Numerical experience with solving MPECs as NLPs*

Roger Fletcher, Sven Leyffer†

20 August 2002

Abstract

This paper describes numerical experience with solving MPECs as NLPs on a large collection of test problems. The key idea is to use off-the-shelf NLP solvers to tackle large instances of MPECs. It is shown that SQP methods are very well suited to solving MPECs and at present outperform Interior Point solvers both in terms of speed and reliability. All NLP solvers also compare very favourably to special MPEC solvers on tests published in the literature.

Keywords: MPEC, equilibrium constraints, nonlinear programming, SQP, interior point methods.

AMS-MSC2000: 90C30, 90C33, 90C55, 49M37, 65K10.

1 Introduction

We consider Mathematical Programs with Equilibrium Constraints (MPECs) of the form

$$\begin{array}{ll} \text{minimize} & f(z) \\ \text{subject to} & c_{\mathcal{E}}(z) = 0 \\ & c_{\mathcal{I}}(z) \geq 0 \\ & 0 \leq z_1 \ \perp \ z_2 \geq 0, \end{array} \tag{1.1}$$

where $z = (z_0, z_1, z_2)$, $z_0 \in \mathbb{R}^n$ are the control and $(z_1, z_2) \in \mathbb{R}^{2p}$ are the state variables. The equality constraints $c_i(z) = 0,\ i \in \mathcal{E}$ are abbreviated as $c_{\mathcal{E}}(z) = 0$ and similarly, $c_{\mathcal{I}}(z) \geq 0$ are the inequality constraints. Problems of this type arise frequently in applications, see [15, 24, 26] for references. Problem (1.1) is also referred to as a Mathematical Program with Complementarity Constraints (MPCC).

Clearly, any MPEC with a more general complementarity condition such as for instance $0 \leq y \ \perp \ F(x, y) \geq 0$, can be written in the form (1.1) by introducing slack variables. It is easy to show that the reformulated MPEC has the same properties (such as constraint qualifications or second order conditions) as the original MPEC. In this sense, nothing is lost by introducing slacks.

*Numerical Analysis Report NA/210, Department of Mathematics, University of Dundee.

†Department of Mathematics, University of Dundee, {fletcher,sleyffer}@maths.dundee.ac.uk

One interesting way of solving (1.1) is to consider its equivalent nonlinear programming NLP formulation,

$$\begin{array}{ll} \text{minimize} & f(z) \\ \text{subject to} & c_{\mathcal{E}}(z) = 0 \\ & c_{\mathcal{I}}(z) \geq 0 \\ & z_1 \geq 0 \\ & z_2 \geq 0 \\ & z_1^T z_2 \leq 0, \end{array} \tag{1.2}$$

and solve (1.2) with existing NLP solvers. This paper introduces a new collection of MPEC test problems and demonstrates that a large class of MPECs can in fact be solved successfully as NLPs.

This paper is organized as follows. The next section reviews recent theoretical advances which are relevant to the NLP approach as well as previous numerical experience. Section 3 describes a new ampl interface to filter that enables the SQP method to solve MPECs exploiting their special structure. Section 4 introduces a new test problem library with over 100 MPECs and describes some of their characteristics. Section 5 presents our numerical experience with some large scale NLP solvers and, finally, Section 6 sums up our experience and points to future developments.

2 MPEC Background

This section summarizes some background material that is relevant to the NLP approach to solving MPECs as (1.2). We also comment briefly on previous numerical experience.

2.1 Theoretical background to MPECs

It is easy to see that MPECs violate the Mangasarian Fromovitz Constraint Qualification (MFCQ), see [6, 27]. This failure of MFCQ implies that the multiplier set is unbounded and this has been taken as an argument against the use of NLP techniques for solving MPECs.

On the other hand, it is shown in [19], that strong stationarity [27] of (1.1) is equivalent to the existence of KKT multipliers for the equivalent NLP (1.2), i.e. the existence of multipliers $\mu := (\lambda, \nu_1, \nu_2, \xi)$ such that $\lambda_{\mathcal{I}} \geq 0$, $\nu_1, \nu_2 \geq 0$, $\xi \geq 0$ and

$$\begin{aligned} \nabla f(z) - \nabla c(z)^T \lambda - \begin{pmatrix} 0 \\ \nu_1 \\ 0 \end{pmatrix} - \begin{pmatrix} 0 \\ 0 \\ \nu_2 \end{pmatrix} + \xi \begin{pmatrix} 0 \\ z_2 \\ z_1 \end{pmatrix} &= 0 \\ c_{\mathcal{E}}(z) \geq 0 \text{ and } c_{\mathcal{I}}(z) &\geq 0 \\ z_1 \geq 0 \text{ and } z_2 &\geq 0 \\ z_1^T z_2 &\leq 0 \\ c_i(z)\lambda_i = 0,\ z_{1j}\nu_{1j} = 0,\ z_{2j}\nu_{2j} = 0,\ \xi z_1^T z_2 &= 0\ . \end{aligned} \tag{2.1}$$

Note, that the complementarity condition $\xi z_1^T z_2 = 0$ is implied by feasibility of z_1, z_2.

These KKT conditions also indicate, how the unboundedness of the multiplier set arises. For a given set of multipliers, define $\pi_1 = \nu_1 - \xi z_2$ and $\pi_2 = \nu_2 - \xi z_1$, then any other $\widehat{\nu_1}, \widehat{\nu_2}, \widehat{\xi} \geq 0$ that keeps (π_1, π_2) constant also satisfies these conditions. The multipliers $\pi_1,\ \pi_2$ are in fact the multipliers of the relaxed NLP, see [19].

It is shown in [19] that there exist a certain minimal multiplier vector, defined by

$$\xi = \max\left\{0\ ,\ \max_{i\in\mathcal{Z}_2^\perp}\frac{-\hat{\nu}_{1i}}{z^*_{2i}}\ ,\ \max_{i\in\mathcal{Z}_1^\perp}\frac{-\hat{\nu}_{2i}}{z^*_{1i}}\right\}. \tag{2.2}$$

The non-zero components this multiplier correspond to a linearly independent set of constraint normals (provided an MPEC-LICQ holds). Thus the multiplier set is a ray whose base is the minimal or basic multiplier vector. It can be shown that SQP methods converge quadratically to this basic multiplier, provided all QPs remain consistent. The assumption that all QPs remain consistent is shown to be satisfied for a certain class of MPECs with vertical complementarity constraints, such as

$$\begin{array}{ll}\text{minimize} & f(z)\\ \text{subject to} & 0 \le G(z)\ \perp\ H(z) \ge 0.\end{array}$$

The most important consequence of the failure of MFCQ is the fact that a QP approximation to the equivalent NLP (1.2) can be inconsistent arbitrarily close to a strongly stationary point. This has important implications for the design of SQP methods for MPECs. Anitescu [1] shows that an SQP method with elastic mode (such as `snopt` [21]) converges globally for MPECs. An alternative is to relax the linearization of the complementarity constraint and this idea is adopted here together with a restoration phase.

An important ingredient in the success of NLP solvers has been the replacement of the usual complementarity condition

$$z_1^T z_2 = 0 \ \text{ by the equivalent condition } \ z_1^T z_2 \le 0.$$

Without this relaxation, SQP methods cannot in general converge quadratically near a strongly stationary point. Consider for example

$$\left\{\begin{array}{ll}\text{minimize} & z_1 + z_2\\ \text{subject to} & 0 \le z_1\ \perp\ z_2 \ge 0\end{array}\right.$$

whose solution is at $z^* = (0,0)$. Writing the complementarity condition as $z_1 z_2 = 0$ and starting at any point with $z_1 = z_2$, e.g. $z^{(0)} = (1,1)$ results in SQP converging to z^* at linear rate (the iterates are $z^{(k)} = \frac{1}{2^k} z^{(0)}$). The reason for this slow convergence is that by forcing the iterates to lie on the linearization of $z_1 z_2 = 0$, complementarity is only reduced linearly at each iteration. The multiplier ξ diverges to infinity in this case. In contrast, using $z_1 z_2 \le 0$, gives convergence in 1 iteration, allowing SQP to identify the correct active set rapidly.

2.2 Previous numerical experience with MPECs

Numerical experience with (1.2) in the past has been rather disappointing. Bard [2] reports failure on 50 - 70 % of some bilevel problems for a gradient projection method. Conn et al. [7] and Ferris and Pang [15] attribute certain failures of lancelot for `LUBRIF` to the fact that the problem contains a complementarity constraint. However, examining the model it turns out that this model does not have a complementarity constraint. Instead, complementarity is minimized and is therefore not responsible for this failure. We believe that the failure of lancelot may be due to the fact that `LUBRIF` is (locally) inconsistent.

The most obvious approach to smoothing is to replace $z_1^T z_2 \leq 0$ by

$$z_1^T z_2 \leq \tau_k$$

and solve a sequence of NLPs, driving τ_k to zero, see Ferris and Kanzow [14]. This approach is also analysed in Scheel and Scholtes [27]. An alternative is to penalize the complementarity constraint, solving a sequence of NLPs where the objective is modified as

$$\text{minimize } f(z) + \nu_k z_1^T z_2$$

for a sequence of increasing penalty parameters $\nu_k > 0$. This approach is also related to Anitescu [1], who shows that under reasonable assumptions, a finite $\nu_k > 0$ is sufficient to give an exact penalization. Ferris and Tin Loi [16] conducted some numerical experiments comparing both approaches.

Another interesting idea due to Facchinei et.al. [13] is to replace the complementarity constraint by the smoothed min-function,

$$\psi_\mu(z_{1i}, z_{2i}) = \sqrt{(z_{1i} - z_{2i})^2 + 4\mu} - z_{1i} - z_{2i} = 0,$$

where $\mu > 0$ is a parameter. For $\mu = 0$, it can be shown that $\psi_0(a, b) = -2\min(a, b)$. The resulting NLP satisfies MFCQ and is differentiable with bounded gradients, as long as $(z_{1i}, z_{2i}, \mu_k) \neq 0$. In [13], various algorithms based on solving a sequence of NLPs for decreasing $\mu \to 0$ (inexactly) are suggested and their convergence properties are investigate.

The Penalty Interior Point Algorithm of [24] aims to maintain $z_1 > 0$ and $z_2 > 0$ whilst using an SQP approach to solving NLP (1.2). Unfortunately, PIPA can converge to non-stationary points under certain circumstances [23].

The implicit programming approach due to Zowe et. al [26] (see also Dirkse and Ferris [10]), solves the lower level complementarity problem for fixed controls and computes sensitivities of the controls. At the upper level, a nonsmooth optimization problem is then solved using bundle trust-region techniques.

Jiang and Ralph [22] propose two smooth SQP methods for solving MPECs. They replace the complementarity constraint by the smooth Fischer-Burmeister functional

$$\psi_\mu(z_{1i}, z_{2i}) = \sqrt{z_{1i}^2 + z_{2i}^2 + \mu} - z_{1i} - z_{2i} = 0,$$

and propose two methods for solving the resulting NLP. The first method applies SQP to the problem where μ is reduced at every iteration, while the second method add the equation $\exp(\mu) - 1 = 0$ to the NLP and then applies SQP. This equation ensures that in the limit, $\mu = 0$. In order to overcome the problems caused by infeasible QP subproblems, they apply a special ℓ_1 exact penalty function approach, similar to the elastic mode of snopt, which has been analysed in [1]. Fukushima, Luo and Pang [20] propose a similar algorithm, based on parametrically reducing μ at every iteration of SQP and report a numerical comparison to PIPA.

Recently, Dirkse, Ferris and Meeraus [11] have investigated automatic reformulation techniques for MPECs formulated in GAMS. They considered 23 (!) smooth reformulations and report numerical experience on a range of large MPECs.

3 filtermpec: an ampl solver interface

This section describes an Sequential Quadratic Programming (SQP) algorithm which has been adapted to solve MPECs. The SQP method used here is a trust-region SQP algorithm, filterSQP [17]. It uses a "filter" to promote global convergence. A filter accepts a trial point whenever the objective or the constraint violation is improved compared to all previous iterates. Much of the comments in this section would also apply to other generic SQP methods.

We have written an ampl interface for our NLP approach to MPECs. This interface takes the MPEC information generated by ampl and passes an equivalent NLP in form (1.2) to the NLP solver. The complementarity constraint which ampl presents to the solver is of the form

$$l_{y_i} \leq y_i \leq u_{y_i} \;\perp\; l_{F_i} \leq F_i(x,y) \leq u_{F_i}, \tag{3.1}$$

where exactly two bounds are finite. Note that ampl re-arranges more complex complementarity constraints in such a way that one side always involves only a simple variable bound.

If $l_{y_i} = u_{y_i}$ or $l_{F_i} = u_{F_i}$, then the complementarity condition (3.1) is replaced by an equality constraint and a free variable (or equation). Otherwise, slacks are introduced, so that (3.1) becomes

$$l_{y_i} \leq y_i \leq u_{y_i} \;\perp\; l_{F_i} \leq s_i \leq u_{F_i}\; s_i = F_i(x,y). \tag{3.2}$$

If $F_i(x,y)$ is also a simple bound constraint, then no slacks are needed and the interface does not introduce them in this situation.

The interface currently does not handle complementarity constraints of the form

$$y_i \;\perp\; l_{F_i} \leq F_i(x,y) \leq u_{F_i},$$

unless $l_{F_i} = u_{F_i}$ in which case an equation is detected. If $l_{F_i} < u_{F_i}$ then it is not clear, how best to model this type of complementarity condition within an NLP, using e.g. (3.3). Note that this situation can always be modeled with two complementarity conditions.

If $l_{y_i} > -\infty$ and $l_{F_i} > -\infty$, then the complementarity can be expressed as

$$(y_i - l_{y_i})(s_i - l_{F_i}) \leq 0. \tag{3.3}$$

Similar expressions are readily derived for all the other cases. The interface adds a single nonlinear equation, summing all complementarity conditions to the NLP. The modifications to the constraint, gradient and Hessian evaluation are relatively minor and the additional overhead is negligible, if a sparse matrix storage scheme is used (as in our case). The interface also contains some pre- and post-processing routines for setting up data structures and evaluating/printing the results. The whole package is available from the authors by request.

Of course, it would have been possible to solve MPECs formulated as NLP ampl models, i.e. without the `complements` command in ampl. However, the MPEC interface has several advantages.

1. The use of the `complements` structure makes the models more transparent and details such as the addition of slacks are hidden from the user.

2. All pre-solve facilities for complementarity problems are directly available, while they would not be available, if the NLP model were solved in ampl.

3. It extends NLP solvers to MPECs at negligible additional overhead. filtermpec works both as an NLP and as an MPEC solver.

It has been shown in [19], that linearizations of the complememtarity constraint can cause the QP approximations to (1.2) to become inconsistent arbitrarily close to a strongly stationary point. To avoid the difficulties associated with this, a simple heuristic has been proposed which perturbs the linearization of $z_1^T z_2 \leq 0$ to

$$z_1^{(k)^T} z_2^{(k)} + z_2^{(k)^T} d_1 + z_1^{(k)^T} d_2 \leq \delta \left(z_1^{(k)^T} z_2^{(k)} \right)^{1+\kappa}, \tag{3.4}$$

where $0 < \delta, \kappa < 1$ are constants that define the perturbation of the right hand side. Note, that for $\kappa = 0$, it is easy to find examples, where this perturbation would only give linear convergence. Properties of this perturbation are currently being investigated.

Additional solver options have been added to filtermpec to handle MPECs. These are

- `initial_slack`
  If set to 0 (default), then the slack variables in (3.2) are initialized to 0.
  If set to 1, then the slacks are initialized as

  $$s_i = \max \left\{ 0, F_i(x^{(0)}, y^{(0)}) \right\}$$

  where $(x^{(0)}, y^{(0)})$ is the initial point.

  Preliminary tests indicate no significant advantages of one option over the other, though initializing slacks in this way can be beneficial in some cases and is important part in hot-starting MPECs.

- `compl_tol`
  This is the value, μ, of the right hand side of $z_1^T z_2 \leq \mu$. This allows the solver to be used in smoothed mode for a decreasing sequence of μ. Its default value is `0.0E0`.

  This option is useful in finding initial points which satisfy the constraints apart from complementarity in case the NLP reformulation is inconsistent.

- `delta` is the value of δ in (3.4), default `1.0E-2`.

- `kappa` is the value of κ in (3.4), default `1.0`.

4 MPEC test problem library

This section describes an MPEC problem library for benchmarking MPEC and NLP solvers. All models are written in ampl and are available at

`www.maths.dundee.ac.uk/~sleyffer/MacMPEC/` .

The library also contains extensive comments and references for all models as well as plots of some of the solutions. Some of the models are novel and are described in more detail towards the end of the section.

Interfacing a solver (written for instance in C, Fortran or matlab) to ampl is relatively straightforward, see

`www.ampl.com/cm/cs/what/ampl/REFS/HOOKING/index.html` .

There are numerous examples and our own interface is available upon request. The solver can then either be invoked from the ampl prompt or directly from a command line without the need for ampl.

The motivation for collecting this test set is two-fold. Firstly, we are interested in testing our own ideas on a large collection of test problems. Secondly, we hope that the ready availability of this library of problems, motivates research into numerical aspects of MPECs and leads to the development of algorithm with both good theoretical properties and practical performance.

4.1 A classification scheme for MPECs

In order, to maintain a large collection of test problems and draw meaningful conclusions from numerical tests, it is important to suitably classify the test library. A standard NLP classification scheme has been extended to cover MPEC test problems. Each problem is identified by a string

`XXX-XX-CCC-n-m-p`

consisting of 3 groups of characters and 3 integers. The first two groups of characters describe the general aspect of the MPEC, the third group describes the aspect of the complementarity constraint and the integers represent the problem dimensions. Each group of characters and integers is now described in turn.

The first group of characters is identical to the scheme used in [4] and describes the type and regularity of the objective and constraints. The second group of characters describes the origin of the model (A = academic, M = modeling exercise and R = real application) and whether or not the lower level variables appear in the general constraint (Y = yes, N = no).

The third group of characters describes the type of the complementarity constraint and follows [3]. It is MCP for (general) Mixed Complementarity Problems, LMCP for Linear Mixed Complementarity Problems, NCP for Nonlinear Complementarity Problems, LCP for Linear Complementarity Problem and NLP if the lower level problem stems from optimality conditions of a nonlinear program. Finally, the integers `n-m-p` represent the number of variables, constraints (excluding complementarity constraints) and complementarity conditions.

Currently, the library contains 137 MPECs. The lower level complementarity constraint arise in various forms: 83 are LCPs, 15 are NCPs and 39 are KKT conditions of some lower level optimization problem. The problem sizes are summarized in Table 1 which shows the quantile distribution of the number of variables, constraints and complementarity constraints. Appendix A gives detailed characteristics of the MPEC test problems.

4.2 Description of selected models

There is a number of novel models in MacMPEC and these are described here.

Quantile	# variables	# constraints	# complementarities
1- 9	34	58	66
10- 99	54	34	29
100- 999	38	41	35
1000-9999	11	4	7

Table 1: Summary of MPEC test problem characteristics

Problems `hs044-i`, `liswet1-*` and `portfl*` are inverse QP problems which give rise to QPECs, following an idea by Stefan Scholtes. These problems are obtained by taking the solution to the original QP and then perturbing some of the QP data. This introduces controls. The inverse problem then arises by seeking the controls for which the solution of the perturbed QP is closest to the original solution. The equilibrium constraints are the complementary slackness conditions of the perturbed QP.

The models `water-*` are two small water distribution systems. The smaller model is derived from a Mixed Integer Nonlinear Programming GAMS model, which includes binary variables to model the pressure loss equation along each arc,

$$h_i - h_j = \frac{c_{ij}}{d_{ij}^p} q_{ij}^2 \text{sign}(q_{ij})$$

where the variables are h_i, the pressure at node i, q_{ij}, the flow along arc ij, d_{ij}, the diameter of the arc. The parameters are c_{ij} and p. This nonsmooth equation is usually modeled by introducing binary variables $z_{ij} \in \{0,1\}$ and splitting q_{ij} into its positive, $q_{ij}^+ \geq 0$ and negative, $q_{ij}^- \geq 0$ part, giving rise to the system

$$\begin{array}{rcl} z_{ij} & \in & \{0,1\} \\ q_{ij}^+ & \leq & M z_{ij} \\ q_{ij}^- & \leq & M(1 - z_{ij}) \end{array} \qquad h_i - h_j = \frac{c_{ij}}{d_{ij}^p} \left((q_{ij}^+)^2 + (q_{ij}^-)^2 \right),$$

where $M > 0$ is some large constant. This model can be found on

`www.gams.com/modlib/libhtml/waterx.htm` .

In `water-*`, the binary variables are replaced by a disjunction,

$$0 \leq q_{ij}^+ \perp q_{ij}^- \geq 0 \qquad h_i - h_j = \frac{c_{ij}}{d_{ij}^p} \left((q_{ij}^+)^2 + (q_{ij}^-)^2 \right).$$

The use of the disjunction removes the need for binary variables. The new model model `water-FL` is a larger network than the original model and represents the water distribution network of *Flatland*.

5 Numerical Results

This section presents the results of a numerical experiment in which a range of NLP solvers has been tested on a collection of MPECs formulated as NLPs (1.2). Details of the test problems can be found in Appendix A. We also compare the performance

of the NLP solvers to some MPEC solvers reported in the literature. In this section, we only summarize the results; detailed results for all solvers and runs can be found in Appendix B. The following large scale NLP solvers on NEOS [8] are included in the test.

Solver	Description & references
filtermpec	SQP trust-region method; global convergence through use of a filter; MPEC ampl interface; exact Hessian; feasibility restoration [17] and [18].
knitro	Trust-region interior point algorithm; step decomposed into normal and tangential step; special rules for trust-region adjustment [5].
loqo	Primal-dual interior-point method; line-search to induce global convergence; exact Hessian [28].
snopt	SQP line-search method; augmented Lagrangian merit function; quasi-Newton Hessian, inertia control; elastic mode for inconsistent QP subproblems.

5.1 Failures of NLP solvers

All NLP solvers make some kind of constraint qualification assumption in order to guarantee convergence. In the absence of an MFCQ, one can therefore not expect NLP solvers to converge. This section examines the type and nature of failures of the four solvers.

We consider the following two classes of failures of NLP solvers.

1. *Catastrophic failures* are runs, where a solver failed to converge within major 1000 iterations or the trust-region or step became too small at a non-stationary point. These are failures of the algorithm which are likely to be due to the absence of MFCQ.

2. *Soft failures* refer to instances, where a solver terminated with a message that the problem is locally inconsistent. Since all solvers are local methods, they cannot be expected to find a feasible points. This type of failure has (most likely) nothing to do with the absence of MFCQ.

Table 2 shows the number of catastrophic failures, locally inconsistent problems and local optimal solutions for each solver. In the context of evaluating the suitability of NLP solvers for solving MPECs, we view only the catastrophic failures as problematic.

	Number of failures	
Solver	catastrophic	infeasible
filtermpec	0	7
knitro	19	6
loqo	20	0
snopt	1	4

Table 2: Number and type of failures for NLP solvers (out of 137 problems)

There are surprisingly few catastrophic failures for the SQP methods, making these solvers very robust choices for solving MPECs. Even the number of termination at infeasible points is small. Note, in addition, that filtermpec can be shown to have found

local infeasibilities where the complementarity error $z_1^T z_2$ is relatively small (this can be seen in Appendix A).

On the other hand, interior point methods are not as robust as SQP methods for solving MPECs. Nevertheless, they solve over 85% of the problems, which is still significantly more than was observed in early tests [2] involving a projected gradient solver.

Previous numerical studies of MPECs and complementarity problems have sometimes made erroneous conclusions, attributing the failure of solvers to the failure of MFCQ. For instance, both [7] and [15] attribute lancelot's failure to solve `LUBRIF.SIF` to the fact that the problem is a complementarity problem, even though the formulation *minimizes* the complementarity subject to ordinary constraints.

It is therefore of interest, to investigate whether the failures of loqo and knitro are solely due to the fact that the problems violate MFCQ. In Tables 3 and 4, we give the reason for failure for each failed MPEC and also show how the solver performed when the complementarity constraint is replaced by

$$z_1^T z_2 \leq 10^{-2}.$$

This experiment will give an indication as to whether the failures are due to the absence of MFCQ or not.

problem	ξ	# iter	outcome
design-cent-4	2.383E-05	1000	iteration limit
incid-set1-32	-1.563E-13	121	optimal solution
incid-set1c-32	-8.783E+08	1000	iteration limit
incid-set2-16	-2.833E-09	81	optimal solution
incid-set2-32	-2.739E-11	151	optimal solution
incid-set2c-32	-4.121E+08	1000	iteration limit
pack-rig2-16	-1.070E-02	1000	iteration limit
pack-rig2-32	-1.152E-01	1000	iteration limit
pack-rig2c-16	-2.583E-02	1000	iteration limit
pack-rig2c-32	-8.469E-05	1000	iteration limit
pack-rig2p-32	-5.011E-278	435	dual infeasible
portfl1	-8.119E-09	45	optimal solution
portfl3	-5.628E-07	52	optimal solution
portfl4	-4.530E-08	53	optimal solution
ralphmod	-6.078E+10	1000	iteration limit
scholtes4	-1.000E+01	13	optimal solution
tollmpec	1.258E+07	1000	iteration limit
tollmpec1	-1.284E+07	1000	iteration limit
water-FL	-7.510E+10	1000	iteration limit
water-net	-4.079E+10	584	primal/dual infeasible

Table 3: Details of loqo failures after replacing complementarity by $z_1^T z_2 \leq 10^{-2}$

All loqo failures are due to it reaching the iteration limit (1000) on the original NLP formulation of the MPEC. After replacing the complementarity constraint by $z_1^T z_2 \leq 10^{-2}$, loqo still fails on 11 out of 20 NLP problems. Only in 7 instances, does loqo terminate with an optimal solution correctly.

The results of Table 3 show that loqo's failures are not solely due to the fact that the MPECs violate MFCQ. Instead, other factors clearly play a role. One such factor might be the difficulty in finding feasible points or terminating satisfactorily, if no such point exists.

knitro's failures on the MPEC set are also mostly due to reaching the iteration limit (1000). The two exceptions are `gnash14`, which terminates after trying to evaluate the objective outside the admissible range and `water-FL` which terminates with the trust-region radius becoming smaller than ten times the machine precision.

problem	ξ	# iter	outcome
dempe	3.325E-03	143	optimal solution
gnash14	-1.705E-08	52	optimal solution
incid-set1-16	1.843E-07	33	optimal solution
incid-set1-32	1.635E-07	56	optimal solution
incid-set1c-16	7.495E-08	32	optimal solution
incid-set1c-32	1.339E-07	20	optimal solution
incid-set2-16	-1.164E-07	81	optimal solution
incid-set2-32	-5.450E-07	124	optimal solution
incid-set2c-16	-2.692E-06	22	optimal solution
incid-set2c-32	-1.351E-05	28	optimal solution
pack-comp1-32	1.826E-05	23	optimal solution
pack-rig1-32	1.396E-04	34	optimal solution
pack-rig1p-32	1.390E-04	33	optimal solution
pack-rig2-16	1.764E-03	1000	iteration limit
pack-rig2-32	3.748E-04	1000	iteration limit
pack-rig2c-16	3.392E-04	1000	iteration limit
pack-rig2c-32	1.823E-03	1000	iteration limit
water-FL	-53.6972	37	trust-region got too small
water-net	-2.08034	1000	iteration limit

Table 4: Details of knitro failures after replacing complementarity by $z_1^T z_2 \leq 10^{-2}$

A special problem is `dempe`, given by

$$\begin{cases} \underset{x,w,z}{\text{minimize}} & (x-3.5)^2 + (z+4)^2 \\ \text{subject to} & z - 3 + 2zw = 0 \\ & 0 \leq x - z^2 \ \perp \ w \geq 0. \end{cases}$$

The solution to the lower level problem is

$$z(x) \ = \ \begin{cases} 3 & \text{for } x \geq 9 \\ \sqrt{x} & \text{for } 0 \leq x \leq 9 \\ \text{infeasible} & \text{for } x < 0. \end{cases}$$

Substituting $z(x)$ into the objective function, it can be shown that there exist two local minima at $x = 1$ and $x = 0$. However, for $x = 0$, the lower level problem, contains the constraint $z^2 \leq 0$, which violates MFCQ and there are no multipliers $w \geq 0$ such that

$z - 3 + 2zw = 0$ is satisfied. Thus, this limit is not even a B-stationary point. In this sense, `dempe` is a pathological example.

For this problem, knitro is the only solver that converges to the global minimum at $x = 0$. All other solvers converge to the local minimum at $x = 1$. The failure of knitro is a failure of the problem formulation, rather than of the method.

It is of interest to observe, that knitro manages to solve most of these NLPs 13 out of 19. The challenge for developers of interior point methods is to match the performance obtained without the complementarity constraint in the NLP formulation of the MPECs. Six problems still fail and we suspect this might be because the problems are locally inconsistent.

This section clearly shows, that interior point methods can experience difficulties in solving NLPs which are not related to the complementarity constraint.

Finally, the only failure of snopt (`incid-set1-16`) is solved successfully in 10 iterations and gives zero complementarity at the end.

5.2 Comparison of NLP solvers

This section summarizes the numerical comparison of different NLP solvers for solving MPECs. Detailed results can be found in Appendix B. The solvers are compared by using performance plots of Dolan and Moré [12]. For a solver s, we plot

$$\log_2\left(\frac{\#\text{ iter}(s,p)}{\text{best\_iter}(p)}\right)\ ,\ \forall p \in \text{ problem},$$

where $\#$ iter(s,p) is the number of iterations solver s took on problem p and best_iter(p) is the smallest number of iteration any solver took. This can be interpreted as a probability distribution that a given solver is at worse x times slower than the best. We also use the CPU time as a performance measure.

We use both number of iterations and CPU time as performance measures, since the former allows comparison between similar solvers excluding the effect of fast linear algebra. The CPU time comparison on the other hand gives an indication how different solver classes compare. Figures 1 and 2 show the performance plots for the iteration count and CPU time respectively.

For a catastrophic failure, we set $\#$ iter$(s,p) =$ `1E6` and CPU$(s,p) =$ `1E8`. This will ensure that the percentage of failures appears as a horizontal asymptotic at the right of the performance plots. Infeasibility failures are not penalized in this way, as they constitute ordinary exits from the NLP solver.

Both in terms of iteration count and CPU time, the SQP methods are clear winners over the interior point solvers. In terms of iteration count, filtermpec is somewhat faster than snopt. This may be due to the fact, that filtermpec is a second order method, while snopt is a quasi-Newton method. However, in terms of CPU time, snopt and filtermpec are roughly equivalent. We believe, that this is due to the faster linear algebra performance of snopt.

One reason for the poorer performance of interior point methods could be the fact that the multipliers are generally larger than for the SQP methods. In [19] it is shown that SQP methods converge locally, to a minimal multiplier. The same may not be true for interior point methods. In Figure 3, we show the relative size of the complementarity multiplier ξ.

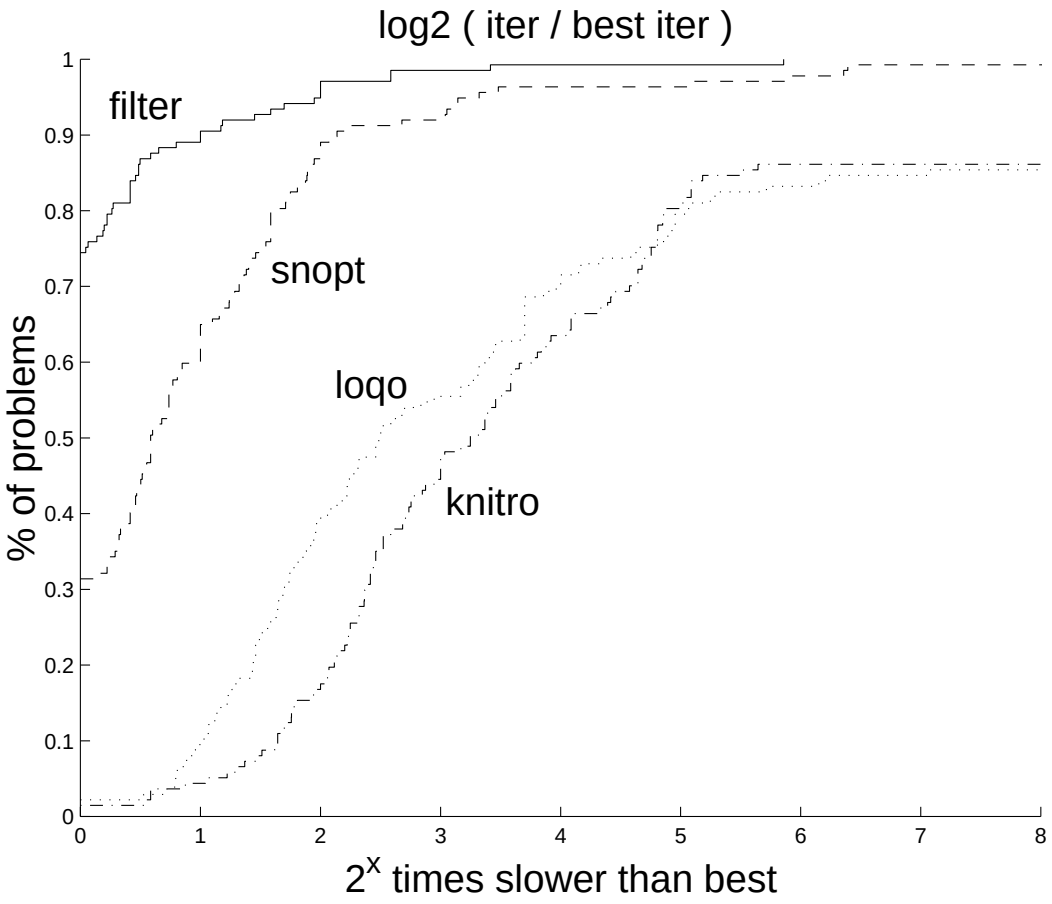

Figure 1: Performance plot (iteration)

Let $\xi(s,p)$ be the complementarity multiplier of solver s on problem p. Figure 3 is obtained by plotting for every solver s,

$$\log_{10}\left(\frac{\xi(s,p)}{\max\left(1,\min_s\left\{\xi(s,p)\right\}\right)}\right),\ \forall p \in \text{problem}$$

which is a measure of the size of the complementarity multiplier compared to the smallest complementarity multiplier (or 1). In this plot, we only include problems, where all solvers obtained the same solution and terminated successfully.

Figure 3 clearly indicates, that the multiplier generated by the interior point methods is several orders of magnitude larger than the multiplier generated by the SQP methods (which is minimal). It can be observed, that the curves on this plot correspond almost identically to the iteration performance plots of Figure 1.

One reason, why the size of the multipliers may matter is that interior point methods follow a central path, along which the product of slacks and multipliers is controlled by the barrier parameter $\mu > 0$. Large multipliers usually imply small slacks, resulting in slow progress. We are currently investigating techniques to improve the performance of interior point methods based on this observation.

Finally, we comment on the nature of the stationary points found by filtermpec. Considering ξ in Appendix A shows that ξ is only large ($\mathcal{O}(\epsilon^{-1})$) for `ex9.2.3`, `ralph1`,

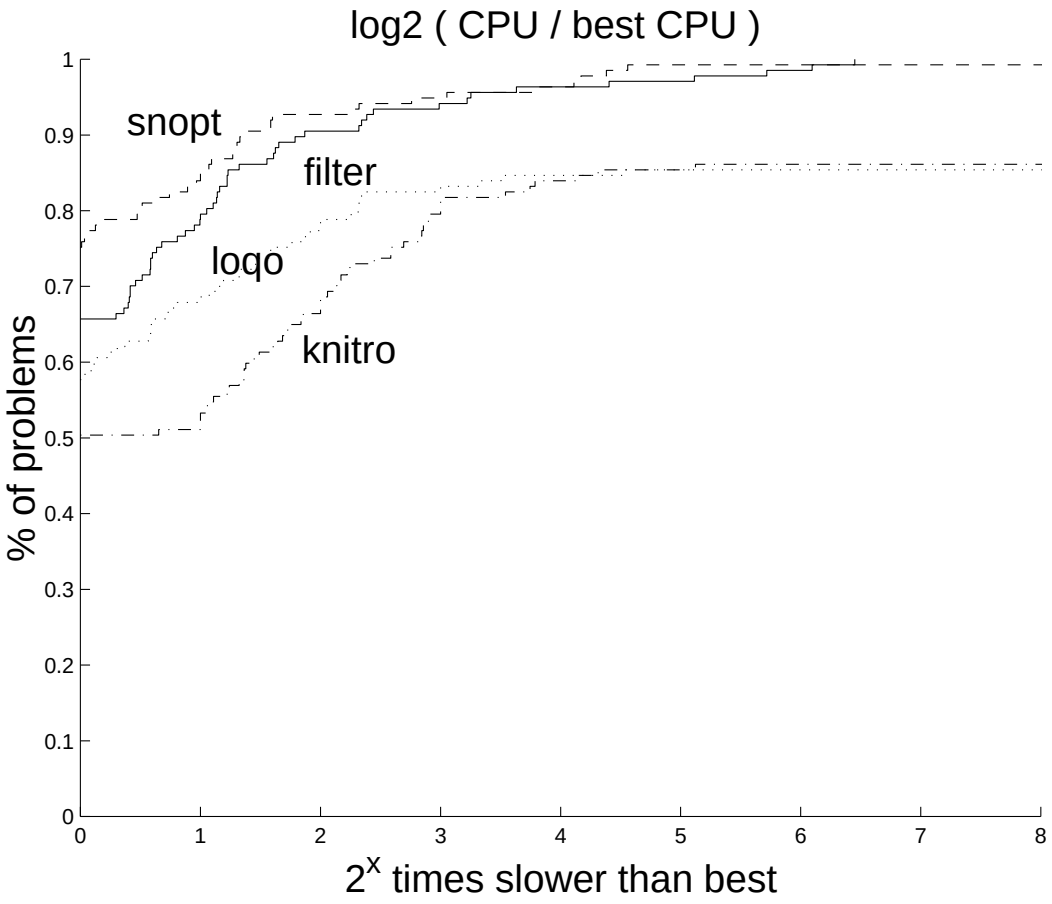

Figure 2: Performance plot (CPU time)

`ralphmod` and `scholtes5`. For all other problems, the limit is strongly stationary. Problems `ralph1` and `scholtes5` are known to have no strongly stationary points. In this case, filtermpec converges to B-stationary point and the rate of convergence can be seen to be linear. We suspect, that the same is true for `ralphmod` and `ex9.2.3`. It is remarkable that filtermpec does not converge to weaker (spurious) stationary points such as C-stationary points or M-stationary points.

5.3 Comparison to MPEC solvers

In this section, the results for the NLP solvers are compared to some results published in the literature for special purpose MPEC solvers. The aim is to give an indication of the potential of using NLP solvers. This comparison has to be tentative, as we have no direct access to the MPEC solvers and all the experiments reported in the literature were run under different circumstances. Nevertheless, we believe that this comparison offers some useful conclusions and insight into the efficiency of the NLP solvers.

The published numerical results considered in this section are the comparisons of [20], [13] and [9]. Tables 5 to 7 show the number of iterations required to solve the MPECs compared to the results published in the literature. Since all solvers (with the exception of the interior point solvers) perform roughly the same amount of work per iteration, this

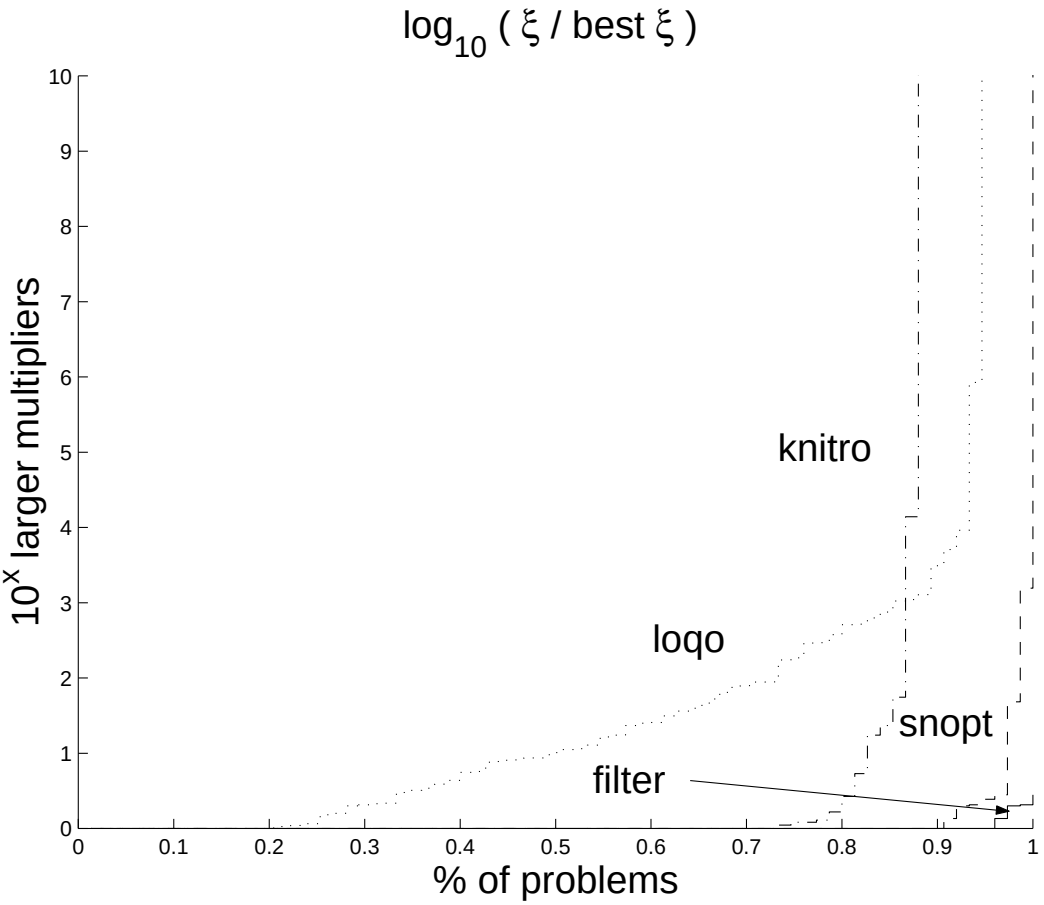

Figure 3: Relative size of complementarity multiplier ξ

comparison seems to be fair.

Fukushima, Luo and Pang [20] use a set of randomly generated problems (flp4-1 to flp4-4). The problems we have generated have the same structure and size as those proposed in [20]. Note, however, that it is unlikely that the actual instances are identical. The random problem generator (written in matlab) used to generate `flp4-*` is available online at

`www.maths.dundee.ac.uk/~sleyffer/MacMPEC/tools/genflp4.m` .

Since the random problems differ from [20], any conclusion based on these instances is by necessity tentative.

It is clear, however, from Table 5, that the SQP solver filtermpec compares very favourably with the MPEC method proposed in [20], often being an order of magnitude faster than the latter. Note also, that the interior point solver perform quite well on the problems. The multiplier ξ of the complementarity constraint is quite small for the interior point solvers. The results for snopt are not as encouraging when compared to the other solvers. It is not immediately clear, why this should be. The only significant difference to the other NLP solvers is that snopt is a quasi-Newton method. However, at the solution, the null-space of `flp4-*` is zero, so second order convergence should be observed.

The next set of test results is taken from [13], where the iteration counts refer to

#	problem	[20]	filtermpec	loqo	knitro	snopt
1	stackelberg1	22	4	14	37	3
2	flp2	48	5	29	27	13
3	bilevel2	31	1	18	50	10
4	flp4-1	50	3	19	17	66
4	flp4-2	50	4	21	16	88
4	flp4-3	51	4	20	16	96
4	flp4-4	50	4	23	20	161

Table 5: Iteration count for Fukushima et. al. [20]

algorithm AS. For the first 4 problems (`outrata31` to `outrata34`), we took starting point (a). For problem 9 (`nash1`), we have taken starting point (a). The number of iterations taken from [13] is "KIN", the number of inner iterations or QP solves.

#	problem	[13]	filtermpec	loqo	knitro	snopt
1	outrata31	27	8	18	38	9
2	outrata32	20	8	14	42	12
3	outrata33	17	7	19	38	9
4	outrata34	14	6	14	32	9
5	desilva	19	2	26	19	5
6	stackelberg1	5	4	14	37	3
7	bilevel1	28	6	15	36	5
8	gnash10	7	8	34	64	11
8	gnash11	7	8	31	55	11
8	gnash12	10	9	25	48	15
8	gnash13	7	13	24	44	18
8	gnash14	8	10	30	fail	17
8	gnash15	5	18	25	86	8
8	gnash16	6	16	22	38	8
8	gnash17	6	16	21	38	12
8	gnash18	18	14	23	31	12
8	gnash19	20	10	24	44	83
9	nash1	7	1	33	17	0
10	bilevel2	51	1	18	50	10
11	bilevel3	7	7	22	36	14

Table 6: Iteration count for Facchinei et. al. [13]

Again, it turns out that the SQP solvers in particular, are very competitive. Whilst they do not always perform better, than algorithm AS, especially filtermpec has a smaller variation in the number of iterations than AS. The results for loqo are also very competitive, especially, if one takes into account, that each iteration is far cheaper than either algorithm AS or SQP.

Dirkse and Ferris [9] test a total of 5 algorithms. In Table 7 we only reproduce results for the two best representative, BUNDLE and PIPA-B. [9] uses slightly different problem

names to ours. These differences are summarized below.

[9]	oz3	mss1	qvi	fjq*	hq1
here	bilevel3	gnash1*	nash1	outrata3*	stackelberg

problem	BUNDLE	PIPA-B	filtermpec	loqo	knitro	snopt
bard3	1	fail	4	22	48	9
desilva	9	145	2	26	19	5
outrata31	10	1000	8	18	38	9
outrata32	13	59	8	14	42	12
outrata33	8	35	7	19	38	9
outrata34	13	24	6	14	32	9
gauvin	13	18	7	20	29	10
stackelberg1	13	18	4	14	37	3
gnash10	14	29	8	34	64	11
gnash14	16	52	10	30	fail	17
gnash19	11	21	10	24	44	83
bilevel1	2	1000	6	15	36	5
nash1	11	1000	1	33	17	0
ralphmod	fail	1000	32	1000	109	3
tollmpec	7	memory	16	1000	381	39

Table 7: Iteration count for Dirkse and Ferris [9]

One surprise from the comparison in Table 7 is the fact that SQP methods appear to be more robust than the specially design MPEC solvers! BUNDLE has one failure which is solved in 32 iterations by filtermpec. snopt converges in only 3 iterations, but appears to get trapped at a non-stationary point or a local minimum (it cannot improve the current point).

PIPA-B has six failures, all of which are handled efficiently by the SQP methods. In addition, it is again clear, that the SQP methods are at least competitive when compared to the specially developed MPEC solvers.

All three comparisons, while tentative by design, point to the potential for NLP solvers when solving MPECs. In some cases, NLP solvers are actually more robust than the specially design MPEC solvers. Clearly, more comparisons are needed, as well as further development of MPEC solvers to achieve a similar degree of robustness and speed as the NLP methods.

6 Conclusions

We have presented a large collection of MPEC test problems and compared the performance of standard NLP solvers on these problems. In particular, SQP methods seem very robust at solving MPECs and their performance also compares favourably with special MPEC solvers on results published in the literature.

Interior point methods are less robust, failing on about 15 % of problems. The reasons for these failures may not be due to the complementarity constraint, as failures are also

observed, if the complementarity constraint is removed. However, we believe, that interior point methods can be made to solve MPECs robustly with small modifications to the code and we are currently investigating these ideas.

In our view, these test demonstrate that SQP solvers can solve MPECs efficiently and reliably. This underlines recent theoretical developments which have shown SQP methods to converge for MPECs (see [1] and [19]). In our view, SQP solvers provide a benchmark against which the efficiency and robustness of existing and future MPEC solvers should be measured.

Acknowledgements

This work was supported EPSRC grant GR/M59549. We are also grateful for the opportunity to using Argonne's computing resources in carrying out our comparisons. We are grateful to Stefan Scholtes, Danny Ralph and Michael Ferris for many fruitful discussions on MPECs.

Many individuals provided help and input for the problem library. We are grateful to David Gay for sharing his ampl expertise with us. Michal Kocvara provided help with the packaging problems. Francis Tin Loi supplied some challenging structural engineering problems. Finally, we have "borrowed" test problems from a variety of sources but most notably from mpeclib of Steven Dirkse.

References

[1] Anitescu, M. On solving Mathematical Programs with Complementarity Constraints as Nonlinear Programs. Preprint ANL/MCS-P864-1200, Mathematics and Computer Science Division, Argonne National Laboratory, Argonne, IL, USA, 2000.

[2] Bard, J.F. Convex two-level optimization. *Mathematical Programming*, 40(1):15–27, 1988.

[3] Billups,S.C., Dirkse, S.P. and Ferris, M.C. A comparison of large scale mixed complementarity problem solvers. *Computational Optimization and Applications*, 7:3–25, 1997.

[4] Bongartz, I. Conn, A.R. Gould, N.I.M. and Toint, Ph.L. CUTE: Constrained and Unconstrained Testing Enviroment. *ACM Transactions on Mathematical Software*, (21):123–160, 1995.

[5] Byrd, R.H. Hribar, M.E. and Nocedal, J. An interior point algorithm for large-scale nonlinear programming. *SIAM Journal on Optimization*, 9(4):877–900, 1999.

[6] Chen, Y. and Florian, M. The nonlinear bilevel programming problem: Formulations, regularity and optimality conditions. *Optimization*, 32:193–209, 1995.

[7] Conn, A.R. Gould, N.I.M. and Toint, Ph.L. Numerical experiments with the lancelot package (Release A) for large–scale nonlinear optimization. *Mathematical Programming*, 73(1):73–110, 1996.

[8] Czyzyk, J., Mesnier, M. and Moré, J. The NEOS server. *IEEE Journal on Computational Science and Engineering*, 5:68–75, 1998. Try it at `www-neos.mcs.anl.gov/neos/`.

[9] Dirkse, S.P. and Ferris, M.C. Modeling and solution environments for mpec: Gams & matlab. In M. Fukushima and L. Qi, editors, *Reformulation: Nonsmooth, Piecewise Smooth, Semismooth and Smoothing Methods*, pages 127–148. Kluwer Academic Publishers, 1999.

[10] Dirkse, S.P. and Ferris, M.C. Modelling and solution environments for MPEC: GAMS & MATLAB. In M. Fukushima and L. Qi, editors, *Reformulation: Nonsmooth, Piecewise Smooth, Semismooth and Smoothing Methods*. Kluwer Academic, Dordrecht, 1999.

[11] Dirkse, S.P., Ferris, M.C. and Meeraus, A. Mathematical programs with equilibrium constraints: Automatic reformulation and solution via constraint optimization. Technical Report NA-02/11, Oxford University Computing Laboratory, July 2002.

[12] Elizabeth D. Dolan and Jorge Moré. Benchmarking optimization software with cops. Technical Report ANL/MCS-246, MCS Division, Argonne National Laboratory, Argonne, IL, USA, November 2000.

[13] Facchinei, F. Jiang, H. and Qi, L. A smoothing method for mathematical programs with equilibrium constraints. Technical Report 03.96, Universita di Roma, "La Sapienza", March 1996.

[14] Ferris, M.C. and Kanzow, C. Complementarity and related problems: A survey. Mathematical Programming Technical Report 98-17, University of Wisconsin, August 1999.

[15] Ferris, M.C. and Pang, J.S. Engineering and economic applications of complementarity problems. *SIAM Review*, 39(4):669–713, 1997.

[16] Ferris, M.C. and Tin-Loi, F. Nonlinear programming approach for a class of inverse problems in elastoplasticity. *Structural Engineering and Mechanics*, 6:857–870, 1998.

[17] Fletcher, R. and Leyffer, S. Nonlinear programming without a penalty function. *Mathematical Programming*, 91(2):239–269, January 2002.

[18] Fletcher, R., Leyffer, S. and Toint, Ph.L. On the global convergence of an SLP-filter algorithm. Numerical Analysis Report NA/183, University of Dundee, UK, August 1998.

[19] Fletcher, R., Leyffer, S., Ralph, D. and Scholtes, S. Local convergence of SQP methods for mathematical programs with equilibrium constraints. Numerical Analysis Report NA/209, Department of Mathematics, University of Dundee, Dundee, DD1 4HN, UK, May 2002.

[20] Fukushima, M. , Luo, Z.-Q. and Pang, J.-S. A globally convergent sequential quadratic programming algorithm for mathematical programs with linear complementarity constraints. *Computational Optimization and Applications*, 10(1):5–34, 1998.

[21] Gill, P.E., Murray, W. and Saunders, M.A. SNOPT: An SQP algorithm for large–scale constrained optimization. Report NA 97-2, Dept. of Mathematics, University of California, San Diego, November 1997.

[22] Jiang, H. and Ralph, D. Smooth SQP methods for mathematical programs with nonlinear complementarity constraints. Manuscript, University of Melbourne, Department of Mathematics, December 1997.

[23] Leyffer, S. The penalty interior point method fails to converge for mathematical programs with equilibrium constraints. Numerical Analysis Report NA/208, Department of Mathematics, University of Dundee, `www.maths.dundee.ac.uk/~sleyffer/`, February 2002.

[24] Luo, Z.-Q., Pang, J.-S. and Ralph, D. *Mathematical Programs with Equilibrium Constraints*. Cambridge University Press, 1996.

[25] Luo, Z.-Q., Pang, J.-S., Ralph, D. and Wu, S.-Q. Exact penalization and stationarity conditions of mathematical programs with equilibrium constraints. *Mathematical Programming*, 75(1):19–76, 1996.

[26] Outrata, J., Kocvara, M. and Zowe, J. *Nonsmooth Approach to Optimization Problems with Equilibrium Constraints*. Kluwer Academic Publishers, Dordrecht, 1998.

[27] Scheel, H. and Scholtes, S. Mathematical program with complementarity constraints: Stationarity, optimality and sensitivity. *Mathematics of Operations Research*, 25:1–22, 2000.

[28] Vanderbei, R.J. and Shanno, D.F. An interior point algorithm for nonconvex nonlinear programming. *Computational Optimization and Applications*, 13:231–252, 1999.

A Test problem characteristics

This section gives details of the characteristics all MacMPEC test problems. In the tables below, n is the number of variables (excluding slacks), m is the number of general constraints (excluding simple bounds), p is the number of complementarity conditions), n_{NLP} is the number of variables plus slacks and k is the dimension of the null-space.

The next three columns give the degrees of degeneracy of the upper level (d_1), the lower level (d_2) problem and the degree of mixed degeneracy (d_m), see [25]. iter is the number of iterations by filtermpec, compl is the final complementarity error, i.e. $z_1^T z_2$ and ξ is the multiplier of the complementarity constraint. For the latter two columns, if no value is given, then the solution returned an exact zero.

name	n	m	p	n_{NLP}	k	d_1	d_2	d_m	iter	compl	ξ
bar-truss-3	35	34	6	35	0	13	0	0	14		-1.45
bard1	5	4	3	8	0	3	0	0	1		-0.762
bard1m	6	4	3	9	0	4	0	0	1		-0.762
bard2	12	9	3	15	0	2	1	0	2		
bard2m	12	9	3	15	0	2	1	0	1		
bard3	6	5	1	7	0	2	0	0	4		
bard3m	6	5	3	9	0	2	0	0	4		-1.09
bem-milanc30-s	3436	3433	1464	3436	2	1698	1	0	56		-119.
bilevel1	10	9	6	12	0	4	0	0	6		-0.150
bilevel2	16	13	8	20	1	5	0	1	1		
bilevel3	11	10	3	11	0	5	0	0	7		-1.09
bilin	8	7	6	14	0	3	0	1	4		-50.0
dempe	3	2	1	4	0	0	0	0	58		-0.571E-05
design-cent-1	12	11	3	15	0	6	0	0	4		-2.17
design-cent-2	13	15	3	16	1	6	0	0	11		-4.39
design-cent-3	15	11	3	18	0	10	0	1	24		
design-cent-4	22	20	8	30	1	12	0	1	4		-0.845
desilva	6	4	2	8	0	2	0	2	2		
df1	2	3	1	3	1	1	0	1	2		
ex9.1.1	13	12	5	13	0	7	0	2	3		-0.500
ex9.1.10	11	9	3	11	0	5	0	2	1		
ex9.1.2	8	7	2	8	0	4	0	0	1		
ex9.1.3	23	21	6	23	0	15	0	2	4		-34.0
ex9.1.4	8	7	2	8	0	3	0	1	4		
ex9.1.5	13	12	5	13	0	10	0	2	1		
ex9.1.6	14	13	6	14	0	7	0	0	3		-1.00
ex9.1.7	17	15	6	17	0	8	0	2	4		-34.0
ex9.1.8	11	9	3	11	0	5	0	2	1		
ex9.1.9	12	11	5	12	0	5	0	1	2		-0.444
ex9.2.1	10	9	4	10	0	3	1	0	6		-2.42
ex9.2.2	9	8	3	9	0	3	0	1	27	0.297E-12	-0.303E+07
ex9.2.3	14	13	4	14	0	5	1	0	2		
ex9.2.4	8	7	2	8	0	3	0	0	7		-1.000
ex9.2.5	8	7	3	8	0	3	0	0	8		-6.00
ex9.2.6	16	12	6	16	0	8	0	4	1		-0.500
ex9.2.7	10	9	4	10	0	3	1	0	6		-2.42
ex9.2.8	6	5	2	6	0	2	1	0	1		-0.500
ex9.2.9	9	8	3	9	0	7	0	1	1		

name	n	m	p	n_{NLP}	k	d_1	d_2	d_m	iter	compl	ξ
gauvin	3	2	2	5	0	1	0	0	7		-0.250
gnash10	13	12	8	13	1	0	0	0	8		-0.142
gnash11	13	12	8	13	1	0	0	0	8		-0.918E-01
gnash12	13	12	8	13	1	0	0	0	9		-0.397E-01
gnash13	13	12	8	13	1	0	0	0	13		-0.149E-01
gnash14	13	12	8	13	1	0	0	0	10		-0.199E-02
gnash15	13	12	8	13	0	3	0	0	18		-7.65
gnash16	13	12	8	13	0	4	0	0	16		-1.95
gnash17	13	12	8	13	1	4	0	0	16		-1.67
gnash18	13	12	8	13	1	4	0	0	14		-12.7
gnash19	13	12	8	13	0	2	0	0	10		-2.80
hakonsen	9	8	4	9	0	2	0	0	10		-0.390
hs044-i	20	14	10	26	0	7	0	2	6		-5.69
incid-set1-16	485	491	225	485	0	232	0	5	34		
incid-set1-32	1989	2003	961	1989	0	188	26	5	112		
incid-set1-8	117	119	49	117	0	52	0	1	40		
incid-set1c-16	485	506	225	485	1	250	0	13	24		
incid-set1c-32	1989	2034	961	1989	2	145	21	0	108		
incid-set1c-8	117	126	49	117	0	64	0	4	41		
incid-set2-16	485	491	225	710	3	214	11	0	26		
incid-set2-32	1989	2003	961	2950	5	940	24	0	69		
incid-set2-8	117	119	49	166	3	42	7	0	39		
incid-set2c-16	485	506	225	710	0	219	12	0	36		
incid-set2c-32	1989	2034	961	2950	1	937	24	0	66		
incid-set2c-8	117	126	49	166	2	46	6	0	33		
jr1	2	1	1	3	1	0	0	0	1		
jr2	2	1	1	3	0	0	0	0	6		-2.00
kth1	2	1	1	2	0	0	1	0	1		
kth2	2	1	1	2	1	0	0	0	2		
kth3	2	1	1	2	0	0	0	0	4		-1.00
liswet1-050	152	103	50	202	1	52	0	0	1		
liswet1-100	302	203	100	402	1	102	0	0	1		
liswet1-200	602	403	200	802	1	202	0	0	1		
nash1	6	4	2	8	2	4	0	0	1	0.153E-08	
outrata31	5	4	4	9	0	0	1	0	8		-0.164
outrata32	5	4	4	9	1	0	0	0	8		-0.168
outrata33	5	4	4	9	1	1	0	0	7		-0.714
outrata34	5	4	4	9	1	1	0	0	6		-2.07
pack-comp1-16	467	511	225	692	4	268	0	2	8		
pack-comp1-32	1955	2107	961	2916	6	1063	0	3	49		
pack-comp1-8	107	121	49	156	0	113	0	0	6	0.728E-06	
pack-comp1c-16	467	526	225	692	1	267	0	1	5		
pack-comp1c-32	1955	2138	961	2916	3	1062	0	2	7		
pack-comp1c-8	107	128	49	156	0	120	0	0	6	0.728E-06	
pack-comp2-16	467	511	225	692	4	268	0	2	33		
pack-comp2-32	1955	2107	961	2916	6	1066	2	2	36		
pack-comp2-8	107	121	49	156	5	62	0	2	9		
pack-comp2c-16	467	526	225	692	4	268	0	2	13		
pack-comp2c-32	1955	2138	961	2916	13	1060	0	2	15		
pack-comp2c-8	107	128	49	156	1	62	0	2	6		

name	n	m	p	n_{NLP}	k	d_1	d_2	d_m	iter	compl	ξ
pack-rig1-16	380	379	158	485	6	207	0	0	16		
pack-rig1-32	1622	1621	708	2087	4	789	0	0	58		-12.6
pack-rig1-8	87	86	32	108	7	48	0	0	12		
pack-rig1c-16	380	394	158	485	4	205	0	0	7	0.390E-06	
pack-rig1c-32	1622	1652	708	2087	0	924	0	0	22	0.310E-06	
pack-rig1c-8	87	93	32	108	4	46	0	0	8	0.153E-07	
pack-rig1p-16	445	444	203	550	3	231	2	0	23		
pack-rig1p-32	1855	1854	861	2320	4	868	2	0	135		-12.6
pack-rig1p-8	105	104	47	126	3	50	2	0	14		
pack-rig2-16	375	374	149	480	0	207	0	0	8	0.706E-06	
pack-rig2-32	1580	1579	661	2045	0	893	0	0	14	0.482E-06	
pack-rig2-8	85	84	30	106	6	43	0	0	12		
pack-rig2c-16	375	389	149	480	0	221	0	0	7	0.379E-06	
pack-rig2c-32	1580	1610	661	2045	0	928	0	0	11	0.399E-06	
pack-rig2c-8	85	91	30	106	2	43	0	0	6	0.477E-08	
pack-rig2p-16	436	435	194	541	1	213	1	0	15		
pack-rig2p-32	1808	1807	814	2273	2	942	1	0	28		
pack-rig2p-8	103	102	45	124	3	45	2	0	9	0.369E-13	
portfl1	87	25	12	87	6	6	0	0	6		-0.896
portfl2	87	25	12	87	5	7	0	0	5		-0.682
portfl3	87	25	12	87	12	6	0	0	4		-31.1
portfl4	87	25	12	87	15	5	0	0	4		-116.
portfl6	87	25	12	87	13	5	0	0	4		-54.4
qpec-100-1	105	102	100	205	0	75	2	0	5		-4.77
qpec-100-2	110	102	100	210	0	57	3	0	6		-2.12
qpec-100-3	110	104	100	210	0	33	2	0	9		-125.
qpec-100-4	120	104	100	220	0	62	6	0	5		-3.66
qpec-200-1	210	204	200	410	0	153	2	0	10		-158.
qpec-200-2	220	204	200	420	0	115	4	0	10		-5.36
qpec-200-3	220	208	200	420	0	48	6	0	12		-35.5
qpec-200-4	240	208	200	440	0	135	10	0	6		-4.96
qpec1	30	20	20	40	0	10	10	0	3		
qpec2	30	20	20	40	0	0	10	0	2		-0.667
ralph1	2	1	1	3	0	0	0	0	27	0.383E-12	-0.538E+06
ralph2	2	1	1	2	1	0	0	0	11	0.240E-06	-2.00
ralphmod	104	100	100	204	1	78	0	0	32		-0.136E+06
scholtes1	3	1	1	4	0	1	0	0	4		
scholtes2	3	1	1	4	0	0	1	0	2		
scholtes3	2	1	1	2	0	0	0	0	4		-1.00
scholtes4	3	3	1	3	0	0	0	0	25	0.609E-13	-0.270E+07
scholtes5	3	2	2	3	2	0	0	0	1		
sl1	8	5	3	11	1	5	0	2	1		
stackelberg1	3	2	1	3	1	0	0	0	4		
tap-09	86	68	32	118	0	67	0	7	29		-40.0
tap-15	194	167	83	277	0	164	0	37	21		-91.9
tollmpec	2403	2376	1748	4151	0	1588	1	86	16		-8.46
tollmpec1	2403	2376	1748	4151	0	1608	0	85	16		-165.
water-FL	213	160	44	213	4	53	0	0	308		-0.174E+05
water-net	66	50	14	66	3	15	0	0	83		

B Detailed results for MacMPEC

B.1 Comparison of iteration counts

Name	filtermpec	loqo	knitro	snopt
bar-truss-3	14	27	120	10
bard1	1	13	25	0
bard1m	1	13	27	4
bard2	2	32	34	4
bard2m	1	28	34	4
bard3	4	22	48	9
bard3m	4	23	66	10
bem-milanc30-s	56	556	139	479
bilevel1	6	15	36	5
bilevel2	1	18	50	10
bilevel3	7	22	36	14
bilin	4	71	48	4
dempe	58	16	1000	0
design-cent-1	4	14	24	5
design-cent-2	11	26	25	8
design-cent-3	24	83	35	17
design-cent-4	4	1000	23	5
desilva	2	26	19	5
df1	2	29	19	7
ex9.1.1	3	15	33	3
ex9.1.10	1	31	34	3
ex9.1.2	1	13	28	3
ex9.1.3	4	39	25	0
ex9.1.4	4	38	31	3
ex9.1.5	1	11	17	0
ex9.1.6	3	14	31	3
ex9.1.7	4	74	26	0
ex9.1.8	1	31	34	3
ex9.1.9	2	20	27	4
ex9.2.1	6	13	29	0
ex9.2.2	27	279	77	33
ex9.2.3	2	38	46	0
ex9.2.4	7	11	19	5
ex9.2.5	8	16	25	8
ex9.2.6	1	25	21	1
ex9.2.7	6	13	29	0
ex9.2.8	1	11	25	0
ex9.2.9	1	13	28	2
gauvin	7	20	29	10
gnash10	8	34	64	11
gnash11	8	31	55	11
gnash12	9	25	48	15
gnash13	13	24	44	18
gnash14	10	30	1234	17
gnash15	18	25	86	8
gnash16	16	22	38	8
gnash17	16	21	38	12
gnash18	14	23	31	12
gnash19	10	24	44	83

Name	filtermpec	loqo	knitro	snopt
hakonsen	10	21	50	64
hs044-i	6	25	33	28
incid-set1-16	34	49	1000	1000
incid-set1-32	112	1000	1000	990
incid-set1-8	40	23	462	35
incid-set1c-16	24	78	1000	70
incid-set1c-32	108	1000	1000	255
incid-set1c-8	41	26	384	15
incid-set2-16	26	1000	1000	85
incid-set2-32	69	1000	1000	66
incid-set2-8	39	34	981	58
incid-set2c-16	36	102	1000	36
incid-set2c-32	66	1000	1000	172
incid-set2c-8	33	32	209	46
jr1	1	9	9	3
jr2	6	11	9	7
kth1	1	9	8	1
kth2	2	11	7	4
kth3	4	10	22	6
liswet1-050	1	20	28	59
liswet1-100	1	24	32	84
liswet1-200	1	31	27	82
nash1	1	33	17	0
outrata31	8	18	38	9
outrata32	8	14	42	12
outrata33	7	19	38	9
outrata34	6	14	32	9
pack-comp1-16	8	52	91	22
pack-comp1-32	49	101	1000	43
pack-comp1-8	6	43	27	4
pack-comp1c-16	5	78	70	9
pack-comp1c-32	7	54	106	18
pack-comp1c-8	6	34	40	8
pack-comp2-16	33	69	705	78
pack-comp2-32	36	97	207	121
pack-comp2-8	9	36	42	15
pack-comp2c-16	13	47	67	29
pack-comp2c-32	15	68	69	24
pack-comp2c-8	6	23	40	7
pack-rig1-16	16	59	70	43
pack-rig1-32	58	79	1000	48
pack-rig1-8	12	40	22	17
pack-rig1c-16	7	50	45	27
pack-rig1c-32	22	104	88	30
pack-rig1c-8	8	25	26	12
pack-rig1p-16	23	50	79	29
pack-rig1p-32	135	68	1000	35
pack-rig1p-8	14	34	43	14

Name	filtermpec	loqo	knitro	snopt
pack-rig2-16	8	1000	1000	6
pack-rig2-32	14	1000	1000	37
pack-rig2-8	12	33	28	16
pack-rig2c-16	7	1000	1000	6
pack-rig2c-32	11	1000	1000	7
pack-rig2c-8	6	28	31	10
pack-rig2p-16	15	55	58	22
pack-rig2p-32	28	1000	401	28
pack-rig2p-8	9	35	25	15
portfl1	6	1000	26	7
portfl2	5	169	21	7
portfl3	4	1000	20	8
portfl4	4	1000	20	7
portfl6	4	210	23	7
qpec-100-1	5	30	41	21
qpec-100-2	6	34	66	67
qpec-100-3	9	28	48	74
qpec-100-4	5	32	63	22
qpec-200-1	10	44	47	37
qpec-200-2	10	39	77	28
qpec-200-3	12	41	50	24
qpec-200-4	6	58	89	24
qpec1	3	10	22	5
qpec2	2	266	44	68
ralph1	27	271	9	33
ralph2	11	21	15	10
ralphmod	32	1000	109	3
scholtes1	4	20	42	5
scholtes2	2	11	16	3
scholtes3	4	29	10	0
scholtes4	25	1000	11	36
scholtes5	1	10	8	3
sl1	1	71	33	3
stackelberg1	4	14	37	3
tap-09	29	50	42	44
tap-15	21	57	71	45
tollmpec	16	1000	381	39
tollmpec1	16	1000	449	60
water-FL	308	1000	37	95
water-net	83	1000	1000	308

B.2 Comparison of CPU times

Name	filtermpec	loqo	knitro	snopt
bar-truss-3	0.01	0.01	0.04	0.00
bard1	0.00	0.00	0.00	0.00
bard1m	0.00	0.00	0.01	0.00
bard2	0.00	0.00	0.01	0.00
bard2m	0.00	0.00	0.01	0.00
bard3	0.00	0.00	0.00	0.00
bard3m	0.00	0.00	0.01	0.00
bem-milanc30-s	409.48	72.89	43.89	604.00
bilevel1	0.00	0.00	0.01	0.01
bilevel2	0.00	0.00	0.01	0.00
bilevel3	0.01	0.00	0.00	0.00
bilin	0.00	0.01	0.01	0.00
dempe	0.00	0.00	fail	0.00
design-cent-1	0.00	0.00	0.00	0.00
design-cent-2	0.01	0.00	0.00	0.00
design-cent-3	0.01	0.04	0.01	0.01
design-cent-4	0.00	fail	0.01	0.00
desilva	0.00	0.00	0.01	0.00
df1	0.00	0.00	0.01	0.00
ex9.1.1	0.00	0.00	0.01	0.00
ex9.1.10	0.00	0.00	0.01	0.00
ex9.1.2	0.00	0.00	0.01	0.00
ex9.1.3	0.00	0.01	0.01	0.00
ex9.1.4	0.00	0.00	0.00	0.00
ex9.1.5	0.00	0.00	0.00	0.00
ex9.1.6	0.00	0.00	0.00	0.00
ex9.1.7	0.00	0.01	0.01	0.00
ex9.1.8	0.00	0.00	0.01	0.00
ex9.1.9	0.00	0.00	0.00	0.00
ex9.2.1	0.00	0.00	0.01	0.00
ex9.2.2	0.01	0.03	0.01	0.00
ex9.2.3	0.00	0.01	0.01	0.00
ex9.2.4	0.00	0.00	0.00	0.00
ex9.2.5	0.00	0.00	0.01	0.01
ex9.2.6	0.00	0.00	0.00	0.00
ex9.2.7	0.00	0.00	0.00	0.00
ex9.2.8	0.00	0.00	0.00	0.00
ex9.2.9	0.00	0.00	0.00	0.00
gauvin	0.00	0.00	0.00	0.00
gnash10	0.00	0.00	0.02	0.00
gnash11	0.00	0.01	0.01	0.00
gnash12	0.00	0.00	0.01	0.01
gnash13	0.00	0.00	0.01	0.01
gnash14	0.00	0.00	fail	0.01
gnash15	0.01	0.00	0.02	0.00
gnash16	0.00	0.00	0.01	0.00
gnash17	0.01	0.00	0.01	0.01
gnash18	0.00	0.00	0.01	0.01
gnash19	0.00	0.00	0.01	0.03

Name	filtermpec	loqo	knitro	snopt
hakonsen	0.00	0.00	0.00	0.01
hs044-i	0.00	0.01	0.01	0.01
incid-set1-16	5.75	2.62	fail	fail
incid-set1-32	429.25	fail	fail	1078.66
incid-set1-8	0.61	0.20	2.76	0.42
incid-set1c-16	4.79	3.91	fail	3.90
incid-set1c-32	613.12	fail	fail	208.71
incid-set1c-8	0.71	0.21	2.44	0.14
incid-set2-16	5.35	fail	fail	4.06
incid-set2-32	578.84	fail	fail	60.74
incid-set2-8	0.56	0.28	4.77	0.24
incid-set2c-16	9.35	5.31	fail	1.79
incid-set2c-32	420.87	fail	fail	115.22
incid-set2c-8	0.44	0.26	1.31	0.24
jr1	0.00	0.00	0.00	0.00
jr2	0.00	0.00	0.00	0.00
kth1	0.00	0.00	0.00	0.00
kth2	0.00	0.00	0.00	0.00
kth3	0.00	0.00	0.00	0.00
liswet1-050	0.02	0.03	0.08	0.36
liswet1-100	0.08	0.06	0.25	1.25
liswet1-200	0.35	0.14	0.45	3.30
nash1	0.00	0.00	0.00	0.00
outrata31	0.00	0.00	0.00	0.00
outrata32	0.00	0.00	0.00	0.00
outrata33	0.00	0.01	0.01	0.00
outrata34	0.00	0.00	0.00	0.00
pack-comp1-16	1.66	1.81	4.02	1.11
pack-comp1-32	1813.46	20.74	fail	50.00
pack-comp1-8	0.07	0.24	0.18	0.03
pack-comp1c-16	1.14	2.58	3.34	0.76
pack-comp1c-32	71.77	9.04	39.15	16.78
pack-comp1c-8	0.07	0.16	0.24	0.04
pack-comp2-16	7.80	2.48	33.25	2.54
pack-comp2-32	1027.19	19.53	69.78	337.57
pack-comp2-8	0.10	0.20	0.25	0.10
pack-comp2c-16	2.65	1.46	3.42	1.13
pack-comp2c-32	434.56	12.55	27.07	17.46
pack-comp2c-8	0.08	0.11	0.29	0.04
pack-rig1-16	1.49	1.47	2.00	0.96
pack-rig1-32	259.83	12.27	fail	101.87
pack-rig1-8	0.10	0.16	0.11	0.07
pack-rig1c-16	0.58	1.22	1.37	0.91
pack-rig1c-32	42.20	15.07	13.69	14.93
pack-rig1c-8	0.06	0.10	0.13	0.04
pack-rig1p-16	2.51	1.30	2.69	1.31
pack-rig1p-32	671.76	9.83	fail	24.29
pack-rig1p-8	0.11	0.14	0.25	0.08

Name	filtermpec	loqo	knitro	snopt
pack-rig2-16	0.56	fail	fail	0.27
pack-rig2-32	25.60	fail	fail	19.30
pack-rig2-8	0.08	0.14	0.13	0.05
pack-rig2c-16	0.50	fail	fail	0.33
pack-rig2c-32	22.35	fail	fail	6.47
pack-rig2c-8	0.03	0.11	0.14	0.03
pack-rig2p-16	1.52	1.52	1.91	0.68
pack-rig2p-32	131.05	fail	87.31	26.26
pack-rig2p-8	0.08	0.15	0.14	0.06
portfl1	0.02	fail	0.09	0.02
portfl2	0.02	0.20	0.08	0.02
portfl3	0.01	fail	0.08	0.02
portfl4	0.01	fail	0.08	0.03
portfl6	0.01	0.23	0.08	0.02
qpec-100-1	0.22	0.83	1.58	0.43
qpec-100-2	0.37	1.84	2.76	0.95
qpec-100-3	0.77	1.14	2.10	1.10
qpec-100-4	0.36	1.72	2.71	0.50
qpec-200-1	2.02	10.23	14.22	3.38
qpec-200-2	8.01	18.87	24.06	3.72
qpec-200-3	7.92	9.80	20.40	9.11
qpec-200-4	2.52	28.57	29.32	2.65
qpec1	0.00	0.00	0.00	0.00
qpec2	0.00	0.05	0.02	0.05
ralph1	0.00	0.01	0.00	0.00
ralph2	0.00	0.00	0.00	0.00
ralphmod	0.76	fail	4.88	0.14
scholtes1	0.00	0.00	0.01	0.00
scholtes2	0.00	0.00	0.00	0.01
scholtes3	0.00	0.00	0.01	0.00
scholtes4	0.01	fail	0.00	0.00
scholtes5	0.00	0.00	0.00	0.00
sl1	0.00	0.01	0.01	0.01
stackelberg1	0.00	0.00	0.01	0.00
tap-09	0.09	0.07	0.18	0.17
tap-15	0.31	0.33	1.47	0.94
tollmpec	164.09	fail	74.44	352.85
tollmpec1	161.54	fail	80.97	548.23
water-FL	3.72	fail	fail	0.30
water-net	0.15	fail	fail	0.32