Solving Mathematical Programs with Complementarity Constraints with Nonlinear Solvers

Helena Sofia Rodrigues[1] and M. Teresa T. Monteiro[2]

[1] Computational Mathematic Master Student, University of Minho, Portugal
helena.rodrigues@ipb.pt
[2] Production and Systems Department,University of Minho, Portugal
tm@dps.uminho.pt

Summary. MPCC can be solved with specific MPCC codes or in its nonlinear equivalent formulation (NLP) using NLP solvers. Two NLP solvers - NPSOL and the line search filter SQP - are used to solve a collection of test problems in AMPL. Both are based on SQP (Sequential Quadratic Programming) philosophy but the second one uses a line search filter scheme.

1 Introduction

There has been an enormous amount of interest in developing algorithms to solve Mathematical Programs with Complementarity Constraints (MPCC). A series of problems that arise from realistic applications in engineering and economics is the main reason for such interest.

In recent years, the advances in computer technology have increased the capability of a modeler to solve large scale problems with complementarity constraints. On the one hand, the appearance of modelling systems allows to directly express complementarity conditions as part of their syntax and to pass on the complementarity model to the solver. On the other, the ability of a modeler to generate realistic large scale models enables the solvers to be tested on much larger and more difficult classes of models, producing in this way new enhancements and improvements in the solver.

However, solving MPCC is a harder task because it can be shown that constraint qualifications typically assumed to prove convergence of standard NLP algorithms fail for MPCC. As a result, applying specific MPCC solvers is problematic. To circumvent these problems, various reformulations of MPCC have been proposed. One of these approaches involves the possibility of solving MPCC by transforming it to a well-behaved nonlinear program. This endeavor is important because it allows to extend the body of analytical and computational expertise of nonlinear programming to this new class of problems.

In this work it is analyzed the possibility of solving MPCC in its NLP reformulation using certain SQP algorithms. Two NLP solvers based on SQP philosophy are used to solve a set of problems - NPSOL and the line search filter SQP. The last one is a promising recent algorithm developed by our research group still in improvement phase. Another goal of this study is testing this new algorithm for MPCC problems in their NLP reformulation. This study is included in a MSc project.

The organization of this paper is as follows. Section 2 introduces the Mathematical Programs with Complementarity Constraints. The next section defines an equivalent nonlinear program. In Section 4, the NLP solvers (NPSOL and line search filter SQP) are presented as well as their main characteristics. Numerical results obtained with a collection of AMPL test problems are presented in Section 5. Finally, the main conclusions are shown.

2 Mathematical Programs with Complementarity Constraints

A Mathematical Program with Complementarity Constraints is an optimization problem with equality and inequality constraints. In fact, it is a nonlinear optimization problem where the constraints have the same form as the first-order optimality conditions for a constrained optimization problem.

A Mathematical Program with Complementarity Constraint (MPCC) is defined as:

$$\begin{aligned} \min\ & f(z) \\ \text{s.t.}\ & c_E(z) = 0 \\ & c_I(z) \geq 0 \\ & 0 \geq z_1 \perp z_2 \leq 0 \end{aligned} \tag{1}$$

where $z = (z_0, z_1, z_2)$, $z_0 \in \mathbb{R}^n$ is the control variable and $z_1, z_2 \in \mathbb{R}^p$ are the state variables; c_E and c_I are the sets of equality and inequality constraints, respectively. The presence of complementarity constraints is the most prominent feature of a MPCC that distinguishes it from a standard nonlinear optimization problem.

To solve this kind of problems one might be tempted to use a standard nonlinear programming algorithm. Unfortunately, the feasible set of MPCC is ill-posed since the constraints qualifications which are commonly assumed to prove convergence of standard nonlinear programming algorithms do not hold at any feasible point of the complementarity constraints [16].

A number of special purpose algorithms have been developed for MPCCs, such as branch-and-bound, implicit nonsmooth approaches, piecewise sequential programming [4, 14]. Nevertheless, most of the algorithms for solving MPCC need strong assumptions to ensure convergence. Hence, the research on the development of effective algorithms remains vigourous.

This kind of problems gained lot of popularity in the last decade, because the concept of complementarity is synonymous of equilibrium and many real applications can be modelled by MPCC.

In Economics, complementarity is used to express Walsarian and Nash equilibrium, spatial price equilibria, invariant capital stock, game-theoretic models and Stackelberg lender-follower games, used in oligopolistic market analysis [3, 5]. More complex general equilibrium models are used for various aspects of policy design and analysis, including carbon abatement and trade form. Other applications use game theory, where new examples are becoming popular due to deregularization of electricity markets.

Engineering applications of MPCC include contact and structural mechanics, structural design, obstacle and free boundary problems, elastohydrodynamic lubrification and traffic equilibrium. Recently, MPCC has been used in optimal control problems for multiple robot systems [9, 15]. In optimization, this kind of problems involves the formulation of the Karush-Kuhn-Tucker conditions.

3 Nonlinear Programs

An extensive theory of first and second order optimality conditions for MPCC has been developed. However, the numerical analysis of large-scale MPCCs is still an area of investigation. Some recent papers have suggested reformulating the MPCC problem as a standard NLP. The idea behind this approach is to take advantage of certain NLP algorithms features in order to obtain rapid local convergence.

Notice that (1) can be written in the equivalent NLP form:

$$\begin{aligned} \min\ & f(z) \\ \text{s.t.}\ & c_E(z) = 0 \\ & c_I(z) \geq 0 \\ & z_1 \geq 0 \\ & z_2 \geq 0 \\ & z_1^T z_2 \leq 0 \end{aligned} \tag{2}$$

For the success of NLP solvers, Leyffer suggested to replace the usual complementarity condition $z_1^T z_2 = 0$ by the relaxed equivalent condition $z_1^T z_2 \leq 0$. Without this relaxation, several methods cannot converge quadratically near a strongly stationary point.

Unfortunately, the complementarity constraint implies that the KKT conditions are rarely satisfied by MPCC since it can be shown that there always exists a nonlinear abnormal multiplier [17]. Boundedness of the set of KKT multiplier vectors is equivalent to the Mangasarian Fromovitz constraint qualification condition arising in nonlinear programming.

Recall that, for a point z^* and active set $A(z^*) = E \cup \{i \in I \mid c_i(z) = 0\}$, Mangasarian Fromovitz Constraint Qualification (MFCQ) holds if there exists

a vector $w \in \mathbb{R}^n$ such that:

$$\nabla c_i(z^*)^T w > 0, \text{ for all } i \in A(z^*) \cap I$$
$$\nabla c_i(z^*)^T w = 0, \text{ for all } i \in E$$
$$\nabla c_i(z^*), i \in E \text{ are linearly independent}$$

In MPCC formulation all the feasible points are nonregular in the sense that they do not satisfy MFCQ, which is the usual condition for global convergence of a NLP algorithm. Nonregularity implies that the multiplier set is unbounded, that the normal vectors to active constraints are linearly dependent, and that the linearization of the NLP formulation can be inconsistent, arbitrarily close to a stationary point - all arguments against the use of the NLP technique for solving MPCCs.

Recent investigation brings good news: studies concluded that new well-established nonlinear programming solvers with minor modifications present exciting computational results.

4 NLP Solvers

Upon the success of SQP methods for nonlinear programming, the SQP approach has been extended to solve MPCC as well. In this work, it is presented two NLP solvers using SQP algorithms - NPSOL and the line search filter SQP.

4.1 NPSOL

NPSOL was created by Gill, Murray, Saunders and Wright [10]. NPSOL is a Fortran Package designed to solve the nonlinear programming problem: the minimization of a smooth nonlinear function subject to a set of constraints on the variables. The functions should be smooth but not necessarily convex. NPSOL employs a SQP algorithm and is specially effective for nonlinear problems whose functions and gradients are expensive to evaluate. The inner QP subproblem is solved by a LSSOL subroutine. An augmented Lagrangian merit function using a line search scheme promotes convergence from arbitrary starting points. The Hessian matrix of the Lagrangian function is updated with a BFGS quasi-Newton approximation.

4.2 Line Search Filter SQP

The line search filter SQP is a new algorithm for solving NLP problems, developed by Antunes and Monteiro [2] and still in improvement phase. It is based on a SQP algorithm with a filter scheme whose goal is to avoid the need of a merit function. This function requires difficult decisions in order to choose

the penalty parameters and handle other difficulties like nondifferentiability. We now proceed to briefly explain the filter scheme. For simplicity, consider the NLP problem written in the form:

$$\begin{cases} \min & f(x) \\ \text{s.t.} & c(x) \leq 0 \end{cases} \tag{3}$$

where $c : \mathbb{R}^n \to \mathbb{R}^m$ is a function incorporating all the constraints, possibly simple bound constraints.

Problem (1) can be reinterpreted as a problem which consists in minimizing simultaneously the objective function $f(x)$ and the term

$$h(c(x)) := \sum_{j=1}^{m} max\{0, c_j(x)\}$$

representing the sum of the constraints violation. A filter is a list of pairs $(f^{(i)}, h^{(i)})$ such that any pair dominates any other (dominance concept from multicriteria optimization [13]). A pair $(f^{(i)}, h^{(i)})$ obtained on iteration i is said to dominate another pair $(f^{(j)}, h^{(j)})$ if and only if both $f^{(i)} \leq f^{(j)}$ and $h^{(i)} \leq h^{(j)}$.

The line search filter SQP is based on a Fletcher and Leyffer idea [6] presented in 2000. While these authors used a trust region (TR) approach, the line search filter SQP uses a line search strategy to promote global convergence. The inner QP subproblem is solved using LSSOL subroutine from the NPSOL. For more details see [1, 2].

4.3 NPSOL *vs* Line Search Filter SQP

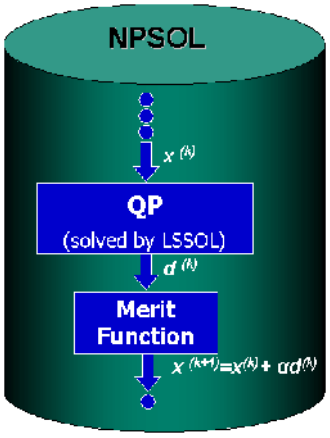

Fig. 1. Scheme of NPSOL

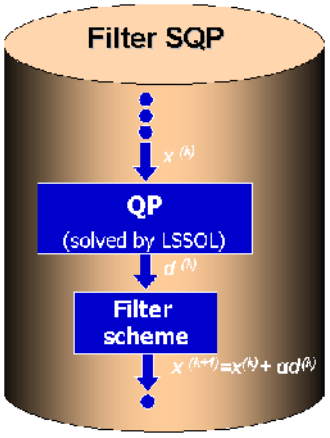

Fig. 2. Scheme of the line search filter SQP

As seen in Figures 1 and 2 for both codes the inner QP subproblem of the SQP algorithm is solved by LSSOL which is a subroutine of the NPSOL. The line search strategy is used also by the two solvers to promote global convergence. The only difference between these codes is the process to obtain the scalar α from the line search - NPSOL uses a merit function and the line search filter SQP consults the filter.

5 Numerical Examples

5.1 MacMPEC

In order to perform some computational experiments using these solvers, a library was used: MacMPEC [11], that is a collection of MPCC models written in the AMPL language [8]. It is a recent library compiled by Sven Leyffer that contains an extensive collection of MPCC problems. Not all the problems in MacMPEC are included in this study. Due to memory limit, it wasn't possible to solve large problems. The numerical tests were done on a Pentium IV 2600Mhz processor with 512Mb Ram in a WindowsXP operating system. More details about the problems and solvers results can be found in Appendix.

5.2 Numerical Results

For all the problems the usual complementarity condition in MPCC formulation (1) was replaced by the equivalent nonlinear condition with relaxation (1). All starting points are standard and fixed by default of AMPL. For both solvers, the stop criterium tolerance was $\epsilon = 1.0E - 06$ or 1000 iterations.

For some problems, the solvers couldn't confirm optimality, because the iteration limit was reached - Table 1 shows, for each solver, the number of the problems where this happened.

Table 1. Failures of NLP solvers

Solver	iter. limit
NPSOL	4
Line search filter SQP	10

Note that the tested problems set contains some problems that are known not to have strongly stationary limit points. For instance, ex9.2.2, and scholtes4 have solutions which are not strongly stationary. Problem gauvin has a global minimum at a point where the lower-level problems fails a constraint violation, so the formulation as MPCC is not appropriate [12]. Both SQP codes are very robust solving MPCC problems in their NLP formulation.

Figure 3 shows the comparison of the CPU time, in seconds. The NPSOL is significantly faster than the line search filter SQP but note that the last one is still in improvement phase.

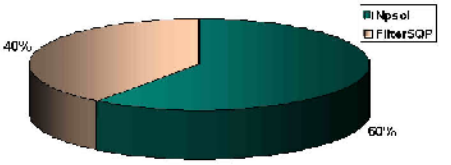

Fig. 3. Percentage of problems with lower CPU time

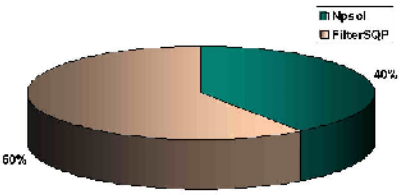

Fig. 4. Percentage of problems with fewer iterations

Figures 4 and 5 show the comparison in terms of iterations. With respect to the number of iterations the line search filter SQP takes advantage when compared with NPSOL - it needs less number of iterations in 60% of problems. It presents also fewer number of iterations used in a general way to solve problems.

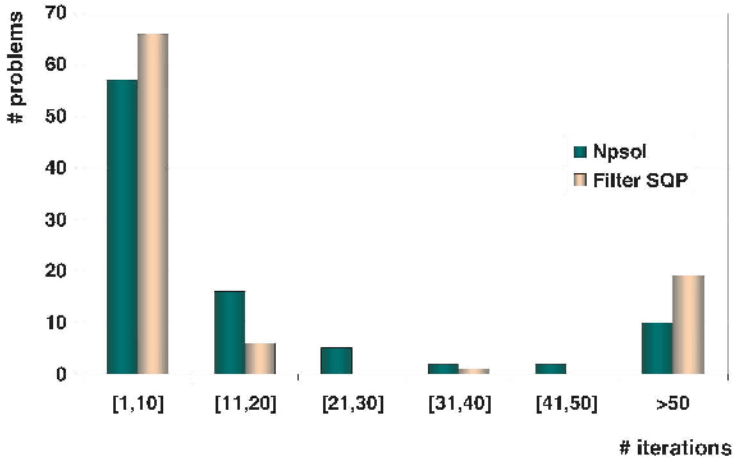

Fig. 5. Number of iterations

Figures 6 and 7 report the ranking of the number of function and gradient evaluations - in 60% of problems the line search filter SQP shows a best behaviour with respect to function and gradient evaluations.

6 Conclusions

A set of MPCC problems were reformulated as NLP problems using the relaxed complementarity condition. Two NLP solvers were tested with these problems and the results confirm their surprising robustness. Using NLP solvers based on SQP algorithms provide an ability to solve a great number of complex problems. The line search filter SQP presents better results than NPSOL with respect to the number of iterations and to the number of function and gradient evaluations. In terms of the CPU time, NPSOL is faster but recall that it is a commercial software already optimized, whereas the line search filter SQP is still in a development phase. The performance of the line search filter SQP, when compared with the NPSOL, is encouraging and this new code should continue to be improved in the future. The research on this area is very important for the modelers, hence it can be a technique for answering important economic and engineering questions.

Appendix - Detailed Numerical Results

Problem	CPU time		Number of iterations		Function Evaluations		Gradient Evaluations	
	Npsol	Filter SQP	Npsol	Filter SQP	Npsol	Filter SQP	Npsol	Filter SQP
bar-truss-3	0.031	0.015	7	9	25	10	25	24
bard1	0	0.031	6	3	8	4	8	4
bard2	0	0	5	2	6	3	6	3
bard3	1	0	1	2	2	3	2	3
bard1m	0	0	6	3	8	4	8	4
bard3m	0	0.015	1	52	2	212	2	56
bilevel1	0	0	3	2	7	3	7	3
bilevel2	0.015	0	5	4	8	5	8	5
bilevel3	0	0.031	13	93	13	673	13	94
bilin	0	0.031	1	104	2	685	2	105
dempe	0	0.015	50	2	173	22	173	3
design-cent-1	0	0.234	29	1000	61	19806	61	1001
design-cent-2	0	0	11	1	14	4	14	2
design-cent-4	0	0	6	2	7	3	7	3
desilva	0	0	2	19	4	27	4	20
df1	0	0	2	2	4	3	4	3
ex9.1.1	0	0	3	1	4	2	4	2
ex9.1.2	0	0.015	2	1	3	2	3	2
ex9.1.3	0	0.015	5	1	6	2	6	2
ex9.1.4	0	0	1	1	5	2	5	2
ex9.1.5	0	0	1	1	2	2	2	2
ex9.1.6	0	0	1	1	1	2	1	2
ex9.1.7	0	0	1	1	2	2	2	2
ex9.1.8	0	0	1	1	4	2	4	2
ex9.1.9	0.015	0	2	1	3	2	3	2
ex9.1.10	0	0	1	1	4	2	4	2
ex9.2.1	0	0	1	1	1	2	1	2
ex9.2.2	0	0.047	25	1	39	2	39	2
ex9.2.3	0	0	1	1	4	2	4	2
ex9.2.4	0	0	5	1	7	2	7	2
ex9.2.5	0.015	0	5	1	9	2	9	2
ex9.2.6	0	0	1	1	1	2	1	2
ex9.2.7	0	0	1	1	1	2	1	2
ex9.2.8	0	0	1	1	1	2	1	2
ex9.2.9	0	0	1	1	1	2	1	2
flp2	0	0	5	2	7	3	7	3
flp4-1	0.05	0.093	1	4	2	19	2	5
flp4-2	0.016	0.156	1	3	2	4	2	4
flp4-3	0.056	1.313	1	5	2	26	2	6
gauvin	0	0	3	8	6	9	6	9
gnash10	0.015	0	10	10	13	17	13	11
gnash11	0	0	10	6	13	7	13	7
gnash12	0	0	9	6	11	7	11	7
gnash13	0	0	10	6	13	7	13	7

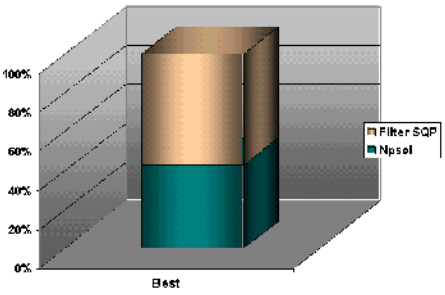

Fig. 6. Ranking of function evaluations

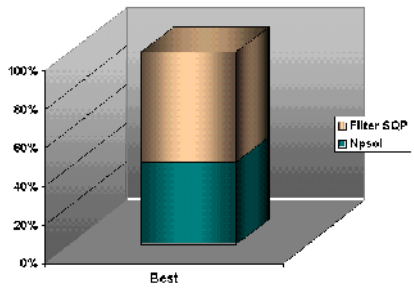

Fig. 7. Ranking of gradient evaluations

Problem	CPU time		Number of iterations		Function Evaluations		Gradient Evaluations	
	Npsol	Filter SQP	Npsol	Filter SQP	Npsol	Filter SQP	Npsol	Filter SQP
gnash14	0	0	10	7	13	8	13	8
gnash15	0	0	10	12	16	16	16	13
gnash16	0	0	10	8	13	12	13	9
gnash17	0	0	10	6	13	7	13	7
gnash18	0	0	14	9	21	14	21	10
gnash19	0	0	10	14	13	15	13	15
hakonsen	0	0.064	54	1	143	3	143	2
hs044-i	0	0.015	9	9	16	11	16	10
incid-set1-8	0.593	0.578	18	5	25	6	25	6
incid-set1c-8	0.562	0.547	14	5	21	6	21	6
incid-set2-8	23.156	151.297	1000	880	2948	16643	2948	881
incid-set2c-8	22.375	107.094	1000	362	2953	1539	2953	363
jr1	0	0	2	2	4	3	4	3
jr2	0	0	9	2	14	3	14	3
kth1	0	0	1	1	2	2	2	2
kth2	0	0	3	3	5	4	5	4
kth3	0	0	5	1	8	2	8	2
liswet1-050	0.14	0.546	6	2	9	3	9	3
nash1	0	0	12	2	16	3	16	3
outrata31	0	0.015	88	14	272	17	272	15
outrata32	0	0	16	8	19	16	19	9
outrata33	0	0	88	12	270	13	270	13
outrata34	0	0	62	36	168	39	168	37
portfl-i-1	0.078	0.171	16	6	17	25	17	7
portfl-i-2	0.046	0.281	12	8	13	38	13	9
portfl-i-3	0.109	0.156	18	6	21	24	21	7
portfl-i-4	0.062	0.328	12	13	14	48	14	14
portfl-i-6	0.062	0.218	13	8	14	22	14	9
qpec-100-1	0.0424	107.747	30	1000	31	8918	31	1001
qpec-100-2	0.609	127.062	24	1000	26	8163	26	1001
qpec-100-3	0.984	219.782	43	1000	44	17355	44	1001
qpec-100-4	0.89	24.125	33	109	34	367	34	110
qpec1	0.031	0.015	63	3	102	4	102	4
qpec2	0.015	0.828	2	1000	4	19830	4	1001
ralph1	0	0.62	31	1000	48	1001	48	1001
ralph2	0	0	20	2	21	3	21	3
ralphmod	3.375	35.985	103	143	148	1451	148	144
scholtes1	0	0	6	4	11	5	11	5
scholtes2	0	0	14	4	34	5	34	5
scholtes3	0	0.046	20	100	32	20001	32	1001
scholtes4	0	0.859	28	1000	41	1680	41	1001
scholtes5	0	0	3	3	4	4	4	4
sl1	0	0.047	4	2	14	3	14	3
stackelberg1	0	0	7	5	8	6	8	6
tap-09	0.062	39.344	6	1000	8	19967	8	1001
tap-15	71.641	330.953	1000	162	3008	3225	3008	163
water-net	0.062	7.828	20	1000	30	19841	30	1001
water-FL	39.953	239.812	1000	1000	2882	19701	2882	1001

References

1. A.S. Antunes and M.T. Monteiro. A SQP-filter algorithm with line search in nonlinear programming. Preprint, 2004.

2. A.S. Antunes and M.T. Monteiro. A filter algorithm and other NLP solvers: performance comparative analysis. Preprint, 2004.
3. S.C. Billups and K.G. Murty. Complementarity problems. JCAM inveted paper, 2000.
4. F. Facchinei, H. Jiang and L. Qi. A smoothing method for mathematical programs with equilibrium constraints. Tech. Rep. AMR 96/15, Univ. of New South Wales, 1996.
5. M.C. Ferris and C. Kanzow. Complementarity and related problems: a survey. In: P.M. Pardalos PM and M.G.C. Resende(eds), Handbook of Applied Optimization. Oxford Univ. Press, New York, 514–530, 2002.
6. R. Fletcher and S. Leyffer. Nonlinear Programming without a penalty function. Math. Programming 91:239–270, 2002.
7. R. Fletcher, S. Leyffer, and P.L. Toint. On the global convergence of a filter-SQP algorithm. SIAM J. Optim., 13(1): 44-59, 2002
8. R. Fourer, D.M. Gay, and B.W. Kernighan. AMPL: A Modelling Language for Mathematical Programming. Duxburg Press, 1993.
9. R. Fourer, M.C. Ferris, and G.M. Gay. Expressing complementary problems and communicating them to solvers. SIAM J. Optim., 9: 991-1009, 1999.
10. P.E. Gill, W. Murray, M.A. Saunders, and M.H. Wright. User's guide for NPSOL 5.0: a fortran package for nonlinear programming. Tech. Rep. SOL 86-1 , 1998.
11. S. Leyffer. MacMPEC, webpage: www.mcs.anl.gov/~leyffer/MacMPEC/, 2000.
12. S. Leyffer. Complementarity constraints as nonlinear equations: theory and numerical experience. Tech. Rep. NA/209, Univ. of Dundee, 2002.
13. K.M. Miettinen. Nonlinear Multiobjective Optimization. Kluwer Academic Publishers, 1999.
14. Z.Q. Luo, J.S. Pang, and D. Ralph. Mathematical Programs with Equilibrium Constraints, Cambridge University Press, 1996.
15. J. Peng, M. Anitescu, and S. Akella. Optimal control of multiple robot systems with friction using MPCC. Tech. Rep. W-31-109-ENG-38, 2003.
16. H. Pieper. Algorithms for mathematical programs with equilibrium constraints with applications to deregulated electricity markets. Dissertation, Stanford University, 2001.
17. J.J. Ye. Optimality conditions for optimization problems with complementarity constraints. SIAM J. Optim., 9(2):374-387, 1999.