Preprints of the 21st IFAC World Congress (Virtual)
Berlin, Germany, July 12-17, 2020

The ℓ_1-Exact Penalty-Barrier Phase for Degenerate Nonlinear Programming Problems in Ipopt

David Thierry, Lorenz Biegler *

* *Chemical Engineering Department, Carnegie Mellon University, Pittsburgh, PA 15213 USA (e-mail: biegler@cmu.edu)*

Abstract: Failure to satisfy Constraint Qualifications (CQs) leads to serious convergence difficulties for state-of-the-art Nonlinear Programming (NLP) solvers. Since this failure is often overlooked by practitioners, a strategy to enhance the robustness properties for problems without CQs is vital. Inspired by penalty merit functions and barrier-like strategies, we propose and implement a combination of both in Ipopt. This strategy has the advantage of consistently satisfying the Linear Independence Constraint Qualification (LICQ) for an augmented problem, readily enabling regular step computations within the interior-point framework. Additionally, an update rule inspired by the work of Byrd et al. (2012) is implemented, which provides a dynamic increase of the penalty parameter as stationary points are approached. Extensive test results show favorable performance and robustness increases for our ℓ_1-penalty strategies, when compared to the regular version of Ipopt. Moreover, a dynamic optimization problem with nonsmooth dynamics formulated as a Mathematical Program with Complementarity Constraints (MPCC) was solved in a single optimization stage without additional reformulation. Thus, this ℓ_1-strategy has proved useful for a broad class of degenerate NLPs.

Keywords: Nonlinear programming; Interior point methods; Numerical methods for optimal control; Constraint Qualifications; Complementarity Constraints

1. INTRODUCTION

Nonlinear optimization algorithms have evolved to a state where they demonstrate reasonable performance in solving large-scale problems. This has been enhanced with the availability of accurate first and second-derivatives by means of Automatic Differentiation, and efficient sparse linear algebra packages. These elements are reflected in the implementation of state-of-the-art solvers like Knitro, (Byrd et al. (2006)) and Ipopt, (Wächter and Biegler (2006)). Nevertheless, in many cases nonlinear optimization is avoided by practitioners in favor of convex programming or metaheuristic frameworks, because of the inherent assumptions that NLP entails. Often these assumptions are either overlooked or not understood well enough by the users, so the success rate of the nonlinear optimization solvers is often underappreciated.

Typically, most NLP solvers require several assumptions for convergence, among which regularity of the linearized constraints is fundamental. Moreover, as the nonlinear nature of the problem makes it difficult to satisfy the constraints at all iterates, the linearized problem may become *degenerate*, i.e., constraint gradients are linearly dependent. Additionally, numerical and modeling errors can lead these constraints to lose consistency even at feasible points, and convergence cannot be established. Examples of inconsistent linearizations include problems with bilinear terms, such as gasoline blending and network-flow, Mathematical Programs with Complementarity Constraints (MPCCs) and discretized forms of high-index Differential-Algebraic Equation (DAE) systems.

Several NLP approaches have been designed to work under weaker assumptions in order to overcome these degeneracies. Curtis et al. (2009), present a method that computes normal steps towards feasibility by solving a trust-region problem, followed by solving a perturbed primal-dual system to compute the tangential step. In the limit, even if the linearizations of the constraints are degenerate, the algorithm still converges to local stationary points. Byrd et al. (2012) incorporated an SQP strategy with an exact penalty of the constraint violation and a line-search. Provided that the reduced Hessian of the Lagrangian is positive definite, this strategy has inherent regularizing properties. Nevertheless, the algorithm must allocate considerable effort to finding appropriate values of the penalty parameter.

In terms of available NLP solvers, Ipopt is one of the most competitive codes for large-scale optimization. At its core Ipopt employs a filter line-search with an interior-point method to converge to the solution. To deal with loss of regularity, it uses inertia controlling mechanisms, which detect linear dependencies and attempt to regularize the linearized subproblem. Nevertheless, these mechanisms compete with several factors, such as ill-conditioning and numerical instability, which limit their success. Wang et al. (2013) and Wan and Biegler (2017) showed that these degeneracies can be removed directly at the linear algebra level by analyzing the pivot sequence and perturbing or

removing null rows of the Jacobian block of the KKT matrix. However, the effectiveness of this strategy is linked to the tuning parameters of the sparse-linear algebra routines.

In this work we present a strategy that combines exact penalties, filter line-search and barrier algorithms within Ipopt to solve problems with degenerate and inconsistent constraint linearizations. The fundamental penalty-barrier problem will be introduced alongside a strategy to adjust the value of the penalty parameter. Finally, numerical results for a test set of degenerate problems and a control problem with degenerate constraints will be presented.

2. THE INTERIOR-POINT ℓ_1-EXACT PENALTY-BARRIER PHASE

Throughout this work we consider a general, nonconvex NLP in the following compact form:

$$\underset{x\in\mathbb{R}^n}{\text{minimize}}\ f(x) \quad \text{subject to } c(x)=0, \quad x\geq 0; \tag{1}$$

where $f:\mathbb{R}^n\to\mathbb{R}$ and $c:\mathbb{R}^n\to\mathbb{R}^m$ are assumed to be C^2 bounded Lipschitz continuous functions and derivatives over $\mathbb{R}^n$. Typically, algorithms that attempt to find a solution of problem (1) do so by finding a point that satisfies KKT (Karush-Kuhn-Tucker) conditions. However, such strategies tacitly assume that the set of constraints are *regular*, in other words, they satisfy *constraint qualifications* (CQs). To define them we denote $\nabla c(x)$ as the constraint gradients, and assume all vectors to be column vectors unless specified otherwise.

Definition 1. (Active set). Consider the active set of bounds for a feasible point $\overline{x}$,

$$\mathcal{A}(\overline{x})=\{i:\overline{x}_i=0\}, \tag{2}$$

with $n_{\mathcal{A}}=|\mathcal{A}(\overline{x})|$. We define matrix $E_{\mathcal{A}}(\overline{x})\in\mathbb{R}^{n\times n_{\mathcal{A}}}$ such that $E_{\mathcal{A}}(\overline{x})^T\overline{x}=0$. Here column j of $(E_{\mathcal{A}})$ is defined by a column of an identity matrix of appropriate size, i.e. $E_{\mathcal{A},j}(\overline{x})=I_{i(j)}$ where $i(j)$ is the j^{th} element of $\mathcal{A}(\overline{x})$.

Definition 2. (LICQ). The linear independence constraint qualification holds at a feasible point $\overline{x}$ of the NLP (1) if the matrix of combined gradients of equality constraints and active bounds, $[\nabla c(\overline{x})\ \ E_{\mathcal{A}}(\overline{x})]$, is full column rank.

Definition 3. (MFCQ). The Mangasarian Fromovitz Constraint Qualification holds at a feasible point $\overline{x}$ of NLP (1) if the gradients of the equality constraints at $\overline{x}$, $\nabla c_i(\overline{x}), i\in\{1,...,m\}$ are linearly independent and there exists a nonzero vector s such that,

$$\begin{aligned}&\nabla c_i(\overline{x})^T s=0 \quad \forall i\in\{1,...,m\}, \quad \text{and}\\ &E_{\mathcal{A},i}(\overline{x})^T s>0 \quad \forall i\in\mathcal{A}(\overline{x}).\end{aligned} \tag{3}$$

Though there are entire families of CQs, the MFCQ is commonly used as part of the foundation to guarantee convergence of most NLP solvers. It should be noted that LICQ implies MFCQ but not the reverse. Also, if MFCQ is satisfied, the multipliers associated with a nonlinear program will be bounded. For degenerate problems, these CQs can fail in a neighborhood of points within the feasible region, or in some severe cases, at all points. The latter is the typical case for redundant constraints, i.e. sets of constraints that add no information to the feasible region, or are already implied by the rest of the constraints.

Whenever CQs fail, most solvers will engage strategies that attempt to counteract the degeneracies. Most notably, active set methods that are able to detect this situation, could in principle remove degenerate constraints. However, as the size of the problem grows, the combinatorial nature of such a mechanism becomes problematic. On the other hand, interior-point methods like Ipopt do not deal with single sets of degenerate constraints at a given time, but rather attempt to regularize all the constraints at once, with the hope that the degeneracies are limited to an isolated neighborhood of points. Otherwise, it is necessary to further analyze the linearizations of the set of constraints to determine the source of the redundancy and eliminate it from the set. To further expand on this topic, the next section summarizes the barrier strategy in Ipopt.

2.1 Barrier Problem and Solution of Primal-Dual System

Barrier methods work on the premise of solving a family of parametric problems that asymptotically approaches the solution of (1). These are constructed by adding a *barrier* term in the objective function that remains bounded as long as a strictly feasible point is being evaluated; and as the boundary of the feasible region is approached, the barrier smoothly increases to infinity. In other words, x remains in the strict interior of the positive orthant. The associated barrier problem for (1) is written as,

$$\min_{x\in\mathbb{R}^n}\varphi_{\mu_j}(x)\coloneqq f(x)-\mu_j\sum_i^n\ln x^{(i)} \text{ s.t. } c(x)=0, \tag{4}$$

where $\mu_j\in\mathbb{R}^1_{>0}$ is the barrier parameter, and $\varphi_{\mu_j}(x)$ denotes the barrier objective function for a fixed barrier parameter at iteration j. Under the MFCQ, as $\lim_{j\to\infty}\mu_j=0$, the solution of (4) $(x(\mu_j))$ describes a differentiable path to the solution of (1), x^* (Wright and Orban (2002)). In Ipopt, the algorithm first attempts to solve directly the set of primal-dual equations, which are related to the KKT conditions of (4). To illustrate this, consider the solution triplet consisting of primal-dual variables $(x(\mu_j),\lambda(\mu_j),z(\mu_j))\in\mathbb{R}^n\times\mathbb{R}^m\times\mathbb{R}^n$, where $x\geq 0$, and $z\geq 0$; also let $g(x):=\nabla f(x)$, $A(x):=\nabla c(x)$, and $X:=\text{diag}(x)$, $Z:=\text{diag}(z)$. Then the primal-dual equations are given as

$$\begin{bmatrix} g(x)-z+A(x)\lambda \\ c(x) \\ XZe-\mu_j e\end{bmatrix}=0. \tag{5}$$

To solve system (5), one can use Newton's algorithm to generate k steps towards the solution, i.e. $x_k:=x_k+\alpha_k d_k^x$, $z_k:=z_k+\alpha_k^z d_k^z$, and $\lambda_k:=\lambda_k+\alpha_k^\lambda d_k^\lambda$. This entails solving a linear system at iteration k for some fixed value of μ_j to find the search direction $\left(d_k^x,d_k^\lambda,d_k^z\right)$. The $(2n+m)$ order linear system associated with the Newton step of (5) is not symmetric; however it can be decomposed to an $(n+m)$ symmetric linear system that is solved instead. Let the Lagrange function be $\mathcal{L}(x_k,\lambda_k,z_k):=f(x_k)-z_k^Tx_k+c(x_k)^T\lambda_k$, so that the Hessian matrix of the Lagrange function is $H_k:=\nabla^2_{xx}f(x_k)+\sum_i^m\nabla^2_{xx}c_i(x_k)\lambda_k^{(i)}$. Then the diagonal primal-dual terms are $\Sigma_k:=X_k^{-1}Z_k$, and finally the augmented Hessian matrix is $W_k:=H_k+\Sigma_k$. (To further simplify the notation, whenever the equation is related to a Newton type iteration k, we use subscripts

to denote function values of the iterates; e.g. $g_k \coloneqq g(x_k)$). The resulting symmetric linear system is then

$$\begin{bmatrix} W_k & A_k \\ A_k^T & 0 \end{bmatrix} \begin{bmatrix} d_k^x \\ d_k^\lambda \end{bmatrix} = - \begin{bmatrix} g_k - \mu_j X_k^{-1} e + A_k \lambda_k \\ c_k \end{bmatrix}, \quad (6)$$

where the remaining solution vector is given in terms of d_k^x, i.e. $d_k^z = \mu_j X_k^{-1} e - z_k - \Sigma_k d_k^x$. The system in (6) is usually sparse, so it can be solved using an indefinite sparse linear solver, like `MA57`. After successfully computing the search direction, Ipopt will attempt to compute an appropriate value of $\alpha_k \in (0, 1]$, that promotes convergence to the solution of (1). This strategy assumes that the linearizations of the constraints are regular, i.e. the LICQ holds for (4). Thus, whenever A_k becomes rank deficient, factorization of (6) cannot be performed. As a result regularization strategies are fundamental for the convergence of the algorithm.

Remark 4. Failure of the LICQ for problem (4) will compromise the convergence properties. Moreover, it is required that the projection of W_k onto the null-space of A_k^T to be positive definite, so that computed directions for x have a descent property. Both can be verified by checking the *inertia* of the matrix from system (6). If the inertia is not correct, the augmented KKT matrix can be modified by defining regularization scalars, $\delta, \delta_c \in \mathbb{R}^1_{\geq 0}$, and then adding them to the two diagonal blocks respectively. i.e.

$$\begin{bmatrix} W_k + \delta I & A_k \\ A_k^T & -\delta_c I \end{bmatrix}. \quad (7)$$

Therefore if either positive definiteness of the projected H_k or the LICQ are not satisfied, the corresponding regularization scalar will be strictly positive. For the case in which A_k is rank-deficient (i.e. violates the LICQ), then $\delta_c \leftarrow \overline{\delta}_c \mu_j^{\kappa_c}$ and its value is proportional to the current barrier parameter. On the other hand, for particularly ill-conditioned problems, this regularization may not be sufficient to fix the rank deficiencies, and the algorithm will be sent into *restoration* phase (see Wächter and Biegler (2006)).

2.2 ℓ_1-*Exact Penalty-Barrier Primal-Dual Strategies*

In this study we reformulate the interior point strategy to provide a self-regularizing effect. We create an exact penalty form of the barrier problem, which yields an equivalent solution to that of the problem (1) and at the same time does not require the LICQ assumption for the equality constraints. To begin, we penalize the ℓ_1-norm of the constraints from (4) and write the nonsmooth formulation of the ℓ_1-penalty-barrier form as:

$$\underset{x \in \mathbb{R}^n}{\text{minimize}} \quad f(x) - \mu_j \sum_i^n \ln x^{(i)} + \rho \left\| c(x) \right\|_1, \quad (8)$$

where $\rho \in \mathbb{R}^1_{\geq 0}$ is the penalty parameter. If this problem has $\rho \geq \rho^* := \left\| (\lambda^*, z^*) \right\|_\infty$, and $c(x) = 0$; then the solution of (8) is the same as the solution of (4) (Han and Mangasarian (1979)). Therefore, it is critical to select a sufficiently large penalty parameter to reach a KKT point. Moreover, it is desirable to solve a differentiable version of problem (8), which requires augmenting the set of constraints with additional penalty-variables p and $n \in \mathbb{R}^m$. This is denoted as the *direct* penalty strategy, i.e.

$$\begin{aligned} \underset{x \in \mathbb{R}^n;\, p,n \in \mathbb{R}^m}{\min} \; & f(x) - \mu_j \sum_i^n \ln x^{(i)} + \rho (p+n)^T e \\ \text{subject to} \quad & c(x) - p + n = 0,\; p \geq 0, n \geq 0, \end{aligned} \quad (9)$$

where $e = [1, 1, ...1]^T$ denotes a vector of all ones of the appropriate dimension. The resulting problem is feasible at all points (there exist nonnegative p or n such that the constraints are satisfied); however this has the trade-off of higher dimensionality and it needs a mechanism to determine value of the penalty parameter. Moreover, this problem satisfies LICQ at infeasible points of (4), thus enabling computation of steps for degenerate problems. This is shown in the following remark.

Remark 5. Consider a point $\overline{x}$ for which $c(\overline{x}) \neq 0$ and $p + n > 0$, then the LICQ for problem (9) still holds, even if $A(\bar{x})$ is rank deficient.

With these properties, the resulting primal-dual equations include new barrier terms for p and n inequalities, and stationarity conditions as well. Using a similar strategy as the one from the previous section, the solution vector of this problem is given as

$$(x(\mu_j), \lambda(\mu_j), z(\mu_j), p(\mu_j), n(\mu_j), z_p(\mu_j), z_n(\mu_j))$$

and we solve corresponding linear systems to determine the Newton search direction $\left(d_k^x, d_k^\lambda, d_k^z, d_k^p, d_k^n, d_k^{z_p}, d_k^{z_n}\right)$. Let $Z_{p,k} \coloneqq \operatorname{diag}(z_{p,k})$, $Z_{n,k} \coloneqq \operatorname{diag}(z_{n,k})$ so that $\Sigma_{p,k} \coloneqq P_k^{-1} Z_{p,k}$ and $\Sigma_{n,k} \coloneqq N_k^{-1} Z_{n,k}$. In a similar way as system (6), this search direction can be found by solving the following set of equations,

$$\begin{bmatrix} W_k & A_k \\ A_k^T & -\Sigma_{p,k}^{-1} - \Sigma_{n,k}^{-1} \end{bmatrix} \begin{bmatrix} d_k^x \\ d_k^\lambda \end{bmatrix} = - \begin{bmatrix} g_k - \mu_j X_k^{-1} e + A_k \lambda_k \\ \tilde{c}_k \end{bmatrix} \quad (10)$$

where

$$\tilde{c}_k \coloneqq c_k + Z_{p,k}^{-1} [\rho P_k - \mu_j e] + Z_{n,k}^{-1} [-\rho N_k + \mu_j e] - \lambda_k [\Sigma_{p,k} + \Sigma_{n,k}]. \quad (11)$$

Finally, the remaining search directions are defined in terms of $\lambda_k^+ := \lambda_k + d_k^\lambda$ as follows,

$$\begin{aligned} d_k^p &= Z_{p,k}^{-1} \left[\mu_j e + P_k \lambda_k^+ - \rho p_k \right], \\ d_k^n &= Z_{n,k}^{-1} \left[\mu_j e - N_k \lambda_k^+ - \rho n_k \right], \\ d_k^{z_p} &= \mu_j P_k^{-1} e - z_{p,k} - \Sigma_{p,k} d_k^p, \\ d_k^{z_n} &= \mu_j N_k^{-1} e - z_{n,k} - \Sigma_{n,k} d_k^n. \end{aligned} \quad (12)$$

Remark 6. Even though the ℓ_1-penalty-barrier problem of (9) avoids degeneracies in the linear system (10), the values of the multipliers are proportional to the value of the penalty parameter. At some point their contribution to the ill-conditioning of the linear system might become significant and the sparse linear solver will encounter difficulties. Therefore, an *inverse* penalty strategy is also proposed. This is given as

$$\begin{aligned} \underset{x \in \mathbb{R}^n;\, p,n \in \mathbb{R}^m}{\text{minimize}} \quad & \frac{1}{\rho} \left[f(x) - \mu_j \sum_i^n \ln x^{(i)} \right] + (p+n)^T e \\ \text{subject to} \quad & c(x) - p + n = 0, \quad p, n \geq 0. \end{aligned} \quad (13)$$

From the optimality conditions of (13), it can be deduced that this formulation limits the values of the multipliers, and as $\rho \to \infty$, the corresponding primal-dual system will converge to a (possibly infeasible) stationary point of (1).

2.3 About the Penalty Parameter Update

The value of the penalty parameter is critical for the convergence of the algorithm; therefore, an appropriate update rule needs to be implemented.

A possible strategy is to use linearized models of the infeasibility and the barrier objective function, so that predictions on the minimum value of ρ can be made. First, consider a linear model of the ℓ_1-norm infeasibility,

$$m_k\left(d^x\right) := \left\|A_k^T d^x + c_k\right\|_1. \tag{14}$$

We note that for large values of the penalty parameter, the penalty-barrier objective function creates a bias towards feasibility. As with trust region algorithms, it is desirable to select a penalty parameter value, such that the predicted reduction of the quadratic model of the penalty-barrier objective for a search direction d^x is commensurate to the current norm of the infeasibility. For this let the quadratic model of the penalty-barrier objective function be given by,

$$q_{\rho,k}\left(d^x\right) := \varphi_{\mu_j,k} + \nabla\varphi_{\mu_j,k}^T d^x + d^{xT} W_k d^x + \rho\, m_k\left(d^x\right), \tag{15}$$

and the predicted barrier function change through this quadratic model be given by $q_{\rho,k}(0) - q_{\rho,k}(d)$. As mentioned earlier an adequate penalty parameter will generate search directions that balance stationarity and feasibility. Thus, an acceptable penalty parameter enables the predicted reduction to be greater than the penalized infeasibility, i.e.,

$$q_{\rho,k}(0) - q_{\rho,k}(d) \geq \rho\kappa_\rho \left\|c_k\right\|_1. \tag{16}$$

This rule is sufficient to derive a lower bound on the penalty parameter. One can use information from the Newton step of the primal-dual system of (9), and then derive this lower bound as the following result,

$$\rho \geq \rho_{\text{trial}} := \frac{\nabla\varphi\left(x_k\right)^T d_k^x + \dfrac{\gamma_\rho}{2} d_k^{xT} W_k d_k^x}{\left(1-\kappa_\rho\right)\left\|c_k\right\|_1 - \left(p_k^+ + n_k^+\right)^T e}, \tag{17}$$

with the vectors $p_k^+ = p_k + d_k^p$, $n_k^+ = n_k + d_k^n$, and the parameter $\gamma_\rho \in \{0,1\}$ is added so that whenever $d_k^{xT} W_k d_k^x < 0$, $\gamma_\rho = 0$; otherwise $\gamma_\rho = 1$. Using (17), an update of the penalty parameter can be done by checking that $\rho \geq \rho_{\text{trial}}$ and updating $\rho := \rho_{\text{trial}} + \epsilon_\rho$ otherwise. This rule defines the update of the penalty parameter based on current local information. However, on some problems the rate at which the penalty is updated is not large enough. Therefore, an additional heuristic is imposed to improve the rate of growth of the penalty parameter; we *double* the value of ρ if there is insufficient progress towards feasibility, as measured using the denominator of (17), i.e.,

$$\rho = \begin{cases} 2\,\rho & \text{if} \quad \left(1-\kappa_\rho\right)\left\|c_k\right\| - \left(p_k^+ + n_k^+\right)^T e < 0 \\ \text{unchanged} & \text{otherwise} \end{cases}. \tag{18}$$

3. NUMERICAL RESULTS

In order to perform numerical tests, the Ipopt libraries were compiled with GCC and GFortran 7.4.0. For the BLAS and LAPACK routines, the sequential version of the Intel MKL library v2018.1-163 was used alongside HSL's MA57 as the linear solver. All the relevant options for Ipopt are set to their default values and all of the problems were compiled using AMPL in a Linux machine with a Intel(R) Xeon(R) E5-2440 CPU.

3.1 CUTEr Test Set

To evaluate the performance and compare our proposed NLP solver, we run a variant of the CUTEr test set and construct performance plots based on the results. However, to the best of our knowledge most of the equality constrained problems in CUTEr do not pose enough CQ failure situations that lead IPOPT to terminate without convergence. Therefore, in addition to selecting 275 problems from the standard CUTEr test set, a modified version was generated in which all 275 problems include a globally-dependent constraint. Since none of these modified problems satisfies LICQ, this provides a more informative certificate of robustness of the solver. For each of the 275 CUTEr problems, we choose a particular constraint, called $c_1(x)$, and add the corresponding redundant constraint,

$$c_1(x) = 0, \text{ and } \quad c_1(x) - c_1^2(x) = 0. \tag{19}$$

This is the same modification of the test used by Curtis et al. (2009). For all the test problems, the ℓ_1-penalty phase was set up to use 6 different updates of the penalty parameter, based on (17). These were compared against direct and inverse problems with penalty parameters fixed at 1000, as well as the normal version of Ipopt, which is designated here as *vanilla*. We tested and compared the following ℓ_1-penalty options:

- *direct*-quadratic update of (ρ)
- *direct*-quadratic-no-Σ_k
- *direct*-linear update of (ρ_L, $\gamma_\rho = 0$ in (17))
- *direct*-fixed (ρ_0)
- *inverse*-quadratic ($1/\rho$)
- *inverse*-quadratic-no-Σ_k
- *inverse*-linear ($1/\rho_L$, $\gamma_\rho = 0$ in (17))
- *inverse*-fixed ($1/\rho_0$).

Since *vanilla* Ipopt is already competitive with the standard CUTEr set, we show that replacing the current *feasibility restoration* in IPOPT with the ℓ_1-penalty phase problem (9) leads to a positive improvement on robustness. For the modified CUTEr problems that have globlal rank-deficiencies, we apply ℓ_1-penalty strategies throughout the optimization, and start from the ℓ_1-phase directly. It should be noted that, in contrast to the current *feasibility restoration* phase within Ipopt, our ℓ_1-penalty strategies do not revert back to the standard algorithm (based on (4)), but remain engaged until the problem terminates.

The Dolan-Morè plots for the standard CUTEr set are shown in Figure 1 and for the modified (degenerate) CUTEr set in Figure 2. For both tests, the fixed penalty options $(\rho_0, 1/\rho_0)$ perform worse than *vanilla* Ipopt in terms of robustness, because it is likely that ρ_0 remains too small when an ill-conditioned stationary point is approached. Nevertheless, from Figure 1 we see that there is a significant benefit from the updates; most notably the quadratic rule (ρ) outperforms all of the rest of the updates.

For the modified CUTEr set, we note that the *inverse* penalty strategies are at least as robust as the *vanilla* counterpart. On the other hand, the *direct* penalty options (ρ, *linear* and *no* Σ) are the top in terms of robustness, among which the *linear* update option is the most robust. In spite of this, a few failure cases are still encountered with the ℓ_1-penalty phase. These are due to reaching

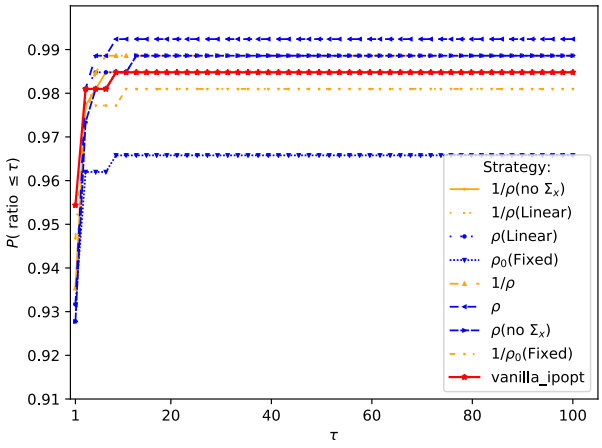

Fig. 1. Dolan-Morè performance plots for the standard CUTEr test set. (p is the fraction of problems solved by a given solver within a factor τ of the minimum time to solve.)

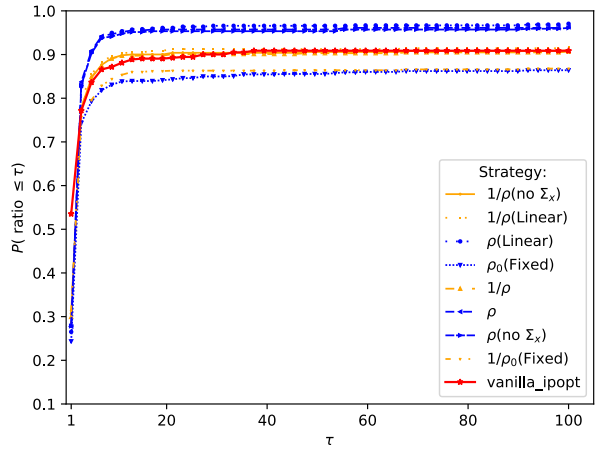

Fig. 2. Dolan-Morè performance plots for modified CUTEr test set with a degenerate constraint

infeasible stationary points, and attaining feasible points $(p, n \to 0)$ at a faster rate than progress towards stationarity. It should be noted that both cases are linked to several factors that include the initial guesses for primal variables, and most notably the initial penalty parameter. In particular, initializaton for the penalty parameter influences the convergence path significantly and the above failures can be avoided with better initial values of ρ.

In contrast to the *vanilla* version, the *direct* ℓ_1-penalty phase in Ipopt is successful in solving several problems, such as *hs047*, *ssnlbeam*, *eigenc2* and the *brainpc-x* problems. At the same time, problems like *qpnboei1*, *steenbrf* and *gridnetc* converge to feasible points at which the penalty parameter grows indefinitely. These failures are avoided with the *inverse* approaches, which solve these problems to an acceptable level (i.e. relaxed tolerances). On the other hand, the *inverse* approaches encounter other failures, which drop their robustness to the same level as *vanilla*. These occur because feasibility is attained faster than optimality in several cases. When continuing towards stationarity, the line-search selects the minimum possible step size; as the penalty-variables are already at the bound, this triggers immediate failure. This is experienced in problems *blockqp5* and *ncvxqp9*. Nevertheless, the overall results show a significant increase of robustness for Ipopt, along with some caveats including the possibility of finding stationarity points that are not optimal solutions.

3.2 Optimal Control of a Nonsmooth DAE

We also consider a discretized dynamic optimization problem, which is an instance of a Mathematical Program with Complementarity Constraints (MPCC). The optimization problem is given as,

$$\begin{aligned}
&\min\ \alpha \left\|\mathbf{x}\left(T\right)-\mathbf{x}_{\mathrm{tgt}}\right\|^2+T\\
&\text{s.t. } \dot{\mathbf{x}}=\mathbf{v},\\
&\quad \dot{\mathbf{v}}=a\left(t\right)\mathbf{t}\left(\theta\right)+F\mathbf{n}\left(\theta\right),\\
&\quad \dot{\theta}=s\left(t\right)\left(\mathbf{t}\left(\theta\right)^T\mathbf{v}\right),\\
&\quad F\in-\mu N\mathrm{Sgn}\left(\mathbf{n}\left(\theta\right)^T\mathbf{v}\right),\\
&\quad y_{cl}\left(x\right)=\begin{cases}\sin\left(x\right) & x\le\pi\\ \pi-x & \pi\le x\le 2\pi\\ -\pi-\sin\left(x\right) & 2\pi\le x\end{cases},\\
&\quad \left|a\left(t\right)\right|\le a_{\max},\ \left|s\left(t\right)\right|\le s_{\max}\quad\forall t,\\
&\quad \mathbf{x}\left(0\right)=\mathbf{0},\ \mathbf{y}\left(0\right)=\mathbf{0},\ \theta\left(0\right)=0,\\
&\quad \mathbf{x}\in C=\left\{(x,y)\mid\left|y-y_{cl}\left(x\right)\right|\le w/2\right\}.
\end{aligned}\tag{20}$$

The problem models a racing car undergoing frictional forces, with the goal to finish the track in the least amount of time. The model has as state variables a 2-dimensional position vector $\mathbf{x}=[x,y]^T$, the velocity vector $\mathbf{v}$ and a scalar angle θ which indicates the direction of the car. The controls are the throttle level $a\left(t\right)$ of the car, that dictates whether the car accelerates or decelerates, and the steering control $s\left(t\right)$ which sets the direction of the car. There is also the unit vector that points into the direction of θ, $\mathbf{t}\left(\theta\right)=\left[\cos\left(\theta\right),\sin\left(\theta\right)\right]^T$, and its normal vector $\mathbf{n}\left(\theta\right)=\left[-\sin\left(\theta\right),\cos\left(\theta\right)\right]^T$. The model is a DAE system with equations for velocity, momentum balance, a differential inclusion, where the frictional force depends on the sign of the normal velocity, and constraints on the position of the car, which is limited by the walls of an s-shaped track. The objective minimizes T, the time to finish the track plus the difference between the final position and a target $\mathbf{x}_{\mathrm{tgt}}$.

Typically, optimization problems with smooth DAEs can be solved by fully discretizing the time domain, so that the discrete solution approximates the infinite-dimension solution. However, the differential inclusions (and piecewise function) introduce an additional layer of complexity. Rather than using smoothing functions, this problem can be solved by transforming such constraints into complementarity constraints (Baumrucker and Biegler (2009)) thus turning the problem into a MPCC. Although MPCCs are an equivalent smooth representation of the nonsmooth parts of the model, they are particularly difficult to handle because MPCCs violate constraint qualifications. Therefore in order to apply off-the-shelf NLP solvers for these kinds of problems, it is necessary to apply reformulations, for example a max operator followed by a smoothing function or solving sequences of relaxed problems. Such reformulations are not required here, since the ℓ_1-penalty

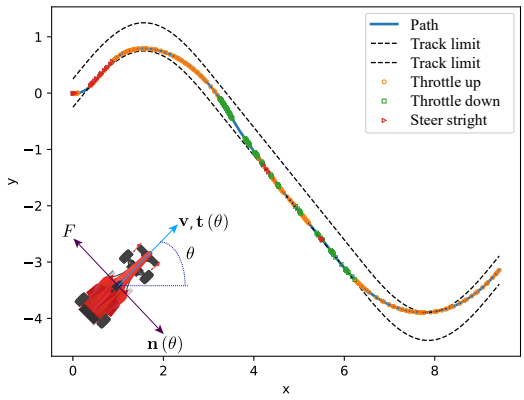

Fig. 3. State profiles for the racing car MPCC problem

modes presented in this work are already designed to handle the lack of regularity of the MPCC *directly*.

The MPCC derived from (20) was discretized with Radau collocation with 100 finite elements and 3 collocation points in GAMS 28.2.0 (using the AMPL writer). Additionally the ℓ_1-exact penalty phase was used to solve this problem with `MA57` as the linear solver. With this discretization the problem has 8896 variables and 8296 constraints; also for the complementarity constraints, the equality constraint formulation was used. This problem was solved in 9.5258 CPUs and 315 iterations to acceptable level in a *single* optimization problem. For reference, the *vanilla* version of Ipopt always fails to solve this problem.

The profiles for the optimal time trajectory are displayed in Figure 3. For this particular run, a final time of 4.947 seconds was obtained. It also shows that as the car approaches both bends, it throttles all the way up and turns sharply at the same time.

We note that it is possible to get even better *local* solutions, as suggested by Baumrucker and Biegler (2009), if careful pre- and post-processing is done. (In fact, with further mesh refinement, the best reported time is 4.0169 seconds.) Nevertheless, the robustness of our ℓ_1-penalty algorithms show that ill-posed MPCCs can be solved directly, even when standard NLP solvers fail.

4. CONCLUSION

Ill-posed problems lack regularity of local constraint linearizations. This situation curtails the performance and robustness of state-of-the-art solvers like Ipopt. Our proposed ℓ_1-exact penalty-barrier approach has the advantage of satisfying the LICQ as feasible stationary points are reached. Moreover, combined with penalty update rules, it is possible to enhance the robustness properties of Ipopt.

Numerical testing reveals that there is considerable increase in robust performance, although pathological problems may still lead to failures. On the other hand, switching to *inverse* strategies of penalty shows favorable results on these problems. Finally, a challenging MPCC was solved directly as an NLP without introducing additional strategies to handle complementarity constraints. These results demonstrate that our ℓ_1-penalty algorithm can be useful for a broad class of degenerate problems.

5. DISCLAIMER

REFERENCES

Baumrucker, B. and Biegler, L. (2009). MPEC strategies for optimization of a class of hybrid dynamic systems. *J. Process Control*, 19(8), 1248–1256.

Byrd, R.H., Lopez-Calva, G., and Nocedal, J. (2012). A line search exact penalty method using steering rules. *Math. Prog.*, 133(1-2), 39–73.

Byrd, R.H., Nocedal, J., and Waltz, R.A. (2006). K nitro: An integrated package for nonlinear optimization. In *Large-scale nonlinear optimization*, 35–59. Springer.

Curtis, F.E., Nocedal, J., and Wächter, A. (2009). A matrix-free algorithm for equality constrained optimization problems with rank-deficient jacobians. *SIAM J. Opt.*, 20(3), 1224–1249.

Han, S.P. and Mangasarian, O.L. (1979). Exact penalty functions in NLP. *Math. Prog.*, 17(1), 251–269.

Wächter, A. and Biegler, L.T. (2006). On the implementation of an interior-point filter line-search algorithm for large-scale NLP. *Math. Prog.*, 106(1), 25–57.

Wan, W. and Biegler, L.T. (2017). Structured regularization for barrier NLP solvers. *Computational Opt. & Applics.*, 66(3), 401–424.

Wang, K., Shao, Z., Lang, Y., Qian, J., and Biegler, L.T. (2013). Barrier NLP methods with structured regularization for optimization of degenerate optimization problems. *Comp. Chem. Engr.*, 57, 24–29.

Wright, S.J. and Orban, D. (2002). Properties of the log-barrier function on degenerate nonlinear programs. *Mathematics of Operations Research*, 27(3), 585–613.