Solving MPCC Problem with the Hyperbolic Penalty Function

Teófilo Melo*, M. Teresa T. Monteiro† and João Matias**

*School of Technology and Management of Felgueiras, Porto Polytechnic Institute, Portugal
tmm@estgf.ipp.pt
†Department of Production and Systems, University of Minho, Portugal
tm@dps.uminho.pt
**Centre of Mathematics, CM-UTAD, Portugal
j_matias@utad.pt

Abstract. The main goal of this work is to solve mathematical program with complementarity constraints (MPCC) using nonlinear programming techniques (NLP). An hyperbolic penalty function is used to solve MPCC problems by including the complementarity constraints in the penalty term. This penalty function [1] is twice continuously differentiable and combines features of both exterior and interior penalty methods. A set of AMPL problems from MacMPEC [2] are tested and a comparative study is performed.

Keywords: Penalty technique, MPCC, SQP, NLP
PACS: 87.55.de

INTRODUCTION

Equilibrium constraints as complementarity conditions frequently appear in many Engineering and Economics applications. There are many practical problems that can be modulated using MPCC formulation, such as friction problems, traffic congestion and process design in Engineering or game theory models like Nash and Stackelberg equilibrium, finance and taxes in Economics. Complementarity optimization problems are most common in these areas because the concept of complementarity is synonymous of the notion of system equilibrium. This optimization problem is hard to solve due to the failure of the standard Mangasarian-Fromovitz constraint qualification (MFCQ) in nonlinear programming [3]. Another difficulty is the combinatorial nature of the complementarity constraints, which make difficult to develop efficient algorithms. Several strategies have been proposed to solve MPCC namely regularization, smoothing, nonlinear program reformulation, penalty techniques and sequential quadratic programming (SQP). Fukushima and Scholtes [4] studied the convergence of smoothing and regularization methods, respectively, determining stationary points of a sequence of nonlinear programs. Fletcher [5], has shown that SQP methods converge locally to strongly stationary points. Hu and Ralph [6] concluded that a penalty framework shares convergence properties with regularization and smoothing methods.

PROBLEM DEFINITION

We consider mathematical problems with complementarity constraints (MPCC):

$$\begin{array}{ll} min & f(x) \\ s.t. & c_i(x) = 0,\ i \in E, \\ & c_i(x) \geq 0,\ i \in I, \\ & 0 \leq x_1 \perp x_2 \geq 0, \end{array} \tag{1}$$

where $x = (x_0, x_1, x_2)$ is the decomposition of the problem variables into $x_0 \in \mathbb{R}^n$ (control variables) and $(x_1, x_2) \in \mathbb{R}^{2q}$ (state variables). f and c are the nonlinear objective function and constraint functions respectively, assumed to be twice continuously differentiable. E and I are two disjoined finite index sets with cardinality p and m. The notation $\perp$ represents complementarity, it means that $x_{1j}x_{2j} = 0$ for $j = 1, ..., q$. The complementarity condition has a disjunctive

Numerical Analysis and Applied Mathematics ICNAAM 2011
AIP Conf. Proc. 1389, 763-766 (2011); doi: 10.1063/1.3636844

nature - $x_{1j} = 0$ or $x_{2j} = 0$ for $j = 1,...,q$. More general complementarity constraints can be included in such $0 \leq G(x) \perp H(x) \geq 0$ by adding slack variables. Adding slacks does not destroy any properties of the MPCC such as constraint qualification or second-order condition.

One convenient way of solving MPCC problems replaces the complementarity constraints by $x_1, x_2 \geq 0$ and $x_{1j}x_{2j} \leq 0$ for $j = 1,...,q$ transforming the MPCC formulation into an equivalent nonlinear program (NLP):

$$\begin{array}{ll} min & f(x) \\ s.t. & c_i(x) = 0,\ i \in E, \\ & c_i(x) \geq 0,\ i \in I, \\ & x_1 \geq 0, x_2 \geq 0, \\ & x_{1j}x_{2j} \leq 0,\ j = 1,\ldots,q. \end{array} \tag{2}$$

This reformulation allows to solve MPCC problems using some NLP techniques.

OPTIMAL CONDITIONS

The NLP formulation has no feasible point that strictly satisfies the inequalities. This fact implies that the Mangasarian-Fromovitz constraint qualification (MFCQ) is violated at every feasible point [3]. This failure has consequences: the multiplier set is unbounded, the central path fail to exist, the active constraints normals are linearly dependent and linearizations of the NLP formulation can be inconsistent arbitrarily close to the solution. Recent developments show that a SQP method with an elastic mode to solve MPCC converges locally [7] and that there is a relationship between strong stationarity defined by Scheel and Scholtes and the Karush-Kuhn-Tucker (KKT) points. This relationship established convergence of SQP methods for MPCC formulated as NLP. Some optimality concepts used in this work are based on the study of [5].

HYPERBOLIC PENALTY METHOD

A way to deal with the complementarity constraints is to apply a penalty technique. In this work an hyperbolic penalty function presented by Xavier [1] is used to penalize the complementarity constraints ($-x_{1j}x_{2j} \geq 0,\ j = 1,\ldots,q$) in (2):

$$P(x,u,v) = f(x) + P_t \qquad \text{and} \qquad P_t = \sum_{j=1}^{q} u(x_{1j}x_{2j}) + \sqrt{u^2(x_{1j}x_{2j})^2 + v^2}$$

where P_t is the penalty term and u, v are parameters with $u, v \geq 0$, $u \to \infty, v \to 0$. The graphic representation of the penalty term, based on the geometric idea in Figure 1, is an hyperbole with asymptotes forming angles $(\pi - u)$ and 0 (zero) with the horizontal axis and having v as the intercept with ordinates axis. It is a two phase penalty approach:

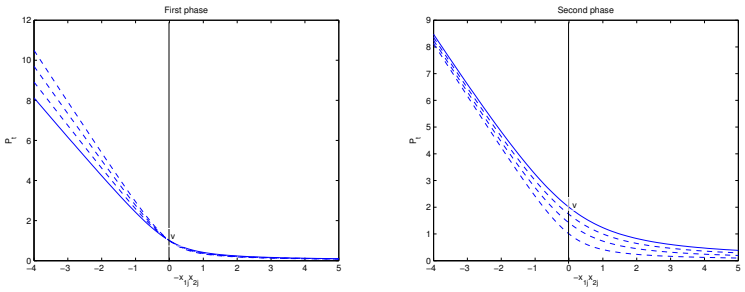

FIGURE 1. a) First phase. b) Second phase

in the first stage, the initial parameter u increases, thus causing a reduction in the penalty to the points outside the feasible region while at the same time there is a reduction in the penalty for the points inside the feasible region. This phase continues until a feasible point is obtained. From this point on, u remains constant and the values of v decrease sequentially.

Since the hyperbolic penalty algorithm is a dynamic penalty, the feasibility test carried out to x_k to update penalty parameters is:

$$\begin{array}{ll} \text{If} & x_{1j}x_{2j} > \dfrac{-v_k}{1000},\ j = 1,...,q \ \text{ Then} \\ & \quad v_{k+1} \leftarrow v_k \times \rho_2 \ \ (0 < \rho_2 < 1) \\ \text{Else} & \\ & \quad u_{k+1} \leftarrow u_k \times \rho_1, \ \ (\rho_1 > 1) \end{array}$$

In this approach, the complementarity terms are penalized and a sequence of the following nonlinear constrained optimization problem is solved as far u is incremented and v decreased:

$$\begin{array}{ll} min & P(x,u,v) \\ s.t. & c_i(x) = 0, i \in E, \\ & c_i(x) \geq 0, i \in I, \\ & x_1 \geq 0, \ \ x_2 \geq 0, \end{array} \tag{3}$$

NUMERICAL EXPERIMENTS

An algorithm, the hyperbolic penalty algorithm (A1), with two iterative procedures was implemented in MATLAB. The inner iterative procedure is performed by the `fmincon` routine from MATLAB optimization toobox, that uses the SQP method.

In order to compare the hyperbolic penalty function performance, another penalty algorithm (Algorithm A2) was also implemented. This algorithm is based on the well known sequential penalty technique [8], [9]. The penalty function is $P(x,r) = f(x) + r\sum_{j=1}^{q}(x_{1j}x_{2j})^2$ with $r > 0$, $r \to \infty$. A sequence of minimization problems is solved as far r is incremented. The strategy is similar to the Algorithm A2 - only the penalty functions change.

Algorithm A1: Hyperbolic Penalty Algorithm

Initialization: u_0,v_0;
Inner, external iterations and function evaluations counters: $it\_int = 0, f\_count = 0, k = 0$;
Tolerances definition: $v_{min}, u_{max}, k_{max}, \varepsilon_1, \varepsilon_2, \varepsilon_3$;
Problem information (amplfunc): x_0, lb, ub, cl, cu, cv; Problem dimension: n, m, p, q;
REPEAT
  Built penalty function $P(x_k, u_k, v_k)$ and constraints;
  Run the MATLAB function: $[x, f, lambda, output]$=`fmincon('function',....,'constraint');`
  Lagrange multipliers estimation ($\perp$);
  Feasibility test to update v or u;
  Update x, inner, external iterations and function evaluations counters:
  $x_{k+1} \leftarrow x, \quad it\_int \leftarrow it\_int + output.iterations, \quad f\_count \leftarrow f\_count + output.fcount$;
UNTIL Stop criterium ($v_{min}, u_{max}, k_{max}, \varepsilon_1, \varepsilon_2, \varepsilon_3$)

The initial parameters are $u_0 = 6$, $v_0 = 1$ and $r_0 = 1$ for the A1 and A2 algorithm, respectively. The Algorithm A1 uses, $\rho_1 = 10$ and $\rho_2 = 0.001$ to update the penalty parameters whereas the Algorithm A2 uses $\rho_1 = 10$. To estimate the complementarity constraints Lagrange multipliers the following formulas are used: $(\lambda_\perp) = -u + \dfrac{u^2 x_{1j}x_{2j}}{\sqrt{u^2(x_{1j}x_{2j})^2 + v^2}}$, $j = 1,...,q$ (Algorithm A1) and $(\lambda_\perp) = 2rx_{1j}x_{2j}$, $j = 1,...,q$ (Algorithm A2).

CONCLUSIONS

This section summarizes the numerical experiences using 75 AMPL test problems (A Modeling Language for Mathematical Programming) from MacMPEC [2]. On this set of problems the maximum number of variables and complementarity constraints are up to 200 and 100, respectively. In order to analyze the relative performance of algorithms, the performance profiles of Dolan and Moré [10] were used. The performance metrics are external iterations, internal iterations, function evaluations and Cpu time. The graphics of performance profiles are in Figures 2 and 3 using the

log scale. Both algorithms present very good accuracy and robustness. From the figures, it is clear that the Algorithm A2 has the highest probability of being the optimal solver for about 70% of the problems with respect to the internal iterations. On the other hand, concerning the external iterations, function evaluations and Cpu time, the Algorithm A1 has higher probability of being the optimal solver. These numerical experiences lead to the conclusion that combination of the SQP strategy and the penalty technique are effective to solve MPCC. The hyperbolic penalty represents a good alternative for solving MPCC with a good accuracy in the solution, when compared with the one provided from the MacMPEC set of problems.

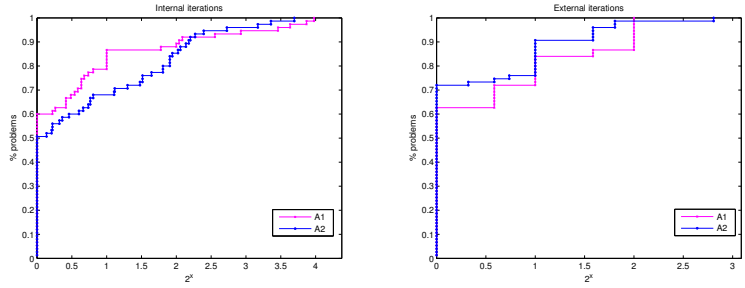

FIGURE 2. a) Internal iterations. b) External iterations.

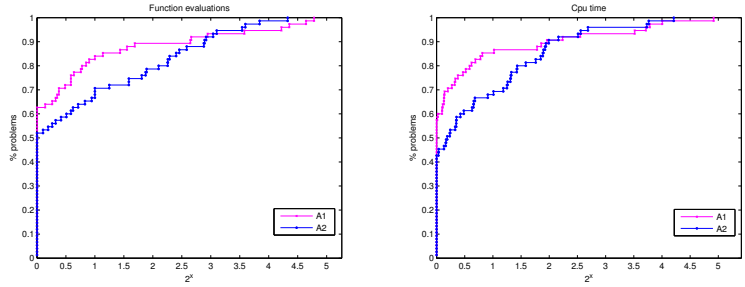

FIGURE 3. a) Function evaluations. b) Cpu time

REFERENCES

1. A. E. Xavier, *International Transactions in Operational Research* **8**, 659–671 (2001), ISSN 0969-6016, Hyperbolic penalty: A new method for nonlinear programming with inequalities.
2. S. Leyffer (2011), URL `http://wiki.mcs.anl.gov/leyffer/index.php/MacMPEC`.
3. H. Scheel, and S. Scholtes, *Mathematics of Operations Research* **25**, 1–22 (2000), Mathematical programs with complementarity constraints: Stationarity, optimality, and sensitivity.
4. S. Scholtes, *SIAM Journal on Optimization* **11**, 918 (2001), ISSN 10526234, Convergence properties of a regularization scheme for mathematical programs with complementarity constraints.
5. R. Fletcher, S. Leyffer, D. Ralph, and S. Scholtes, *SIAM Journal on Optimization* **17**, 259 (2006), ISSN 10526234, Local convergence of SQP methods for mathematical programs with equilibrium constraints.
6. X. M. Hu, and D. Ralph, *Journal of Optimization Theory and Applications* **123**, 365–390 (2004), ISSN 0022-3239, Convergence of a penalty method for mathematical programming with complementarity constraints.
7. M. Anitescu, P. Tseng, and S. J. Wright, *Mathematical Programming* **110**, 337–371 (2006), ISSN 0025-5610, Elastic-mode algorithms for mathematical programs with equilibrium constraints: global convergence and stationarity properties.
8. M. T. T. Monteiro, and J. F. Meira, *International Journal of Computer Mathematics* **88**, 145–149 (2011), ISSN 0020-7160, A penalty method and a regularization strategy to solve MPCC.
9. M. T. T. Monteiro, and H. S. Rodrigues, *Journal of Computational and Applied Mathematics* **235**, 5348–5356 (2011), ISSN 03770427, Combining the regularization strategy and the SQP to solve MPCC — a MATLAB implementation.
10. E. D. Dolan, and J. J. Moré, *Mathematical Programming* **91**, 201–213 (2002), ISSN 0025-5610, Benchmarking optimization software with performance profiles.