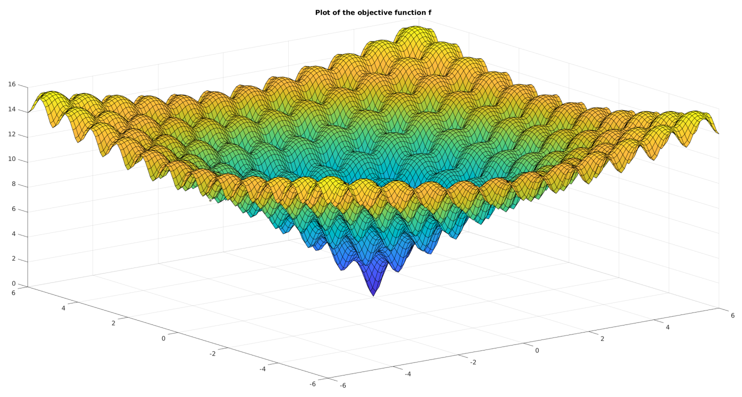


Figure 1: The graph of the Ackley function f, one of the functions used as test objective functions.

for violating the constraints. By increasing the magnitude of the penalty function we assure that the algorithm converges to a minimum of the objective function, where the constraint violation is small.

The unconstrained problems have the form

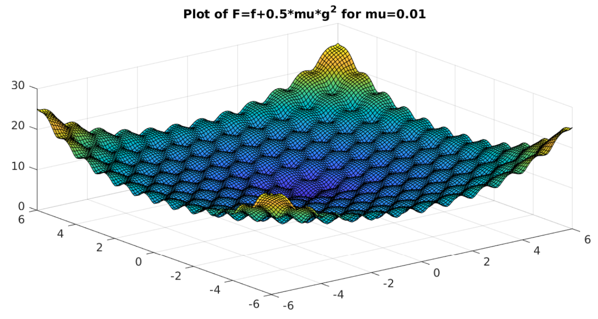
$$\min_{x\in\mathbb{R}^n} f(x) + Q_\mu(x) = f(x) + \frac{\mu}{2}\cdot g^2(x) =: F(x;\mu) \tag{2}$$

where $f:\mathbb{R}^n \to \mathbb{R}$ is the objective function from problem (1) and $Q_\mu:\mathbb{R}^n\to\mathbb{R}$ is the Quadratic Penalty function derived from the constraint function g. $\mu\in\mathbb{R}_+$ is the penalty parameter which can be used to control the influence of Q_μ. In order for the method to work, the penalty function Q_μ has to satisfy at least the following properties [6]:

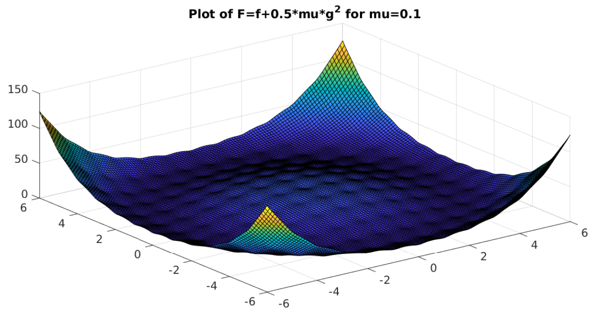
(i) $Q_\mu(x)\geq 0$ for all $x\in\mathbb{R}^n$

(ii) $Q_\mu(x) = 0 \quad \Leftrightarrow \quad g(x)=0$

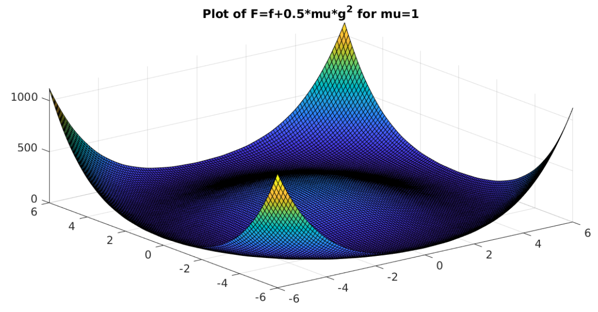
(iii) $Q_\mu \to \chi_{\{g=0\}}$ pointwise for $\mu\to\infty$



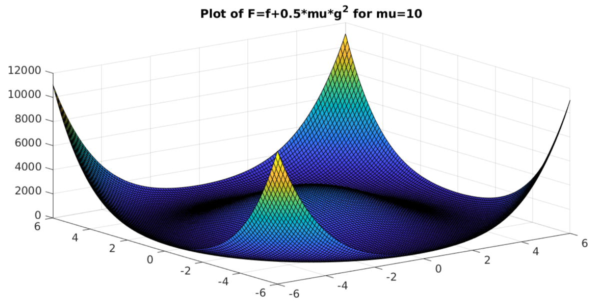
(a) The graph of the combined function $F = f + 0.5\mu\cdot g^2$ for $\mu = 0.01$



(b) The graph of the combined function $F = f + 0.5\mu\cdot g^2$ for $\mu = 0.1$



(c) The graph of the combined function $F = f + 0.5\mu\cdot g^2$ for $\mu = 1$



(d) The graph of the combined function $F = f + 0.5\mu\cdot g^2$ for $\mu = 10$

Figure 2: The graph of the objective function F of the unconstrained problem 2 for different values of μ with f being the Ackley function shown in Figure 1. One can see very well how the increasing values of μ lead to a smoothening of the function and a convergence towards the characteristic function $\chi_{\{g=0\}}$ but that the objective function f still plays a role in the exact shape of F. Note that the values for μ are very small in this example to ensure visibility of the changes. We use much higher values for μ in the actual calculations.

In this list the first two conditions ensure that constraint violations are punished (since we search a minimizer, the penalty has to be positive) and that the penalty is 0 when the constraints are not violated. The third condition secures that the augmentation of the penalty parameter in general leads to a higher penalty on constraint violations.

The Quadratic Penalty function $Q_\mu(x) = 0.5 \cdot \mu \cdot g^2$ additionally preserves the convexity of the constraint function g and is therefore a convenient and very popular choice of the penalty function, which explains the name „Quadratic Penalty method“. Given a starting point $x_0 \in \mathbb{R}^n$ and an initial penalty parameter $\mu_0 \in \mathbb{R}_+$ the unconstrained problem (2) can now be solved using an iterative method which yields a first solution x_0^*. This solution can be used to initialize the next unconstrained problem with a higher value of μ. Following the approach of solving multiple problems with increasing penalty parameter μ simplifies the finding of a minimum of the objective function because a high initial value of μ leads to a small influence of the objective function.

One can summarize the approach of the Quadratic Penalty method in the following algorithm.

Algorithm 1 (Quadratic Penalty method)**.**

```
Start with $\mu_0 > 0$, a nonnegative tolerance sequence $\{\tau_k\}$ with $\tau_k \to 0$ and a starting point $x_0$
for $k = 0, 1, 2, \ldots$
        Find an approximate minimizer $x_k$ for $F(x; \mu) = f(x) + 0.5 \cdot \mu \cdot |g(x)|^2$
        if $||\nabla_x F(x_k; \mu_k)|| \leq \tau_k$
                break
        end if
        Choose new penalty parameter $\mu_{k+1} > \mu_k$
        Set the new starting point as the obtained approximate solution
end for if maximum count of iterations is received
```

3 Implementation

In the following section we will describe the concrete implementation of the method. We used Matlab and the symbolic framework of CasADi [1].

3.1 Newton's method

We implemented the exact Newton's method to solve the unconstrained optimization problem 2.

Theorem 3.1 ([4])**.** *Let $F : \mathbb{R}^n \to \mathbb{R}$ a twice differentiable function with $\nabla^2 F$ is Lipschitz and $x^* \in \mathbb{R}^n$ a minimizer of f. Then there exists an $\varepsilon > 0$, so that for every $x_0 \in B_\varepsilon(x^*)$ the sequence recursively defined by*

$$x_{k+1} = x_k - \nabla^2 f(x_k)^{-1} \cdot \nabla f(x_k)$$

converges to the minimizer x^*. Here $\nabla^2 F(x_k)$ signifies the Hessian matrix of f at the point x_k and $\nabla f(x_k)$ is its gradient. The vector $-\nabla^2 F(x_k)^{-1} \cdot \nabla F(x_k) =: p_k$ is called the step direction.

If x^* is a local minimizer of $F : \mathbb{R}^n \to \mathbb{R}$, we know that $\nabla F(x^*) = 0$, which is equivalent to $||\nabla f(x^*)|| = 0$. Therefore we can give the Newton algorithm to find a minimizer of F:

Algorithm 2.

```
Start with $x_0 \in \mathbb{R}^n$ and a tolerance $\tau$. Set $k \leftarrow 0$
while $||\nabla F(x_k)|| > \tau$
        $p_k \leftarrow -\nabla^2 F(x_k)^{-1} \cdot \nabla F(x_k)$
        $x_{k+1} \leftarrow x_k + p_k$
        $k \leftarrow k + 1$
end while
$x^* \leftarrow x_k$
```

For computing the gradient and the Hessian of F we used CasADi's algorithmic differentiation tools. For globalization we also included line search with the Armijo-condition, which ensures

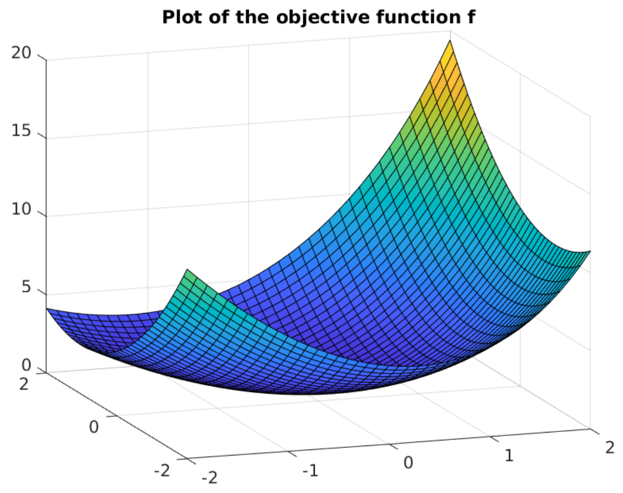


Figure 3: The graph of the function $f(X,Y) = X^2 + X \cdot Y + Y^2 + \exp X$, which is one of the test objective functions we used. One can see very well that it is convex and not symmetric in X and Y.

that the step length is small enough for a sufficient decrease of the Newton step [4]. The Armijo-condition accepts a Newton-step in direction p_k only if

$$F(x_{k+1}) \leq F(x_k) + \gamma \nabla f(x_k)^T \cdot p_k$$

where γ is a constant, which we set in our code to $\frac{1}{10}$. A full transcript of our approach to pseudo-code can be found in the Appendix.

3.2 Code structure

We implemented the penalty parameter augmentation step in the main function where one can define the basic settings like the maximum iterations for the penalty augmentation, the Newton step and the line search. Furthermore, one can determine if the algorithm uses our implementation of the Newton step (by setting `solve_method` to '`exact_Newton`') or if the unconstrained problem (2) is directly solved by IpOpt (by setting `solve_method` to '`Ipopt`'). Also, the type of objective function can be set here: We implemented four different possible test functions, the Ackley, Rosenbrock and Rastrigin function which are all quite challenging objective functions for optimization algorithms, as well as the convex function

$$\begin{array}{rccc} f: & \mathbb{R}^2 & \to & \mathbb{R} \\ & (X,Y) & \mapsto & X^2 + X \cdot Y + Y^2 + \exp X \end{array}$$

to see how our program handles easy problems and asymmetric functions. The graph of the Ackley function and the convex function can be seen in Figure 1 and Figure 3, respectively. The graphs of the other two functions (the Rosenbrock and the Rastrigin function) are displayed in Figure 6 in the Appendix.

The computation of every of the unconstrained problems is done in a separate file, depending if one wants to use our native Newton implementation or the via IpOpt. Additionally, because it has to be done very often the calculation of derivatives is also outsourced, as well as the plotting of results at the end. To check if the algorithm really converged to a minimum, we implemented a check of the second order sufficient condition, i.e. we check if the eigenvalues of the Hessian of the penalty function at a possible solution are positive without exception. For performance we included a backstop for the algorithm, which is used to stop the calculation if the algorithm converges away from the feasible set.

We also implemented the problem directly via the framework for constrained optimization of CasADi in the file `maini.m` to compare its computation times to the ones of our algorithm. Finally, for testing the convergence regions of our algorithm, we included two auxiliary files, called `main_test.m` and `test_convergence.m`, which can be used to call the optimization algorithm multiple times with different starting values. The results can be seen in subsection 4.2

4 Results

In this section we describe some convergence properties of the Quadratic Penalty method and show how our algorithm actually performs for different starting values. Furthermore, we show empirical findings on convergence of our algorithm towards exact solutions. First, we introduce a general convergence property.

4.1 Convergence analysis

For the following theorem we assume that the penalty function $F(x; \mu_k)$ has a finite minimizer for each value of μ_k.

Theorem 4.1. *[6] Suppose that each x_k is the exact global minimizer of $F(x; \mu_k)$ and that $\mu_k \to \infty$. Then every limit point x^* of the sequence $\{x_k\}$ is a global solution of the problem (1).*

The result in this theorem requires us to find the global minimizer for each subproblem but we have to take into consideration that the convergence to the global solution of (1) cannot be attained in general. In [5] N. M. Gould makes several assumptions to ensure global convergence. The main assumptions are:

- The iterates generated by the framework all lie within a bounded domain Ω.
- The sequence $\{\mu_k\}$ converges to infinity as k tends to infinity.
- Any limit point x^* of the sequence $\{x_k\}$ satisfies the second-order condition.
- The third derivatives of the functions $f(x)$ and $g_i(x)$ exist and are bounded for all points in Ω.

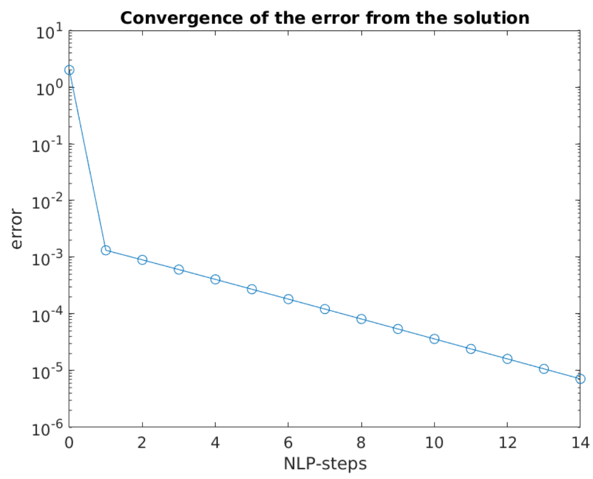
These assumptions are needed to ensure that x^* is a global minimizer. Because Newton-type algorithms can always be attracted to infeasible points we still have to consider how to ensure that x^* lies in the feasible region. Furthermore it is also possible that the function $F(x_k; \mu)$ is unbounded from below for any value of the penalty parameter. A possibility to ensure that the feasibility conditions are satisfied is the usage of the augmented Lagrangian method [6] or of the interior point penalty function [7] which set a barrier against leaving the feasible region. The latter is characterized by the property of preserving strict constraint feasibility at all times by using an interior point penalty term which is infinite on the constraint boundaries.

4.2 Empirical findings

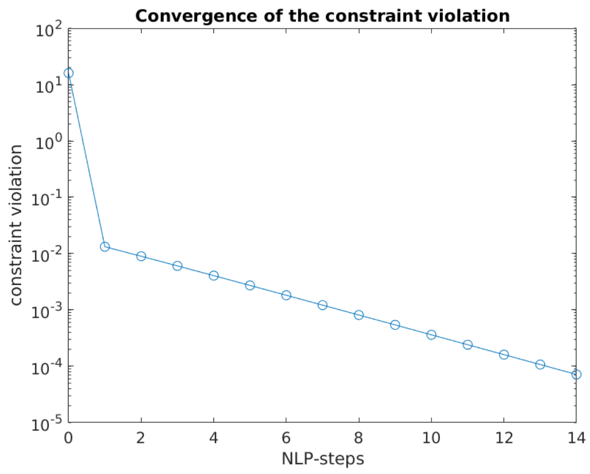
We ran our algorithm and the direct CasADi solver for different values of the maximum constraint violation. We chose the starting point $[x, y] = [2, 3]$ and Rastrigin as test function, so that we knew that the methods would converge. The results can be seen in Figure 7 in the Appendix. It shows that for large constraint violations our algorithm can keep up to the Ipopt-solver. However, for small constraint violations, the computation time of our implementation rises approximately linearly, whereas Ipopt produces nearly constant computation times.

For adequate starting points, our algorithm converges towards a minimum, which satisfy the constraint. We choose $[x, y] = [-3, 0.1]$ as a starting point for minimizing the Ackley function

(displayed in Figure 1). A locally optimal solution for our constrained optimization problem is for example $[x^*, y^*] = [-5, 0]$. In Figure 4 one can see the linear convergence both of the error and the constraint violation in this case.
Finally, we conducted an analysis on the possible starting points. Therefore, we let run the algorithm for multiple starting points in the square $[-5, 5]^2$ and checked for each point if the algorithm converged to a solution. For the Rosenbrock function, this analysis can be seen in Figure 5.



(a) Convergence of the error from the exact solution of the algorithm.



(b) Convergence of the constraint violation of the algorithm.

Figure 4: The convergence of constraint violation and error for the implemented Quadratic Penalty method. In this example we used the Ackley function with starting point $[x, y] = [-3, 0.1]$, which converges to the exact solution $[x^*, y^*] = [-5, 0]$. The penalty parameter μ was set to 10 in the beginning and multiplied by 10 in every NLP-step.

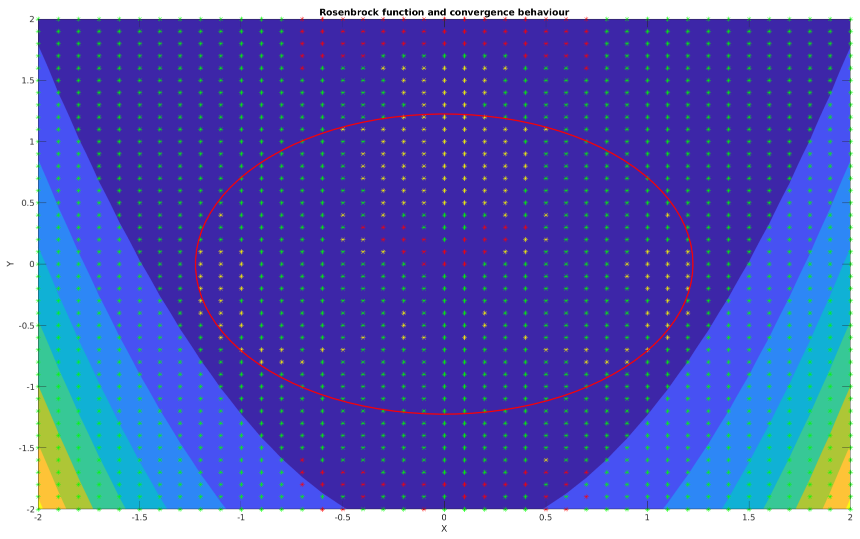


Figure 5: The convergence behaviour of our algorithm for different starting values. Green points signify, that the method converges toward the constraint and that the second order sufficient condition is satisfied at the approximate solution, meaning that it converged to a minimum on the feasible set. Yellow point signify a convergence toward the feasible set, but to a point, where the second order sufficient condition is not satisfied, i.e. not a local minimum. Red point mean, that the algorithm did not converge to the feasible set.

References

[1] Joel A E Andersson et al. “CasADi – A software framework for nonlinear optimization and optimal control”. In: *Mathematical Programming Computation* 11.1 (2019), pp. 1–36. DOI: 10.1007/s12532-018-0139-4.

[2] Sohail Bahmani, Bhiksha Raj, and Petros T Boufounos. “Greedy sparsity-constrained optimization”. In: *The Journal of Machine Learning Research* 14.1 (2013), pp. 807–841.

[3] Michael Carter. *Foundations of mathematical economics.* MIT press, 2001.

[4] Moritz Diehl. *Numerical Optimization. Lecture Notes.* Albert-Ludwigs-Universität Freiburg, 2020.

[5] Nicholas Ian Mark Gould. “On the Convergence of a Sequential Penalty Function Method for Constrained Minimization”. In: *SIAM Journal on Numerical Analysis* 26.1 (1989), pp. 107–128.

[6] Jorge Nocedal and Stephen J. Wright. *Numerical Optimization.* Ed. by Thomas V. Mikosch, Sidney I. Resnick, and Stephen M. Robinson. Springer, 2006. ISBN: 978-0387-30303-1.

[7] W. Sun and Y. Yuan. *Optimization Theory and Methods. Nonlinear Programming.* Springer, 2006.

Appendix

The full algorithm for the Quadratic Penalty method with Newton-step and Armijo linesearch is given here:

Algorithm 3.

```
Start with $\mu_1 > 0$, a nonnegative sequence $\{\tau_k\}$ with $\tau_k \to 0$, a starting point $x_1$ and a minimal step length $t_{min}$
$F(x;\mu) \leftarrow f(x) + 0.5 \cdot \mu_1 \cdot g(x)^2$
$k \leftarrow 1$
while $||\nabla F(x_k)|| > \tau_k$                                    Quadratic Penalty step
    $j \leftarrow 1$
    $y_j <- x_k$                                                initialization of Newton step
    while $\left\|\nabla F(y_j)\right\| > \tau$                                    Newton step
        $p_k \leftarrow -\nabla^2 F(y_j)^{-1} \cdot \nabla F(y_j)$
        $t \leftarrow 1$                                                    Armijo linesearch
        while $F(y_j + t \cdot p_k) > f(y_j) + 0.1 \nabla f(y_j)^T \cdot p_k$   &   $t > t_{min}$
            $t \leftarrow t \cdot 0.8$
        end while
        $y_j \leftarrow y_j + t \cdot p_k$
        $j \leftarrow j + 1$
    end while
    $x_{k+1} \leftarrow y_j$
    Choose new penalty parameter $\mu_{k+1} > \mu_k$
    $F(x;\mu) = f + 0.5 \cdot \mu_{k+1} \cdot g^2$                          renew penalty function
    $k \leftarrow k + 1$
end while
```

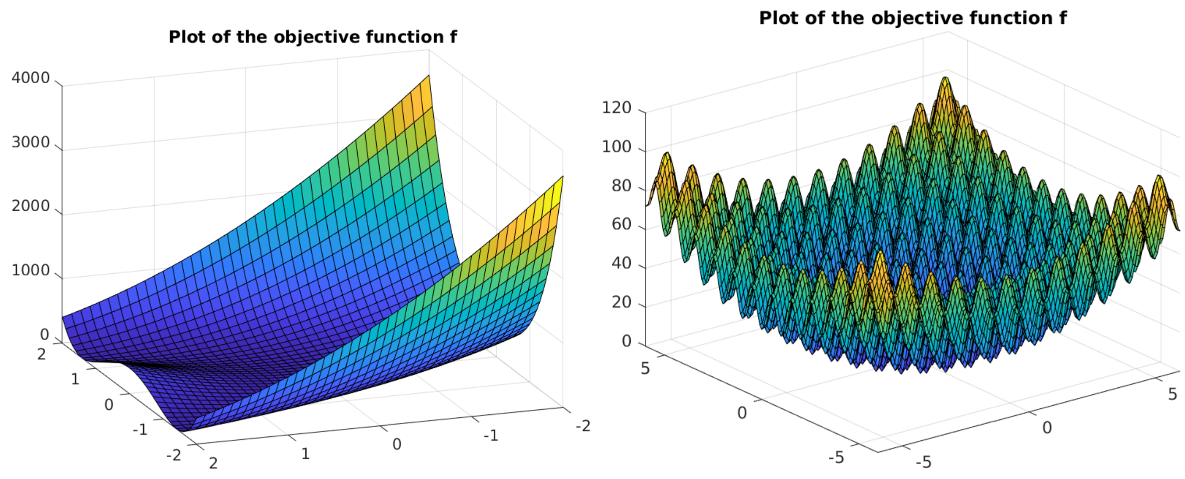


Figure 6: The graph of the Rosenbrock function (left) and the Rastrigin function (right).

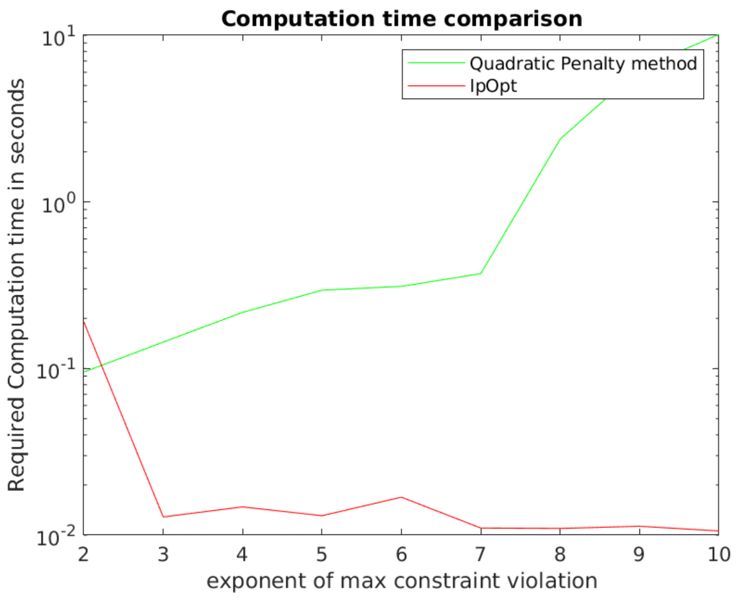


Figure 7: Computation times of our implementation and direct minimizing using the CasADi framework for constrained optimization. On the axis the exponent of the maximal constraint violation is displayed meaning that for $x = 2$, the graph shows the computation time of the algorithms, when the maximal constraint violation is set to 10^{-2}. The starting point for this computation is $[x, y] = [2, 3]$ and the objective function is the Rastrigin function.