Interior Point Methods applied to Quadratic Programming*

Daniel Bergmann [1]

Abstract— **Interior-point-methods allow to treat convex optimization problems that include inequality constraints. Via a barrier function, approximated inequality constraints are transferred to the objective function implicitely, while conclusions can be drawn about the optimal value of the original problem, by solving the approximated one. Two interior-point algorithms are presented. For one of them, we give an implementation in MATLAB.**

I. INTRODUCTION

Subject of this paper is the concept of interior point methods in convex optimization to solve problems including inequality constraints. Therefore we introduce the concept of barrier functions. With that we can approximate the problem without any inequality constraints. We begin with a brief introduction of Newton's method, with which equality constrained convex problems can be solved effectively. Then the barrier concept in general is explainedm, including an argument to show a bound on suboptimality, that a optimal point of the approximative problem has regarding to the original one. Further, we proceed with two algorithms using barrier funcitons: Firstly, the barrier method with full newton search, secondly the Primal-Dual Method, which provides better performance for certain problem classes. After we gave an idea, how to find a feasible initial point, we conclude the paper by applying an MATLAB-Implementation of the Primal-Dual Method on two examples.

Notation. Throughout this paper, we denote the set of all real numbers as $\mathbb{R}$ and the set of all natural numbers as $\mathbb{N}$, respectively $\mathbb{N}_0$ if zero is included. Further we denote the Jacobian of a function $f : \mathbb{R}^n \longrightarrow \mathbb{R}^m$, differentiated along direction v, evaluated at point $x \in \mathbb{R}^n$ as $J_v f(x)$. The index v is omitted, if it is contextually clear. For a collection of scalar valued functions $h_i : \mathbb{R}^n \longrightarrow \mathbb{R}, i = 1, \ldots, m$ with the same domain, we define

$$h(x) := \begin{pmatrix} h_1(x) \\ \vdots \\ h_m(x) \end{pmatrix}$$

for the stacked values of all h_i evaluated at $x \in \mathbb{R}^n$. The sign $\mathbb{1}$ denotes a column vector filled with ones in appropriate dimensions.

II. PROBLEM STATEMENT AND BACKGROUND

For theoretical discussions, we consider the convex constrained optimization problem

$$
\begin{aligned}
\underset{x}{\text{minimize}} \quad & f_0(x) \\
\text{subject to} \quad & f_i(x) \leq 0, \ i = 1, \ldots, m. \\
& A_{\text{eq}} x = b_{\text{eq}}.
\end{aligned}
\tag{1}
$$

with $f_0 : \mathbb{R}^n \longrightarrow \mathbb{R}$ convex and twice differentiable, $f_i : \mathbb{R}^n \longrightarrow \mathbb{R}$ for $i = 1, \ldots, m$ convex and differentiable, $A_{\text{eq}} \in \mathbb{R}^{n \times p}, b_{\text{eq}} \in \mathbb{R}^p$ with equality and inequality constraints. For such an optimization problem, we call its Lagrangian $L : \mathbb{R}^n \times \mathbb{R}^m \times \mathbb{R}^p \longrightarrow \mathbb{R}$ with

$$L(x, \lambda, \nu) = f_0(x) + \lambda^T f(x) + \nu^T (A_{\text{eq}} x - b_{\text{eq}}).$$

Further, we denote its dual problem by

$$
\begin{aligned}
\underset{(\lambda, \nu)}{\text{maximize}} \quad & g(\lambda, \nu) \\
\text{subject to} \quad & \lambda \geq 0, \nu \in \mathbb{R}^p
\end{aligned}
\tag{2}
$$

with

$$g(\lambda, \nu) = \inf_{x \in \mathbb{R}^n} L(x, \lambda, \nu).$$

Moreover, we give a MATLAB-implementation of a primal-dual interiorpoint method for convex quadratic optimization problems. Quadratic problems are a subclass of (1) and denote as

$$
\begin{aligned}
\underset{x}{\text{minimize}} \quad & f_0(x) = \frac{1}{2} x^T Q x + c^T x \\
\text{subject to} \quad & A_{\text{in}} x - b_{\text{in}} \leq 0, \ A_{\text{in}} \in \mathbb{R}^{m \times n}, b_{\text{in}} \in \mathbb{R}^m \\
& A_{\text{eq}} x - b_{\text{eq}} = 0, \ A_{\text{in}} \in \mathbb{R}^{p \times n}, b_{\text{in}} \in \mathbb{R}^p
\end{aligned}
\tag{3}
$$

with matrices $0 \prec Q \in \mathbb{R}^{n \times n}, c \in \mathbb{R}^n$. Quadratic programming has multiple applications in various fields, such as artificial intelligence or control. A way to apply it for example on a zero terminal Model Predictive Control is shown in the appendix.

III. MAIN RESULTS

Convex problems without equality constraints can be solved using newton's method. We at first explain the idea of Newton's method shortly, since basically it is the algorithm we want to apply. The barrier method, presented afterwards, is a way to use Newton's method also for inequality-constrained problems.

A. Newton's Method

Newton's method is an iterative process to solve nonlinear equation systems

$$F(x) = 0 \qquad (4)$$

for a differentiable map $F : \mathbb{R}^n \longrightarrow \mathbb{R}^m$. The algorithm works as follows: At a given point x_k, the zero of the linear approximation of F around x_k is computed. This point is chosen as the next iterate x_{k+1}. In particular, a linear approximation of F in x_k is defined as

$$L(x) := F(x_k) + JF(x_k)(x - x_k) \text{ for } x \in \mathbb{R}^n, \qquad (5)$$

where $JF(x_k)$ is the Jacobian of F at the point x_k. If $JF(x_k)$ is invertible, the point $\tilde{x}$ with $L(\tilde{x}) = 0$ is exactly the solution of the linear equation $JF(x_k)x = -F(x)$. Technical conditions and proofs about convergence rates of Newton's method can be found in [1]. The procedure executing a Newton search is summarized in (1).

Algorithm 1: Newton's Method

Result: $\tilde{x}$, approximate solution of nonlinear system of equalities $F(x) = 0$, residual tolerance $\epsilon_{res} > 0$, cauchy-tolerance $\epsilon_c > 0$

Data: Function $F : \mathbb{R}^n \longrightarrow \mathbb{R}^n$, initial point x_0

while $\|x - x_{\text{last}}\| \geq \epsilon_c$ *or* $\|F(x)\| \geq \epsilon_{res}$ **do**
  compute Newton direction Δx by solving
  $JF(x)\Delta x = -F(x)$;
  remember last interation for checking term. crit.
  $x_{\text{last}} = x$;
  update current point by $x = x + \Delta x$;
end
return $\tilde{x} = x$;

Remark 1: The residual and the cauchy-criterion for termination should be combined for the newton method. Easy examples are known, where one of the criteria is satisfied even though the current iteration is far from the optimal point. For details, see [1]. For theoretical reasoning, or if $\nabla^2 f(x)^{-1}$ can be used explicitly, one can also use the decreasement of $\lambda^2 = \nabla f(x)^T \nabla^2 f(x)^{-1} \nabla f(x)$ (so called newton-decrement) under a certain tolerance.

For the purpose of optimizing a convex, twice differentiable objective function f_0 we want to find a zero of the gradient ∇f_0. Therefore we can apply the Newton Method to solve the non-linear equation

$$F(x) := \begin{pmatrix} \nabla f_0(x) \\ g(x) \end{pmatrix} = 0.$$

By convexity, satisfying $\nabla f_0(x^*) = 0$ is not only neccessary, but also sufficient for x^* to be a global minimum of f_0.

B. Concept of Barrier Methods

Convex optimization problems with no inequality constraints can be solved efficiently by using Newton's method. If inequality constraints are involved, Newton's method can not guarantee feasibility of a found solution. Hence it is desirable, to transform an inequality-constrained optimization problem into one, that is only equality-constrained. Therefore, we move the inequality constraints implicitly to the objective function. A simple and precise way to do this, would be to evaluate an indicator function

$$I_-(x) := \begin{cases} 0 & \text{for } u \neq 0 \\ \infty & \text{for } u > 0 \end{cases} \qquad (6)$$

on the values of the inequality constraints $f_i, i = 1, \ldots, m$. We obtain a problem in the following shape

$$\begin{aligned} \underset{x}{\text{minimize}} \quad & f_0(x) + \sum_{i=1}^{m} I_-(f_i(x)) \\ \text{subject to} \quad & A_{\text{eq}}x - b_{\text{eq}} = 0. \end{aligned} \qquad (7)$$

This problem is equivalent to (1), since it yields an objective value of $+\infty$ for every infeasible point while it is the same problem for every feasible one. We now have a formulation without inequality constraints. However, it is clearly neither convex nor continuous and hence not differentiable. Since we need these properties to solve the optimization problem computationally, we approximate the indicator function I_- by the function

$$\hat{I}_-(u) = \begin{cases} \frac{1}{t}\log(-u) & \text{for } u < 0, \\ \infty & \text{for } u \geq 0, \end{cases} \qquad (8)$$

The parameter $t > 0$ sets the approximation's accuracy. A higher value for t results in a better approximation of the indicator function. By replacing the indicator function by $\hat{I}_-$, we obtain an approximation

$$\begin{aligned} \underset{x}{\text{minimize}} \quad & f_0(x) - \sum_{i=1}^{m} \frac{1}{t}\log(-f_i(x)) \\ \text{subject to} \quad & A_{\text{eq}}x - b_{\text{eq}} = 0 \end{aligned} \qquad (9)$$

of problem (1). Throughout this paper, we denote its Lagrangian by $L_t : \mathbb{R}^n \times \mathbb{R}^p \longrightarrow \mathbb{R}$.

Note that $\frac{1}{t}\log(-u)$ is convex, increasing in u, and differentiable on the feasible set. Hence the entire function $\sum_{i=1}^{m} \hat{I}_-(f_i(x))$ is convex and (9) is a convex Problem with differentiable objective function. These properties allow us to solve (9) computationally. We call an optimal point $x^*(t)$ of (9) with parameter t a central point and a solution to its dual problem $(\lambda^*(t), \nu^*(t))$ a dual central point. The set of (dual) solutions of (9) for all $t > 0$ we call the (dual) central path. Since for points x with $f_i(x) = 0$ for any $x \in \{1, \ldots, m\}$, the objective of (9) is ∞, all central points are in the interior of the set, satisfying the inequality constraints of (1). Thus this framework is named interior point method. One can show, that solutions $(x^*(t), \lambda^*(t), \nu^*(t))$ of (9) converge to the solution (x^*, λ^*, ν^*) of (1) for $t \longrightarrow 0$. The proof can be found in [2].

C. Measure for the Approximation's quality

An immediately arising question is which conclusions about the solution (x^*, λ^*, ν^*) of (1) can be drawn from

knowing a solution of (9) for a certain $t > 0$ about the value $f_0(x^*(t))$ of a central point $x^*(t)$, compared with the optimal value p^* of the original problem. For compactness, we denote the barrier term of the approximated problem as

$$\phi(x) = -\sum_{i=1}^{m} \log(-f_i(x)),$$

with its Jacobian and Hessian being

$$\nabla\phi(x) = \sum_{i=1}^{m} \frac{1}{-f_i(x)}\nabla f_i(x),$$

$$\nabla^2\phi(x) = \sum_{i=1}^{m} \frac{1}{f_i(x)^2}\nabla f_i(x)\nabla f_i(x)^T + \sum_{i=1}^{m} \frac{1}{-f_i(x)}\nabla^2 f_i(x).$$

For the sake of simplifying notation, throughout this section we consider the problem

$$\begin{aligned}
\underset{x}{\text{minimize}} \quad & tf_0(x) + \phi(x) \\
\text{subject to} \quad & A_{\text{eq}}x = b_{\text{eq}}.
\end{aligned} \tag{10}$$

that is obtained by multiplying the objective in (9) with $t > 0$. The original and the obtained problem are equivalent. Any arbitrary $x^*(t)$ a strictly feasible point of (1). Since $x^*(t)$ solves (10), there exists $\hat{\nu} \in \mathbb{R}^p$, such that

$$\nabla L_t(x^*(t), \hat{\nu}) = t\nabla f_0(x^*(t)) + \nabla\phi(x^*(t)) + A_{\text{eq}}^T\hat{\nu} \tag{11}$$

$$= t\nabla f_0(x^*(t))$$
$$+ \sum_{i=1}^{m} \frac{1}{-f_i(x^*(t))}\nabla f_i(x^*(t)) + A_{\text{eq}}^T\hat{\nu}. \tag{12}$$

$$= 0 \tag{13}$$

holds. Note that the Lagrangian only depends on $(x, \hat{\nu})$, since there are no explicit inequality constraints involved. We keep in mind, that $x^*(t)$ minimizes (9). Using this insight, we know that there exists a dual feasible point $(x^*(t), \lambda^*(t), \nu^*(t))$ of the original problem (1). In particular, we choose

$$\lambda^*(t) = -\frac{1}{tf_i(x^*(t))} \text{ for } i = 1, \ldots, m, \quad \nu^*(t) = \frac{\hat{\nu}}{t}.$$

Here, $\lambda^*(t) > 0$ follows from $f_i(x^*) < 0$ for all $i = 1, \ldots, m$ since x^* is strictly feasible.

Note that (13) is the derivative of the Lagrangian

$$L(x, \lambda, \nu) = f_0(x) + \sum_{i=1}^{m} \lambda_i^*(t)f_i(x) + \nu^*(t)^T(A_{\text{eq}}x^*(t) - b_{\text{eq}})$$

dividied by $t > 0$ of the original problem. The Lagrangian is convex in the first coordinate, hence we infer that $x^*(t)$ minimizes the Lagrangian of the original problem for any fixed (λ, ν). For the dual function of the original problem,

we obtain

$$\begin{aligned}
g(\lambda^*(t), \nu^*(t)) &= f_0(x^*(t)) + \sum_{i=1}^{m} \lambda_i^*(t)f_i(x^*(t))) \\
&\quad + \nu^*(t)^T(A_{\text{eq}}x^*(t) - b_{\text{eq}}) \\
&= f_0(x^*(t)) - \frac{m}{t}.
\end{aligned} \tag{14}$$

The second of the three summands adds up to $m \cdot 1$, because of the particular choice of $\lambda^*(t)$, fractions cancel out. The last summand equals zero, since $A_{\text{eq}}x^*(t) - b_{\text{eq}} = 0$.

By weak duality, this means that the optimum $x^*(t)$ of the approximated problem (9) has an objective value $f_0(x^*(t))$ that is maximally larger by $\frac{m}{t}$ (and hence worse) than the real optimal value p^* of the original problem. Thus, one can theoretically force a desired bound on the subobtimality $\epsilon > 0$ by choosing t large enough, in particular $t := \frac{m}{\epsilon}$. However, just solving (9) with a large choice of t does not work out in general, since numerical issues can make convergence of Newton's Method dependent on the choice of the initial point x_0.

D. Algorithmic Use of the Barrier Concept

As already mentioned in section III-C, one can not solve (9) generally without a good guess of the initial value x_0. So how to make use of the barrier concept? The idea of interior methods is, to find points along the problem's central path. Two methods are introduced in the following. Emphasis of the explanations as well as the implementation in MATLAB will be on the Primal-Dual Interior Point Method.

1) Barrier Method: As mentioned before, for large t a good initial point x_0, meaning an initial point that is not far away from the actual minimum of (1), is crucial for avoiding large numerical errors. This can be achieved by starting with optimization of (9) for small $t = t_1$, which leads to a rather bad approximation of the original problem, but also to better numerical behavior. After finding $x^*(t_1)$ via Newton's method, t is increased to $t = t_2 > t_1$ by a certain rate and (9) is solved again with parameter $t = t_2$, with choice $x_0 = x^*(t_1)$ for the initial point.

We call finding the minimum $x^*(t)$ of (9) the centering step or outer iteration, while we call a single Newton step inside the centering step an inner iteration.

E. Complexity Analysis for the Barrier Method

For the algorithm presented above, bounds on its time complexity have been shown. We here discuss the time complexity of the barrier method, meaning the total number of newton steps needed to solve (1). An upper bound of these iterations can be proven for problems with objectives that are self-concordant. While linear and quadratic functions satisfy selfconcordance in general, any other convex optimization problem can be rewritten as a self-concordant one, so this condition is not very restrictive. The upper bound

$$\frac{f(x) - p^*}{\gamma} + c \tag{15}$$

on the maximal number of newton iterations that is needed to get a newton decrement (see remark 1) smaller than ϵ_{nt},

Algorithm 2: Barrier Method with full Newton search

Result: $x^*(t)$, approximate solution of (1) with
$$f_0(x^*(t)) - p^* < \frac{m}{t}$$
initialization: Matrices $0 \prec Q \in \mathbb{R}^{n \times n}, c \in \mathbb{R}^n$.
defining the objective function, matrices
$A_{\text{eq}} \in \mathbb{R}^{m \times n} b_{\text{eq}} \in \mathbb{R}^p, A_{\text{in}} \mathbb{R}^{m \times n}, b_{\text{in}} \in \mathbb{R}^m$ defining
constraints, initial point x, initial approximation
parameter $t > 0$, rate for increasing apppox. param.
$\mu > 1$ tolerance ϵ;

while $\frac{m}{t} \geq \epsilon$ **do**

   Compute $x^*(t)$ by solving (9) via Newton's
   Method, starting at x;
   Update $x := x^*(t)$;
   Increase t by $t := \mu t$

end

---

where c depends on ϵ_{nt} by $\log_2 \log_2(1/\epsilon_{\text{nt}})$, p^* is the primal problem's optimal value and γ depends on choice of the backtracking parameters α, β with

$$\frac{1}{\gamma} = \frac{20 - 8\alpha}{\alpha\beta(1 - 2\alpha)^2}.$$

The derivation of this bound is shown in [2], section 9.

One can further show that this bound holds uniformly for any parameter t for all problems (9). Since there are exactly

$$\left\lceil \frac{\log(m/\epsilon t_0)}{\log \mu} \right\rceil$$

outer steps neccessary to solve (9) with inital parameter $t = t_0$ and tolerance ϵ, the entire barrier method needs maximally

$$N = \left\lceil \frac{\log(m/\epsilon t_0)}{\log \mu} \right\rceil \left(\frac{m(\mu - 1 - \log \mu)}{\gamma} + c \right)$$

inner newton iterations, where m denotes the number of inequality constraints on (9). to yield a result with a sub-optimality of ϵ or smaller. Detailed reasoning can be found in [2], section 11.5.

1) Primal-Dual Interior Point Method: Like the previously introduced algorithm, the Primal-Dual Interior Point method uses the barrier concept to handle inequality constraints. It is motivated by the following idea: Since the points generated by each outer iteration converge to the desired optimum on the central path, one does not gain much advantage by computing the central points with a high level of accuracy. This results in many newton-steps being computed, without improving the convergence towards the optimum value of (1). Hence, it would be useful to reduce the accuracy of each outer iteration as much as possible, without losing convergence to the optimum. Therefore, in this method only one newton step will be computed for each parameter t in the approximated problem (9). Furthermore, this Newton step is computed differently. While in the barrier method with full newton search, the search directions are computed only considering the primal problem, the Primal-Dual Methodalso takes the dual

problem of (9) into account. In particular Newton's method is applied to a system of residual terms, which all have to equal zero by the modified KKT-conditons, here presented like in [2].

Theorem 1 (Modified KKT-Conditions): For a convex Optimization Problem with a logarithmic barrier function (9), the following conditions on a primal-dual point $(x, \lambda, \nu) \in \mathbb{R}^n \times \mathbb{R}^m \times \mathbb{R}^p$ are neccessary and sufficient for x being a solution to the primal problem and (λ, ν) being a solution to the dual problem:

$$f_i(x) \leq 0, \quad \text{for } i = 1, \ldots, m \tag{16a}$$
$$A_{\text{eq}}x - b_{\text{eq}} = 0 \tag{16b}$$
$$\lambda_i \geq 0, \quad \text{for } i = 1, \ldots, m \tag{16c}$$
$$-\lambda_i f_i(x) = \frac{1}{t}, \quad \text{for } i = 0, \ldots, m \tag{16d}$$
$$\nabla f_0(x) + \sum_{i=1}^m \lambda_i \nabla f_i(x) + \sum_{i=1}^p \nu_i \nabla h_i(x) = 0. \tag{16e}$$

Stacked in one vector, this yields the system of equalities

$$r_\mu(x, \lambda, \nu) = \begin{pmatrix} r_{\text{dual}} \\ r_{\text{cent}} \\ r_{\text{pri}} \end{pmatrix}$$
$$= \begin{pmatrix} \nabla f_0(x) + Jf(x)^T \lambda + A_{\text{eq}}^T \nu \\ -\text{diag}(\lambda)f(x) - \mu\mathbb{1} \\ A_{\text{eq}}x - b_{\text{eq}} \end{pmatrix} \overset{!}{=} 0. \tag{17}$$

to apply Newton's method on. For formulation of the linear Newton equality, we also compute the jacobian

$$J_{(x,\lambda,\nu)} r_\mu(x, \lambda, \nu) \tag{18}$$
$$= \underbrace{\begin{pmatrix} \nabla^2 f_0(x) + \sum_{i=1}^m \lambda_i \nabla^2 f_i(x) & Jf(x) & A_{\text{eq}}^T \\ -\text{diag}(\lambda)Jf(x) & -\text{diag}(f(x)) & 0 \\ A_{\text{eq}} & 0 & 0 \end{pmatrix}}_{:=M_{\text{KKT}}}$$
$$\tag{19}$$

of the residual and refer to it as M_{KKT}. Consequently, the Newton equality for finding the search direction $(\Delta x, \Delta \lambda, \Delta \nu)$ in each newton step is obtained by solving the linear equation

$$M_{\text{KKT}} \begin{pmatrix} \Delta x \\ \Delta \lambda \\ \Delta \nu \end{pmatrix} = b_{\text{KKT}} \tag{20}$$

with $b_{\text{KKT}} = -r_\mu(x, \lambda, \nu)$.

Unfortunately, adding the obtained step direction $(\Delta x, \Delta \lambda, \Delta \nu)$ to (x, λ, ν), does not in general yield a feasible point. Therefore we compute a suitable step-size s^* via a backtracking-linesearch, such that a certain decrease of the residual and feasibility is guaranteed for the next iteration point

$$\begin{pmatrix} x^+ \\ \lambda^+ \\ \nu^+ \end{pmatrix} = \begin{pmatrix} x \\ \lambda \\ \nu \end{pmatrix} + s^* \begin{pmatrix} \Delta x \\ \Delta \lambda \\ \Delta \nu \end{pmatrix}.$$

Algorithm 3: Backtracking linesearch

Result: Stepsize s^*, s.t. $\lambda^+ > 0, f(x^+) < 0$ and r_μ decreases by certain amount.

Data: Problem matrices, current x, λ, ν, Newton direction $\Delta x, \Delta \lambda, \Delta \nu$, barrier parameter μ, backtracking parameters $\alpha \geq 0, \beta \in (0, 1)$. Initial step-size set
$$s_{\max} = \min\{1, \min_{i|\Delta\lambda_i<0} -\lambda_i/\Delta\lambda_i\}$$

compute $r_\mu(x, \lambda, \nu)$;
$s = s_{\max}$;
$found = false$;
while $found == false$ **do**
   set $s = \beta s$;
   compute (x^+, λ^+, ν^+);
   compute $r_\mu(x^+, \lambda^+, \nu^+)$ and $f(x^+)$;
   if $f(x^+) < 0$ *and*
   $\|r_\mu(x^+, \lambda^+, \nu^+)\| \leq (1 - \alpha s) \|r_\mu(x, \lambda, \nu)\|$ **then**
      $found = true$
   end
end

Algorithm 4: Primal-Dual Interior Point Method

Result: approximate optimizer $\hat{x}^*$, approx. opt. value $\hat{p}^*$, approx. dual optimizer $(\hat{\lambda}^*, \hat{\nu}^*)$, surrogate duality gap $\hat{\eta}^*$ as measure of optimality

Data: Problem matrices, primal-dual initial point (x, λ, ν) with $f_i(x) < 0$ for all $i = 1, \ldots, m$, $\lambda > 0, \nu \in \mathbb{R}^p$ (initial point strictly feasibile), reduction factor $\gamma \in (0, 1)$, tolerances $\epsilon_{\text{feas}}, \epsilon_{\text{opt}} > 0$

Initialization;
determine problem dimensions n, m, p;
set $found = false$;
while $found == false$ **do**
   compute surrogate duality gap: $\hat{\eta} = -f(x)^T \lambda$;
   compute KKT residual vector $r_\mu(x, \lambda, \nu)$ via (17);
   compute search direction $(\Delta x, \Delta \lambda, \Delta \nu)$ by solving (20);
   determine suitable step size s via backtracking algorithm 3;
   update current primal and dual points:
   $(x, \lambda, \nu) = (x, \lambda, \nu) + (\Delta x, \Delta \lambda, \Delta \nu)$;
end
return $\hat{x}^* = x, \hat{p}^* = f_0(\hat{x}^*), \hat{\lambda}^* = \lambda, \hat{\nu}^* = \nu, \hat{\eta}^* = \hat{\eta}$;

The detailed procedure of the backtracking linesearch is displayed in Algorithm 3.

The iterates we obtain by Primal-Dual Methodcan not be guaranteed to satisfy the equality constraints. Hence suboptimality can not be measured via the duality gap. We therefore use the surrogate dualiy gap

$$\hat{\eta}(x, \lambda) = -f(x)^T \lambda.$$

Finally, we can present the entire algortithm of the Primal-Dual Method.

Remark 2: If a strictly feasible primal variable $x \in \mathbb{R}^n$ is known, $\lambda = -1/f_i(x) \geq 0, \nu = 0$ is always a valid choice for the initial dual variables.

Especially when results with high accuracy are needed, the Primal-Dual Method allows to omit a lot of newton steps, that would be have computed in the Barrier Method. For such problems, the Primal-Dual Method provides better performance.

Remark 3: The concept of using a barrier method to approximate inequality constraints can also be used for solving optimization problems involving generalized inequalities. Therefore, the barrier function and constraints on the dual problem have to be adjusted. Doing this, the class of problems that apply the usage of barrier methods is widely enlarged. For example it can be used for solving linear matrix inequalities or single order cone problems.

Further, a higher speed of convergence can be shown for the application on some special classes of problems, such as quadratic problems or single order cone problems (SOCPs). Here the Primal-Dual Methodcan perform faster than with linear convergence.

F. How to find a feasible inital point

The Algorithms 2 and 4 both need a strictly feasible initial point to start. Since such a point is in general not trivial to find, one can formulate the search for the initial point as another convex optimization problem, that is easier to solve than the original one. For problem (1), one way to implement this, is solving

$$\begin{aligned} \underset{x}{\text{minimize}} \quad & s \\ \text{subject to} \quad & f_i(x) \leq s, \quad i = 1, \ldots, m \\ & A_{\text{eq}} x - b_{\text{eq}} = 0, \end{aligned} \quad (21)$$

via Newton's method. If a point with optimal value strictly smaller than zero for (21) is found, then this point is strictly feasible. Solving such a first, more simple problem is called a Phase I problem. More examples of such problems can be found in [2].

IV. EXAMPLES

We give an implementation of the Primal-Dual Method (Algorithm 4) in MATLAB, along with two example problems. One with equality constraints, one without.

A. Numerical Examples

1) Example 1: The effectivity of the algorithm can be demonstrated with the following explicit example of (1). For

the problem defined by

$$Q = 2 \cdot \begin{pmatrix} 1 & 0 \\ 0 & 2 \end{pmatrix}; \quad c = \begin{pmatrix} 1 \\ 2 \end{pmatrix}$$
$$A_{eq} = \begin{pmatrix} 1 & -1 \end{pmatrix}; \quad b_{eq} = 0;$$
$$A_{in} = \begin{pmatrix} 0 & 1 \end{pmatrix}; \quad b_{eq} = 10,$$

we choose backtracking parameters $\alpha 0.05, \beta = 0.5$ as well as $\gamma = 0.1$ for the reduction of the approximation parameter. Initial points are set to $x_0 = (1,1)^T, \lambda_0 = 1, \nu_0 = 0$ and tolerances to $\epsilon_{feas} = \epsilon_{opt} = 10^{-4}$.

With this setup, the solver terminates after 20 iterations and yields the following results:

$$x^* = \begin{pmatrix} -1.0000 \\ -1.0000 \end{pmatrix};$$
$$\lambda^* = 5.6650 \cdot 10^{-6};$$
$$\nu = 2.7562 \cdot 10^{-6};$$
$$f_0(x^*) = -1.5000;$$
$$\hat{\eta} = 6.2315 \cdot 10^{-6}.$$

2) Example 2 : To test algorithm (4) on a problem without equality constraints, we moreover provide a second example, taken from [3]. The problem is defined

$$Q = 2 \cdot \begin{pmatrix} 3 & 1 \\ 1 & 1 \end{pmatrix}; \quad c = \begin{pmatrix} 1 \\ 6 \end{pmatrix};$$
$$A_{in} = \begin{pmatrix} -2 & -3 \\ 0 & -1 \end{pmatrix}; \quad b_{in} = \begin{pmatrix} -4 \\ 0 \\ 0 \end{pmatrix},$$

while initial values are chosen as $x_0 = (1,1)^T$ and $\lambda_0 = (0.2, 1, 1)^T$ (like suggested in remark 2). Tolerances and backtracking-paramters are chosen as in Example 1.

With termination after 20 iterations, the solver yields the results

$$x^* = \begin{pmatrix} 0.5000 \\ 1.0000 \end{pmatrix} \tag{22}$$

$$\lambda^* = \begin{pmatrix} 3.0000 \\ 0.0001 \\ 2.812 \cdot 10^{-5} \end{pmatrix} \tag{23}$$

$$f_0(x^*) = 9.2500. \tag{24}$$

V. CONCLUSIONS

Mainly, in this text we introduced the concept of barrier methods for convex optimization problems, which are applied to solve inequality-constraints that include inequality constraints. Therefore we approximate such problems by problems without inequality problems. Solving these problems yields points, that are only suboptimal in the sense of the original problem up to a certain bound. Via analyzing the Lagrangian of the original and the approximative problem, we verified this bound, dependent on the approximation parameter and the number of inequality constraints. The explanation of the barrier concept was followed was by an explanation how it can be applied algorithmically by the barrier method. Further, we gave an idea how to find a feasible initial point and also a time complexity bound of the barrier method, without proof. As an improvement, especially when dealing with quadratic problems, the Primal-Dual Method was introduced, where only one newton step is executed before the approximation parameter is changed. Finally, we showed how quadratic programming can be applied on a linear MPC problem with zero terminal constraints by reformulating it as a quadratic problem.

APPENDIX

A. MATLAB-Implementation

In the following, the MATLAB code of the primal-dual interior point method algorithm is presented

Algorithm main function:

```
1  function [x,fval,lambda,nu,eta] =
       ipquad_pd(Q,c,Aineq,bineq,Aeq,beq,x0,
       lambda0,nu0,gamma,eps_feas,eps_opt,
       ls_alpha,ls_beta)
2  %IPQUAD_PD Quadratic optimization via
       primal-dual-interior-point-method.
3  % Convex Quadr. function f(x) = (1/2)x'
       Qx + c'x with linear equality and
4  % inequality constraints.
5  % ---------------------------------------
6  % Input Arguments:
7  % - Q,c define objective function
8  %   dim(Q) = nxn, dim c = nx1
9  % - Aineq, bineq define inequality
       constraints Aineq*x <= bineq
10 %   dim(Aineq) = mxn, dim(bineq) = mx1
11 % - Aeq, beq define inequality
       constraints Aeq*x == beq
12 %   dim(Aineq) = pxn, dim(beq) = px1
13 % - x0 init. value for the primal
       problem. lambda0, nu0. labda0 >= 0.
14 %   Aineq*x0 <= bineq.
15 %   initial values for the dual problem
16 % - gamma is a reduction factor for
       reducing the barrier weight
       mu_barrier
17 %   in each iteration. gamma in (0,1)
18 % - eps_feas > 0 specifies the tolerance
       for the 2norms of the primal and
19 %   dual residual
20 % - eps_opt > 0 specifies a tolerance on
       the surrogate duality gap
21 % - ls_alpha, ls_beta are parameters for
       the backtracking linesearch,
22 %   performed in each iteration. Typical
       choices: ls_alpha in [0.01,0.1].
23 %   ls_beta in [0.3,0.8]
24 % Outputs:
25 % - x is the best approximation on the
       primal optimum found by the algor.
26 % - lambda, nu are the best
       approximation on the dual optimum
27 % - fval is the objective's value eval.
       at x
28 % - eta is the surrogate duality gap at
       lambda, x
29 % ---------------------------------------
30 % Created: 24.06.20, Daniel Bergmann
31 % ---------------------------------------
32
33 x = x0;
34 lambda = lambda0;
35 nu = nu0;
36
37 % initialize dimensions
38 n = size(Q,1);
39 m = size(Aineq,1);
40 p = size(Aeq,1);
41
42 found = 0;
43 count = 0;
44
45 while found == 0
46     % compute surrogate duality gap
47     eta = -(Aineq*x - bineq)'*lambda;
48     mu_barrier = gamma*eta/m;
49     % Compute KKT residual vector
50     r_mu = res_kkt(x,lambda,nu,Q,c,Aineq
           ,bineq,Aeq,beq,mu_barrier);
51     r_dual = r_mu(1:m);
52     % r_cent = r_mu((m+1):(m+p));
53     r_pri = r_mu((m+p+1):(m+p+n));
54
55     if norm(r_pri) <= eps_feas && norm(
           r_dual) <= eps_feas && eta <=
           eps_opt
56          found=1;
57          break;
58     else
59         % update barrier weighting
               parmeter mu_barrier
60         mu_barrier = gamma*eta/m;
61         % compute search vector
62         [x, lambda, nu] = newtonquad_pd(
               Q, c, Aineq, bineq, Aeq, beq,
               ls_alpha, ls_beta, x,lambda,
               nu, mu_barrier);
63     end
64     count = count+1;
65
66     disp(['Iteration No. ',num2str(count
           ),'; current norm of residual: ',
           num2str(norm([x;lambda;nu])),';
           eta = ',num2str(eta)])
67 end
68
69 % evaluate obj. function at found x
70 fval =  0.5.*x'*Q*x + c'*x;
71 end
```

Newton step including line-search:

```
1  function [x_new, lambda_new, nu_new] =
       newtonquad_pd(Q, c, Aineq, bineq, Aeq
       , beq, ls_alpha, ls_beta, x,lambda,
       nu, mu_barrier)
2  %NEWTONQUAD_PD Computes search direction
       for pd-ip-algorithm via Newton's
```

```matlab
      method and step size
3  %via backtracking line search
4  % Based on a given primal-dual point x,
       lambda,nu, this functions returns new
        points
5  % x_new, lambda_new, nu_new with smaller
        kkt-residual.
6  % First, a search direction is
        determined by applying newton's
        method to
7  % the nonlinear equation system r = 0
        with r the residual of the
8  % kkt-conditions, second a suitable step
        -size is determined via a
9  % backtracking linesearch
10 % --------------------------------------
11 % Input Arguments
12 % - mu_barrier is the current weight on
        the barrier function
13 % - for other input arguments, see
        comments in ipquad_pd.m
14 % Outputs
15 % - x_new, lambda_new, nu_new are the
        new primal-dual point after adding
16 %   the search direction obtained by
        newton's method, multiplied by the
17 %   step size obtained by backtracking
        line search to the old primal-dual
18 %   point
19 % --------------------------------------
20 % Created: 24.06.20, Daniel Bergmann
21 % --------------------------------------
22
23 % initialize dimensions
24 n = size(Q,1);
25 m = size(Aineq,1);
26 p = size(Aeq,1);
27
28 % Define Matrices for KK7T-equality
      M_kkt*deltar = b_kkt
29 if ~isempty(Aeq)
30     M_kkt =  [Q
           Aineq'                   Aeq';...
31          -diag(lambda)*Aineq      -diag(
              Aineq*x-bineq)         zeros(m,p
              );
32          Aeq                      zeros(p,
              m)                     zeros(p,p) ];
33 else
34     M_kkt =  [Q
           Aineq';
35          -diag(lambda)*Aineq      -diag(
              Aineq*x-bineq)];
36 end
37
38 b_kkt = -res_kkt(x,lambda,nu,Q,c,Aineq,
      bineq,Aeq,beq,mu_barrier);
39
40 % Solve KKT-equality to get search
      direction
41 deltaxln = M_kkt\b_kkt;
42
43
44 deltax = deltaxln(1:n);
45 deltalambda = deltaxln((n+1):(n+m));
46 deltanu = deltaxln((n+m+1):(n+m+p));
47
48 % Perform Backtracking linesearch for
      determining suitable step size
49
50 % compute maximal step size smax
51 lambdaquot = [];
52 for i = 1:length(lambda)
53     if deltalambda(i) <0
54         lambdaquot = [lambdaquot, -
              lambda(i)/deltalambda(i)];
55     end
56 end
57
58 smax = min([1  lambdaquot]);
59
60 % Backtracking-Linesearch
61 % Typical Parameter choices:
62 % alpha in [0.01,0.1]; beta in [0.3,0.8]
63 s = 0.99*smax; %S4 0.99 smax??
64 found = 0;
65 while found == 0
66     s = s*ls_beta;
67     x_new = x + s.*deltax;
68     lambda_new = lambda + s.*deltalambda
          ;
69     nu_new = nu + s.*deltanu;
70     r_new = res_kkt(x_new,lambda_new,
          nu_new,Q,c,Aineq,bineq,Aeq,beq,
          mu_barrier);
71     r_old = res_kkt(x,lambda,nu,Q,c,
          Aineq,bineq,Aeq,beq,mu_barrier);
72     if all(Aineq*x_new - bineq < 0) && (
          norm(r_new) <= (1-ls_alpha*s)*
          norm(r_old))
73         found = 1;
74     end
75 end
76 disp(['Backtracking search yields step
      size s = ',num2str(s)])
77 end
```

Computing KKT residual vector from problem matrices and current points:

```matlab
1 function r = res_kkt(x,lambda,nu,Q,c,
      Aineq,bineq,Aeq,beq,mu_barrier)
2 %RES_KKT compute the current KKT-
      residual of a quadratic
3 %optimization problem inside a primal-
```

```
         dual interior point methad.
4  % For the quadr. convex opt. problem
         defined by Q,c,Aineq,bineq,Aeq,beq
5  % (for details see comments on ipquad_pd
         .m) and a current primal-dual point
6  % (x,lambda,nu), this function computes
         the current KKT-residual.
7  %
8  % --------------------------------------
9  % Input Arguments: for detailed
         explanation see comments on ipquad_pd
         .m and
10 % newtonquad_pd.m.
11 % --------------------------------------
12 % Created: 24.06.20, Daniel Bergmann
13 % --------------------------------------
14
15
16 m = size(Aineq,1);
17 if ~isempty(Aeq)
18     % if there are equality constraints
19     r = ...
20         [Q*x + c + Aineq'*lambda + Aeq'*
             nu;...
21         -diag(lambda)*(Aineq*x - bineq)
             - mu_barrier.*ones(m,1);...
22         Aeq*x - beq];
23 else
24     % If there are no equality
         constraints
25     r = [Q*x + c + Aineq'*lambda;...
26         -diag(lambda)*(Aineq*x - bineq)
             - mu_barrier.*ones(m,1)];
27 end
28
29 end
```

B. Model Predictive Control

To illustrate the practical relevance of quadratic programming in e.g. control tasks, we here present the application of interior methods on a Model Predictive Control (MPC) problem. More precisely, a MPCof a linear system with zero terminal constraint (ZTC). We consider the discrete time linear system

$$x_{k+1} = A_d x_k + B_d u_k, \tag{25}$$

where $x_k \in \mathbb{R}^n$, $u_k \in \mathbb{R}^m$ for all $k \in \mathbb{N}_0$, with the constraints that $\|u_k\|_{\max} \leq l_u$ and $\|x_k\|_{\max} \leq l_x$ for all steps $k \in \mathbb{N}_0$. Further we assume that (25) has an equilibrium in $x = 0$ and the current state x_0 is given. The maximum-norm is defined as the maximum absolute value over all entries of the vector, $\|x\|_{\max} = \max_{1 \leq i \leq n} |x_i|$. Goal is to find a input signal that steers the internal state x to zero, while additionally keeping u as small as possibly. Therefore, we consider the next $N \in \mathbb{N}$ timesteps. We call N the prediction horizon. This leads to optimization of a certain objective function over

all possible predicted steering signals $\bar{u} = (\bar{u}_1, \ldots, \bar{u}_{N-1})^T$. We simulate the system for the next N timestep, hence we consider the sequence of states arising from applying a predicted sequence of input signals $\bar{u}$. The sequence of predicted states we denote as $(\bar{x}_0, \ldots, \bar{x}_N)^T$. We choose the quadratic objective function

$$\sum_{k=0}^{N-1} \underbrace{\bar{x}_k^T Q \bar{x}_k}_{=:\|\bar{x}_k\|_Q} + \underbrace{\bar{u}_k^T R \bar{u}_k}_{=:\|\bar{u}_k\|_R} \tag{26}$$

under the condition, that $\bar{x}_N$, the last state in the predicted time, equals zero, to minimize. The regarding predicted states directly follow from the system dynamics, in particular

$$\bar{x}_{k+1} = A_d \bar{x}_k + B_d \bar{u}_k.$$

After finding an optimal signal $\bar{u}$ we apply $\bar{u}_0$ in the next time step. After this first step, we again start an optimization over the next N timesteps. So even though the optimization problem is computed considering all signals up to the prediction horizon, the result is only applied for one timestep, before the next optimization is executed.

We can summarize one optimization step as an optimization problem

$$
\begin{aligned}
\underset{\bar{u}=(\bar{u}_0,\ldots,\bar{u}_{N-1})^T}{\text{minimize}} \quad & \sum_{k=1}^{N-1} \|\bar{x}_k\|_Q + \|\bar{x}_k\|_R \\
\text{subject to} \quad & \bar{x}_0 = x_0, \\
& \bar{x}_N = 0, \\
& \bar{x}_{k+1} = A_d \bar{x}_k + B_d \bar{u}_k \quad \text{for all } k = 0, \ldots, N, \\
& \|\bar{x}_k\| \leq l_x, \|\bar{u}_k\| \leq l_u \quad \text{for all } k = 0, \ldots, N.
\end{aligned} \tag{27}
$$

This problem can be transcribed into the form of (1). We therefore optimize over the whole vector

$$\tilde{x} := \begin{pmatrix} \bar{x} \\ \bar{u} \end{pmatrix}.$$

The objective function (26) then can be written as

$$f_0(\tilde{x}) = \tilde{x}^T H \tilde{x},$$

with the block diagonal matrix

$$H = \text{diag}(\underbrace{Q, \ldots, Q}_{N \cdot n \text{ blocks}}, \underbrace{0}_{n \times n}, \underbrace{R, \ldots, R}_{N \cdot n \text{ blocks}}.)$$

We further formulate the constraints at the maximum norms of state and input as $A_{\text{in}} \tilde{x} \leq b_{\text{in}}$, with $b_{\text{in}} = \mathbb{1} \in \mathbb{R}^{N(n+m)+n \times 1}$ and

$$A_{\text{in}} = \begin{pmatrix} I_{n(N+1)} & 0 \\ -I_{n(N+1)} & 0 \\ 0 & I_{N \cdot m} \\ 0 & -I_{N \cdot m} \end{pmatrix}$$

where the indices of the identity-matrix I denote its dimensional size. Rearranging the system dynamics (25) and taking inital condition as well as zero terminal constraint into account can be written as the equality constraints $A_{\text{eq}} \tilde{x} = b_{\text{eq}}$ with $A_{\text{eq}} = \begin{pmatrix} A_{\text{eq}}^x & A_{\text{eq}}^u \end{pmatrix}$, where

$$A_{\text{eq}}^x = \begin{pmatrix} I_{n \times n} & 0 & \cdots & \cdots & 0 \\ A_d & -I_{n \times n} & \cdots & \cdots & \vdots \\ 0 & \ddots & \ddots & & 0 \\ \vdots & & \ddots & \ddots & 0 \\ \vdots & \cdots & & A_d & -I_{n \times n} \\ 0 & \cdots & \cdots & 0 & I_{n \times n,} \end{pmatrix}$$

$$A_{\text{eq}}^u = \begin{pmatrix} 0 & \cdots & 0 \\ B_d & \ddots & \vdots \\ \vdots & \ddots & 0 \\ 0 & \cdots & B_d \\ 0 & \cdots & 0 \end{pmatrix}.$$

With these matrices, we rewrote (27) as an equivalent problem in shape of (1).

REFERENCES

[1] Carsten Scherer *Vorlesungsskript Einführung in die Optimierung* 2019: Lehrstuhl für Mathematische Systemtheorie, Universität Stuttgart.
[2] Stephen Boyd, Lieven Vandenberghe *Convex Optimization* 2004: Cambridge University Press.
[3] Jack Heider, *Quadratic programming* 2015: online, https://optimization.mccormick.northwestern.edu /index.php/Quadratic_programming.