J Optim Theory Appl (2010) 146: 51–75
DOI 10.1007/s10957-010-9655-8

On the Use of Outer Approximations as an External Active Set Strategy

H. Chung · E. Polak · S. Sastry

Published online: 12 February 2010

Abstract Outer approximations are a well known technique for solving semiinfinite optimization problems. We show that a straightforward adaptation of this technique results in a new, *external*, active-set strategy that can easily be added to existing software packages for solving nonlinear programming problems with a large number of inequality constraints. Our external active-set strategy is very easy to implement, and, as our numerical results show, it is particularly effective when applied to discretized semiinfinite optimization or state-constrained optimal control problems. Its effects can be spectacular, with reductions in computing time that become progressively more pronounced as the number of inequalities is increased.

Keywords Outer approximations · Inequality constrained optimization · Active set strategies

1 Introduction

There are important classes of optimal control and semiinfinite optimization problems, with a continuum of inequality constraints, arising in engineering design, that

Communicated by D.Q. Mayne.

The authors wish to thank Prof. Michael Saunders for his advice on warm start of NPSOL and SNOPT, and to Prof. Rembert Reemtsen for supplying us with the Kautz filter example and data. We are also grateful to our colleague Humberto Gonzalez for installing Tomlab and IPOPT on the machine we used for this paper. This work was supported by ARO SWARMS (W911NF-0510219) and ARO Phase II STTR (W911NF-06-C-0192).

H. Chung (✉) · E. Polak · S. Sastry
Department of Electrical Engineering and Computer Science, University of California, Berkeley, CA 94720-1770, USA
e-mail: hachung@eecs.berkeley.edu

E. Polak
e-mail: polak@eecs.berkeley.edu

S. Sastry
e-mail: sastry@eecs.berkeley.edu

are characterized by the fact that only a small fraction of the constraints is active at a solution. These include design of earthquake resistant structures, which sway during an earthquake and are designed to remain linear for small to moderate earthquakes and to have bounded, nonlinear, deformations for large earthquakes (see, e.g., [1, 2]), robotic manipulator motion design (see, e.g., [3]), electronic filter design (see, e.g., [4]), and computer controlled vehicles [5]. Computer controlled vehicles represent a particularly large and important class of applications for our proposed technique because often the optimization computations must be performed in real time, fast enough to keep up with the motion of the vehicle. Computer controlled vehicles, using receding horizon control laws, include robotic carts that deliver parts and supplies as the need arises on a factory floor [6], drone aircraft [7], and autonomous automobiles [8]. The discretized optimal control problems associated with these applications are characterized by a large number of collision avoidance inequalities, few of which are active. Thus, for example, if there are 16 craft involved and their trajectories are discretized using 128 points, the resulting nonlinear programming problem has $(16 \times 15/2) \times 128 = 15{,}360$ inequalities. The practical implementation of a receding horizon control law is critically dependent on the possibility of solving the associated discretized optimal control problem within a time that is determined by the speed of the moving craft. In the case of airplanes, that time is less than 10 seconds, in the case of ships, it may be around 30 seconds. Our experience with typical nonlinear programming packages such as SNOPT [9], NPSOL [10], KNITRO [11], IPOPT [12], and Schittkowski SQP [13] is that they are not able to meet these computing time requirements. The main reason being that they are unable to exploit the structure of these discretized optimal control problems. A possible exception is CFSQP [14], which was designed with such problems in mind, and which contains an active-set strategy.

The purpose of this paper is to explore the effectiveness of an adaptation of outer approximations as an active-set strategy that can be added as a small preprocessor to a standard nonlinear programming package for the solution of the above described discretized optimal control and semiinfinite optimization problems. In our test examples we include both problems with very few active inequalities and problems with a large number of active inequalities.

Outer approximations algorithms are a generalization of cutting plane methods [15] that are used to solve semiinfinite optimization problems of the form

$$(P_Y) \qquad \min_{x \in \mathbb{R}^n} \{ f(x) \mid \phi(x, y) \le 0,\ y \in Y \}, \tag{1}$$

where $Y \subset \mathbb{R}^m$ is compact (with $m = 1$ quite frequently) and $f : \mathbb{R}^n \to \mathbb{R}$ and $\phi : \mathbb{R}^n \times \mathbb{R}^m \to \mathbb{R}^q$ continuously differentiable. In the most elementary form, these algorithms construct a sequence of finitely constrained problems of the form (see [16])

$$(P_{Y_i}) \qquad \min_{x \in \mathbb{R}^n} \{ f(x) \mid \phi(x, y) \le 0,\ y \in Y_i \}, \tag{2}$$

$i = 0, 1, 2, \ldots$. The set $Y_0 \subset Y$ is arbitrary, and for $i \ge 1$,

$$Y_{i+1} = \hat{Y}_{i+1} \cup Y_i, \tag{3}$$

with $\hat{Y}_{i+1}$ a finite subset of the active constraint set $\{\arg\max_{y\in Y}\phi(\hat{x}_{i+1},y)\}$, where $\hat{x}_{i+1}$ is the solution of P_{Y_i}. It is straightforward to show that any accumulation point $\hat{x}$ of the optimizers $\hat{x}_i$ is an optimizer of P_Y. Since the constraint set of P_Y is a subset of the constraint set of P_{Y_i}, for all i, the naming of this approach "method of outer approximations" is obvious.

In this paper, we show that a simple generalization of the above described outer approximations approach can be reinterpreted as an *external* active-set strategy for solving nonlinear programming problems with a large number of inequality constraints. The generalization consists of applying only a *finite number*, $N_{\text{iter}} > 0$, of iterations of an optimization package to problem (2), and replacing the active set $\hat{Y}_{i+1}$, in (3), by a subset of the ϵ-active set $Y_{\epsilon(x_i)}(x_i) \overset{\triangle}{=} \{y \in Y \mid \phi(x_i,y) \geq \epsilon(x_i)\}$, where $\epsilon : \mathbb{R}^n \to \mathbb{R}_+$. Leineweber [17] used a similar strategy for solving the internal QP problem in his SQP method, which accumulates ϵ-active sets. However, for each selection of the active set, he solves the resulting approximating QP problem to completion, rather than performing a finite number of iterations, as we do.

Not unexpectedly, our scheme is sensitive to the choice of the two parameters, which appear in the scheme: the function $\epsilon : \mathbb{R}^n \to \mathbb{R}_+$ and the positive integer N_{iter}. Empirically, we found that setting $\epsilon(x_i) = \max(\max_y \phi(x_i,y), 0)$, the maximum constraint violation at the current point x_i, usually leads to good results. The choice of N_{iter} requires a trade-off between the cost of recomputing the maximum constraint violation and the loss of efficiency due to using a value of N_{iter} larger than one. The selection of N_{iter} may require some experimentation, and hence our scheme is most valuable in applications where very similar problems are solved over and over again. This is certainly the case in the design of earthquake resistant structures, industrial filter design, and receding horizon control of various vehicles.

Although the favorable outcome is plausible on an intuitive level, it does not appear to be possible to demonstrate theoretically that the modified outer approximations approach will result in substantial reductions of computing time. Nevertheless, our numerical examples, drawn from real world applications, show that computing times can be reduced by fairly large factors over the use of "native" optimization packages. The pattern that emerges is that the effectiveness of our active-set strategy increases as the fraction of constraints that are active at a solution decreases. Our strategy is particularly effective when used with nonlinear programming solvers that allow warm starts. Our present work follows our presentation of a related strategy [18] for solving semiinfinite minimax problems using log-sum-exponential smoothing, as well as a more closely related study involving optimal control problems [19], both of which have proved to be equally effective.

In Sect. 2 we state our strategy in the form of an algorithm and provide a theoretical justification for it, in Sect. 3 we present numerical results, and our concluding remarks are in Sect. 4.

To conclude, the advantages of our active-set strategy are that it is very simple to implement and that it often leads to a considerable reduction in computing time over the use of standard optimization code in "native" form.

Notation We will denote elements of a vector by superscripts (e.g., x^i) and elements of a sequence or a set by subscripts (e.g., x_k), and we set $\mathbb{R}_+ \overset{\triangle}{=} \{\alpha \in \mathbb{R} \mid \alpha \geq 0\}$.

2 Active-Set Strategy

Consider the inequality constrained minimization problem:

$$(P_q) \quad \min\{f^0(x) \mid f^j(x) \le 0,\ j \in q\}, \tag{4}$$

where $x \in \mathbb{R}^n$, and $q \triangleq \{1, \ldots, n_q\}$. We assume that functions $f^j : \mathbb{R}^n \to \mathbb{R}$ are at least once continuously differentiable.

Next we define the index set $q_\epsilon(x)$ with $\epsilon \ge 0$ by

$$q_\epsilon(x) \triangleq \{j \in q \mid f^j(x) \ge \psi_+(x) - \epsilon\}, \tag{5}$$

where

$$\psi(x) \triangleq \max_{j \in q} f^j(x), \tag{6}$$

and

$$\psi_+(x) \triangleq \max\{0, \psi(x)\}. \tag{7}$$

Definition 2.1 We say that an algorithm defined by a recursion of the form

$$x_{k+1} = A(x_k), \tag{8}$$

for solving inequality constrained problems of the form (4), is *convergent* if any accumulation point[1] of a sequence $\{x_i\}_{i=0}^{\infty}$, constructed according to the recursion (8), is a feasible stationary point for P_q.

Finally, we assume that we have a convergent algorithm for solving inequality constrained problems of the form (4), represented by the recursion function $A(\cdot)$, i.e., given a point x_k the algorithm constructs its successor x_{k+1} according to the rule (8).

Algorithm 2.1

```
Data: $x_0$, $\epsilon : \mathbb{R}^n \to \mathbb{R}_+$, $N_{\text{iter}} \in \mathbb{N}$.
Step 0: Set $i = 0$, $Q_0 = q_{\epsilon(x_0)}(x_0)$.
Step 1: Set $\xi_0 = x_i$ and perform $N_{\text{iter}}$ iterations of the form $\xi_{k+1} = A(\xi_k)$ on the problem
    $(P_{Q_i}) \quad \min\{f^0(\xi) | f^j(\xi) \le 0,\ j \in Q_i\}$    (9)
to obtain $\xi_{N_{\text{iter}}}$ and set $x_{i+1} = \xi_{N_{\text{iter}}}$.
Step 2: Compute $\psi(x_{i+1})$.
if $x_{i+1}$ is returned as a global, local, or stationary solution of $P_{Q_i}$ and $\psi(x_{i+1}) \le 0$, then
    STOP,
```

[1] A point $\hat{x}$ is said to be an accumulation point of the sequence $\{x_i\}_{i=0}^{\infty}$, if there exists an infinite subsequence, indexed by $K \subset \mathbb{N}$, $\{x_i\}_{i \in K}$, such that $x_i \xrightarrow{K} \hat{x}$ as $i \xrightarrow{K} \infty$.

```
else
    Compute
```

$$Q_{i+1} = Q_i \cup q_{\epsilon(x_{i+1})}(x_{i+1}), \tag{10}$$

```
    and set $i = i + 1$, and go to Step 1.
end if
```

Lemma 2.1 *Suppose that $\epsilon : \mathbb{R}^n \to \mathbb{R}_+$, that the sequence $\{x_i\}_{i=0}^{\infty}$, in $\mathbb{R}^n$, is such that $x_i \to \hat{x}$ as $i \to \infty$, and that $Q = \bigcup_{i=0}^{\infty} q_{\epsilon(x_i)}(x_i) \subset q$. For any $x \in \mathbb{R}^n$, let*

$$\psi_Q(x) = \max_{j \in Q} f^j(x). \tag{11}$$

If $\psi_Q(\hat{x}) \le 0$, then $\psi(\hat{x}) \le 0$.

Proof Since the set q is finite, there must exist an i_0 such that $Q = \bigcup_{i=0}^{i_0} q_{\epsilon(x_i)}(x_i)$. Since $q_0(x_i) \subset Q$ for all $i \ge i_0$, it follows that $\psi(x_i) = \psi_Q(x_i)$ for all $i \ge i_0$. Now, both $\psi(\cdot)$ and $\psi_Q(\cdot)$ are continuous, and hence $\psi(\hat{x}) = \lim \psi(x_i) = \lim \psi_Q(x_i) = \psi_Q(\hat{x})$. The desired result now follows directly. □

Lemma 2.2 *Suppose that $Q \subset q$ and consider the problem*

$$P_Q \quad \min\{f^0(x) | f^j(x) \le 0, j \in Q\}. \tag{12}$$

Suppose that $\hat{x} \in \mathbb{R}^n$ is feasible for P_q, i.e, $f^j(x) \le 0$ for all $j \in q$.

(a) *If $\hat{x}$ is a global minimizer for P_Q, then it is also a global minimizer for P_q.*
(b) *If $\hat{x}$ is a local minimizer for P_Q, then it is also a local minimizer for P_q.*
(c) *If $\hat{x}$ is a stationary point for P_Q, i.e., it satisfies the F. John conditions* [20] (*or Theorem* 2.2.4, *p.* 188 *in* [16]), *then it is also a stationary point for P_q.*

Proof Clearly, since $\hat{x}$ is feasible for P_q it is also feasible for P_Q.

(a) Suppose that $\hat{x}$ is not a global minimizer for P_q. Then there exists an x^* such that $f^j(x^*) \le 0$ for all $j \in q$ and $f^0(x^*) < f^0(\hat{x})$. Now, x^* is also feasible for P_Q and hence $\hat{x}$ cannot be a global minimizer for P_Q, a contradiction.

(b) Suppose that $\hat{x}$ is not a local minimizer for P_q. Then there exists a sequence $\{x_i\}_{i=0}^{\infty}$ such that $x_i \to \hat{x}$, $f^0(x_i) < f^0(\hat{x})$ and $f^j(x_i) \le 0$ for all i and $j \in q$. But this contradicts the assumption that $\hat{x}$ is a local minimizer for P_Q.

(c) Since $\hat{x}$ satisfies the F. John conditions for P_Q, there exist multipliers $\mu^0 \ge 0$, $\mu^j \ge 0$, $j \in Q$, such that $\mu^0 + \sum_{j \in Q} \mu^j = 1$,

$$\mu^0 \nabla f^0(\hat{x}) + \sum_{j \in Q} \mu^j \nabla f^j(\hat{x}) = 0 \tag{13}$$

and

$$\sum_{j \in Q} \mu^j f^j(\hat{x}) = 0. \tag{14}$$

Clearly, $\hat{x}$ also satisfies the F. John conditions for P_q with multipliers $\mu^j = 0$ for all $j \notin Q$ and otherwise as for P_Q. □

Combining the above lemmas, we get the following convergence result.

Theorem 2.1 *Suppose that the problem P_q has feasible solutions, i.e., there exist vectors x^* such that $f^j(x^*) \le 0$ for all $j \in q$.*

(a) *If Algorithm 2 constructs a finite sequence $\{x_i\}_{i=0}^k$, exiting in Step 2, with $i + 1 = k$, then x_k is a global, local, or stationary solution for P_q, depending on the exit message from the solver defined by $A(\cdot)$.*
(b) *If $\{x_i\}_{i=0}^\infty$ is an infinite sequence constructed by Algorithm 2 in solving P_q. Then any accumulation point $\hat{x}$ of this sequence is feasible and stationary for P_q.*

Proof (a) If sequence $\{x_i\}_{i=0}^k$ is finite, then, by the exit rule, it is feasible for P_q and it is a global, local, or stationary solution for P_{Q_i}. It now follows from Lemma 4, that it is also a global, local, or stationary solution for P_q.

(b) Since the sets Q_i grow monotonically, and since q is finite, there must exist an i_0 and a set $Q \subset q$, such that $Q_i = Q$ for all $i \ge i_0$. Next, it follows from the fact that $A(\cdot)$ is convergent, that for any accumulation point $\hat{x}$, $\psi_Q(\hat{x}) \le 0$ and hence, from Lemma 2.1 that $\psi(\hat{x}) \le 0$, i.e., that $\hat{x}$ is a feasible point for P_q. It now follows from the fact that $A(\cdot)$ is convergent and Lemma 2.2 that any accumulation point $\hat{x}$ is stationary for P_q. □

Remark 2.1 In the numerical experiments presented in the next section, we set

$$\epsilon(x_i) \overset{\triangle}{=} \min\{\psi_+(x_i), 1\}, \tag{15}$$

to prevent Q_i from including too many constraints.

An alternative to setting $\epsilon(x) = \min\{\psi(x), \bar{\epsilon}\}$, for some $\bar{\epsilon} > 0$, is to set $\epsilon(x) = \mathit{const} > 0$, i.e., to use a fixed value. For our optimal control problem, it is possible to find a *const* that yields better results than the ones we have presented, see [21]. However, finding such a *const* requires time-consuming experimentation. Unless one is faced with a situation where a particular problem has to be solved over and over again (with different initial conditions), we recommend the use of $\epsilon(x) = \min\{\psi(x), \bar{\epsilon}\}$, for some $\bar{\epsilon} > 0$.

3 Numerical Results

We will now present four numerical examples. The first involves the control of drones (also known as unmanned aerial vehicles, or UAV's), the second involves a Kautz filter design, the third involves the computation of a minimum-time trajectory for a 2-link manipulator, and the fourth involves of the optimal control of a particle in a plane.

The numerical experiments were performed using MATLAB V7.6 and TOMLAB V7.1 [22] and using MATLAB V7.6 together with IPOPT 3.7 [12] with the linear solver PARDISO 3.3 [23]. We used a desktop with two AMD Opertron 2.2 GHz processors with 8 GB RAM, running Linux 2.6.28.

The TOMLAB optimization solvers tested in this paper were the Schittkowski SQP algorithm with cubic line search [13], NPSOL 5.02 [10], SNOPT 6.2 [9], and KNITRO [11].

It should be clear from the form of Algorithm 2.1, that our strategy benefits considerably from the nonlinear programming solvers' ability to use warm starts. Hence it is desirable to use solvers with as extensive a warm start capability as possible, so that one can transmit the last value of important information from the last iteration of a solver on the problem P_{Q_i} as initial conditions for solving the problem $P_{Q_{i+1}}$. As far as warm starts are concerned, SNOPT allows the user to provide initial variables, their states, and slack variables. NPSOL allows the user to provide initial variables and their states, Lagrange multipliers, as well as an initial Hessian approximation matrix for quasi-Newton updates. conSolve, the TOMLAB implementation of the Schittkowski SQP algorithm, allows the user to provide initial variables and an initial Hessian matrix approximation. IPOPT allows the user to provide initial variables and Lagrange multipliers. KNITRO allows the user to provide initial variables only and hence its performance is not enhanced by the use of our method.

Since different algorithms use different stopping criteria, we evaluated the quality of their solutions using the optimality function θ defined in (2.2.9e) in [16], for problems of the form (4), with all functions continuously differentiable. For convenience, we reproduce this optimality function for problems of the form, below, with all functions continuously differentiable.

$$\theta(x) = -\min_{\mu\in\Sigma^0_{n_q}} \left\{ \mu^0\gamma\psi_+(x) + \sum_{j=1}^{n_q} \mu^j[\psi_+(x) - f^j(x)] \right.$$
$$\left. + \frac{1}{2\delta}\left\| \mu^0 \sum_{k=1}^{n_q} \mu^j \nabla f^j(x) \right\|^2 \right\},$$

where $\gamma = 1$ and $\delta = 0.5$ are parameters, and $\Sigma^0_{n_q} \triangleq \{\mu | \mu \in \mathbb{R}^{n_q+1}_+, \sum_{j=0}^{n_q} \mu^j = 1\}$. Note that $\theta(x) \le 0$ for all x and is an estimate of the "cost-to-go". Also, $\theta(x) = 0$ if x satisfies the F. John optimality conditions. Values of $\theta(x)$ larger than -10^6 are indicative of x being quite close to a local minimizer.

3.1 Control of Eight UAVs

First, we consider the problem of controlling eight identical UAV's which are required to fly, indefinitely, inside a circle in a horizontal plane, without incurring a collision, for an indefinite period of time. Their controls over a given time horizon T are to be determined by a centralized computer, to be used in a control scheme known as Receding Horizon Control (see [24]).

For the sake of simplicity, we assume that each UAV flies at a constant speed v and that the scalar control u_i, for the i-th UAV determines its rate of heading angle change in radians per second. The cost function for this problem is proportional to the sum of the energy used by the UAV's over the interval $[0, T]$, i.e.,

$$f^0(u) \triangleq \sum_{i=1}^{8} \int_0^T \frac{1}{2} u_i^2(\tau) d\tau, \tag{16}$$

where $u \triangleq (u_1, u_2, \ldots, u_8)$. The constraints, to be made explicit shortly, are those of staying inside the circle and collision avoidance.

In order to state the optimal control problem as an end-point problem defined on $[0, 1]$, we rescale the state dynamics of each UAV using the actual terminal time T and augment the 3-dimensional physical state[2] (x_i^1, x_i^2, x_i^3) with a fourth component, x_i^4,

$$x_i^4(t) \triangleq \int_0^t \frac{T}{2} u_i^2(\tau) d\tau, \tag{17}$$

which represents the energy used by the i-th UAV. The resulting dynamics of the i-th UAV have the form

$$\frac{dx_i(t)}{dt} = \begin{bmatrix} Tv \cos x_i^3(t) \\ Tv \sin x_i^3(t) \\ Tu_i(t) \\ \frac{T}{2} u_i(t)^2 \end{bmatrix} \triangleq h(x_i(t), u_i(t)) \tag{18}$$

with the initial state $x_i(0)$ given. We will denote the solution of the dynamic equation (18) by $x_i(t, u_i)$, with $t \in [0, 1]$.

The optimal control problem we need to solve is of the form

$$\min_{u \in L_\infty^8[0,1]} f^0(u) \triangleq \sum_{i=1}^{8} x_i^4(1, u_i) \tag{19}$$

subject to two sets of constraints:

(a) Stay-in-a-circle constraints:

$$f_{\text{bnd}}^i(t, u_i) \triangleq x_i^1(t, u_i)^2 + x_i^2(t, u_i)^2 \le r_{\text{bnd}}^2, \quad \forall t \in [0, 1],\ i = 1, 2, \ldots, 8, \tag{20}$$

and

(b) Pairwise collision avoidance constraints:

$$f_{\text{ca}}^{(i,j)}(t, u_i, u_j) \triangleq (x_i^1(t, u_i) - x_j^1(t, u_j))^2 + (x_i^2(t, u_i) - x_j^2(t, u_j))^2 \ge r_{\text{ca}}^2,$$
$$\forall t \in [0, 1], i \ne j, i, j = 1, 2, \ldots, 8. \tag{21}$$

[2] (x_i^1, x_i^2) denotes the position coordinates of the i-th UAV on a plane, and x_i^3 denotes the heading angle in radian of the i-th UAV, respectively.

To solve this problem, we must discretize the dynamics. We use Euler's method to obtain

$$\bar{x}_i(t_{k+1}) - \bar{x}_i(t_k) = \Delta h(\bar{x}_i(t_k), \bar{u}_i(t_k)), \qquad \bar{x}_i(0) = x_i(0), \quad i = 0, 1, 2, \dots, 8, \tag{22}$$

with $\Delta \overset{\triangle}{=} 1/N$, $N \in \mathbb{N}$, $t_k \overset{\triangle}{=} k\Delta$ and $k \in \{0, 1, \dots, N\}$. We use an over-bar to distinguish between the exact variables and the discretized variables. We will denote the solution of the discretized dynamics by $\bar{x}_i(t_k, \bar{u}_i)$, $k = 0, 1, \dots, N$, with

$$\bar{u}_i \overset{\triangle}{=} (\bar{u}_i(t_0), \bar{u}(t_1), \dots, \bar{u}_i(t_{N-1})). \tag{23}$$

Finally, we obtain the following discrete-time optimal control problem:

$$\min_{\bar{u}_i \in \mathbb{R}^N,\ i \in \{1,\dots,8\}} \bar{f}^0(\bar{u}) \overset{\triangle}{=} \sum_{i=1}^{8} \bar{x}_i^4(1, \bar{u}_i) \tag{24}$$

subject to $|\bar{u}_i(t_k)| \le b$ for $k = 0, 1, \dots, N-1$ and $i = 1, 2, \dots, 8$, the discretized dynamics of each UAV (22), and the discretized stay-in-a-circle and collision avoidance constraints:

$$\bar{f}^k_{\text{bnd},i}(\bar{u}^i) \overset{\triangle}{=} \bar{x}_i^1(t_k, \bar{u}_i)^2 + \bar{x}_i^2(t_k, \bar{u}_i)^2 \le r_{\text{bnd}}^2, \quad k \in \{1, \dots, N\}, \tag{25}$$

and

$$f^k_{\text{ca},(i,j)}(\bar{u}_i, \bar{u}_j) \overset{\triangle}{=} (\bar{x}_i^1(t_k, \bar{u}_i) - \bar{x}_j^1(t_k, \bar{u}_j))^2 + (\bar{x}_i^2(t_k, \bar{u}_i) - \bar{x}_j^2(t_k, \bar{u}_j))^2 \ge r_{\text{ca}}^2,$$
$$k \in \{1, \dots, N\}, i \ne j,\ i, j = 1, 2, \dots, 8. \tag{26}$$

The total number of nonlinear inequality constraints in this problem is $8N(8-1)/2 + 8N$. Clearly, (24), (25), and (26) are a mathematical programming problem that is distinguished from ordinary mathematical programming problems only by the fact that adjoint equations can be used in the computation of the gradients of the functions.

We set $v = 0.5$, $r_{\text{bnd}} = 4$, $r_{\text{ca}} = 1$, $T = 25$, $b = 1$ and $N = 64$, resulting in 2304 nonlinear inequality constraints. The initial conditions and initial controls for each UAV are set as

$$\begin{aligned} &x_0^1 = (2.5, 2.5, \pi, 0), \qquad x_0^2 = (-2.5, 2, -\pi/2, 0),\\ &x_0^3 = (-2.5, -2.5, -\pi/4, 0), \qquad x_0^4 = (2, -2.5, \pi/2, 0),\\ &x_0^5 = (2.5, 0, \pi/2, 0), \qquad x_0^6 = (-2.5, 0, -\pi/2, 0),\\ &x_0^7 = (0, 3, -3\pi/4, 0), \qquad x_0^8 = (0, -3, \pi/4, 0),\\ &\bar{u}_0^i = 1.25 \times 10^{-1} 1_{1\times N} \quad \forall i \in \{1, \dots, N_a\}, \end{aligned} \tag{27}$$

where $1_{1\times N}$ represents a $1 \times N$ vector whose elements are all ones. The numerical results are summarized in Tables 1–3. In these tables, N_{grad}, the total number of gradient evaluations, and t_{CPU}, the total CPU time for achieving an optimal solution using Algorithm 2.1, are defined as follows:

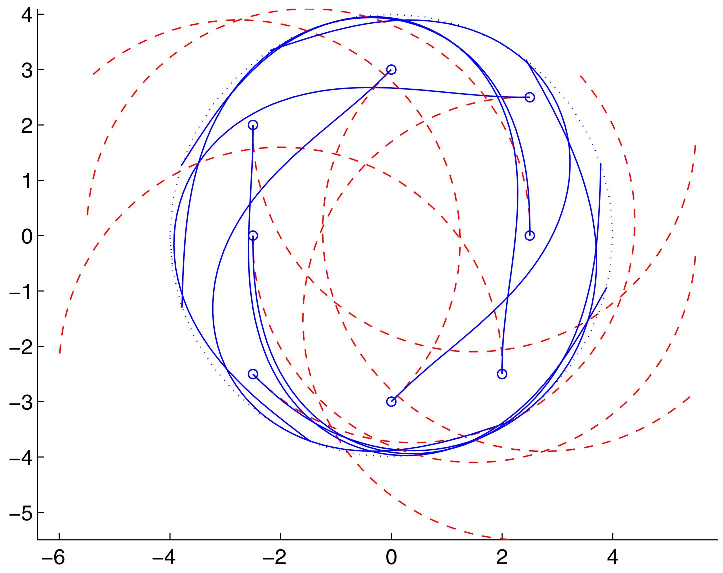

Fig. 1 (Color online) Initial trajectories (*dashed red*) and optimal trajectories (*solid blue*). Bounding circular region is represented by the *dotted blue circle*

$$
\begin{aligned}
N_{\text{grad}} &= \sum_{i=0}^{i_T} |Q_i| \times \text{number of gradient function calls during } i\text{-th inner iteration},\\
t_{\text{CPU}} &= \sum_{i=0}^{i_T} \big[\text{CPU time spent for the } i\text{-th inner iteration}\\
&\qquad + \text{CPU time spent for setting up the } i\text{-th inner iteration}\big]. \qquad (28)
\end{aligned}
$$

In the above, and in the tables, i_T is the value of the iteration index i at which Algorithm 2.1 is terminated by the termination tests incorporated in the optimization solver used, and i_{stab} is the value of the index i at which $|Q|$ is stabilized. Also, $\%_{\text{nat}}$, the percentage of t_{CPU} with respect to the computation time with the *native* algorithm, i.e. using the solver with the full set of constraints (shown in the last row of each table), and $\theta^* \triangleq \theta(x^*)$, the value of the optimality function at the minimizer x^*, is used in tables.

Figure 1 shows the trajectories for a locally optimal solution for the eight UAV problem. There are only 16 active constraints out of 2304 at the end. These are all associated with staying in the circle; there are no active collision avoidance constraints. When properly adjusted, Algorithm 2.1 accumulates fewer than 90 constraints. Consequently, the reduction in the number of gradient computations is huge.

In Table 1, the best result using Algorithm 2.1, with the Schittkowski SQP defining the map $A(\cdot)$, was achieved with data set 3, which required about 1/3 of the CPU time used by the native Schittkowski SQP algorithm. With NPSOL, the reduction was more significant, and a locally optimal solution was obtained using about 1/12

Table 1 External active-set strategy with Schittkowski SQP, eight-UAV example

Data #	N_{iter}	i_T	f^0	N_{grad}	$\|Q\|$	i_{stab}	t_{CPU}	$\%_{\text{nat}}$	θ^*
1	10	37	8.0533	124384	281	31	1590.8	47.7	−1.04e-08
2	20	19	4.0969	70748	305	18	1117.2	33.5	−1.39e-08
3	30	16	3.1467	63622	189	15	1105.3	33.1	−1.41e-08
Native			1.7916	463104	2304		3337.8	100	−2.21e-06

Table 2 External active-set strategy with NPSOL, eight-UAV example

Data #	N_{iter}	i_T	f^0	N_{grad}	$\|Q\|$	i_{stab}	t_{CPU}	$\%_{\text{nat}}$	θ^*
1	10	13	1.7028	10709	115	10	129.5	8.1	−7.31e-11
2	20	11	1.7028	11212	91	9	145.8	9.2	−8.50e-09
3	30	9	1.7028	13606	86	9	169.6	10.7	−2.83e-14
Native			1.7916	237312	2304		1588.9	100	−8.90e-11

Table 3 External active-set strategy with SNOPT, eight-UAV example

Data #	N_{iter}	i_T	f^0	N_{grad}	$\|Q\|$	i_{stab}	t_{CPU}	$\%_{\text{nat}}$	θ^*
1	10	8	1.7028	2137	84	8	34.6	13.7	−2.28e-09
2	20	8	1.7028	2157	84	8	35.8	14.2	−3.71e-10
3	30	8	1.7028	2157	84	8	35.2	14.0	−3.71e-10
Native			1.7916	36864	2304		251.8	100	−7.48e-09

Table 4 External active-set strategy with IPOPT, eight-UAV example

Data #	N_{iter}	i_T	f^0	N_{grad}	$\|Q\|$	i_{stab}	t_{CPU}	$\%_{\text{nat}}$	θ^*
1	10	9	1.7028	2501	84	8	35.4	6.1	−8.08e-09
2	20	8	1.7028	2424	84	8	34.6	6.0	−8.08e-09
3	30	8	1.7028	2424	84	8	34.4	5.9	−8.08e-09
Native			1.7916	71424	2304		578.3	100	−8.39e-09

of the CPU time used by NPSOL with the full constraint set (data set 1 in Table 2). When SNOPT was used as the map $A(\cdot)$ in Algorithm 2.1, the reduction about $1/7$ was achieved with data set 1 in Table 3. In this example, IPOPT with Algorithm 2.1 showed the best performance in overall. The reduction about $1/16$ was achieved with data set 3 in Table 4.

To summarize, our overall fastest solution of the optimal control problem was obtained using data set 3 with the IPOPT algorithm. This computing time was $1/98$ of the time required by the native Schittkowski SQP, $1/7$ of the time required by

Table 5 Result using the native KNITRO, eight-UAV example

Solver	f^0	N_{grad}	t_{CPU}	θ^*
KNITRO	1.7916	94464	627.7	−2.43e-07

the native SNOPT, 1/16 of the time required by the native IPOPT, 1/18 of the time required by the native KNITRO, and 1/46 of the time required by the native NPSOL.

It is also interesting to observe that in their native form, all the algorithms tested computed the same local minimizer, but when enhanced with our active set procedure, with the exception of the Schittkowski algorithm, all algorithms converged to a better local minimizer. The values of θ^*, in Tables 1–4, show that it was not a case of the native algorithms jamming near a local minimum.

Finally, comparing the result in Table 5 with those in Tables 1–4, we conclude that on this problem, in native form, KNITRO is inferior to SNOPT and IPOPT, but better than the others.

3.2 Design of Pink-Noise Kautz Filter of Even Order

This problem requires the computation of coefficients for a Kautz filter so as to get a best fit to the desired profile, defined by the function $\tilde{F}(y)$, below. The best fit is defined in terms of the solution of the semiinfinite minimax problem

$$\min_x \max_y \left|\log_{10}(|H(x,y)|) - \log_{10}(\tilde{F}(y))\right|, \quad x \in \mathbb{R}^{2N},\ y \in [\varepsilon_1, 1-\varepsilon_1], \tag{29}$$

subject to the constraints

$$\begin{aligned}
&x^{2k-1} - \frac{x^{2k}}{1-\varepsilon_2} - (1-\varepsilon_2) \le 0,\\
&-x^{2k-1} - \frac{x^{2k}}{1-\varepsilon_2} - (1-\varepsilon_2) \le 0, \quad k = 1,\dots,N,\ \varepsilon_2 \in (0,1),\\
&x^{2k} - (1-\varepsilon_2)^2 \le 0,
\end{aligned} \tag{30}$$

where

$$\begin{aligned}
H(x,y) = \sum_{k=1}^{N} \Bigg[&(x^{N+2k+1}\ p^k(e^{i2\pi y} - 1) + x^{N+2k} q^k(e^{i2\pi y} + 1))\\
&\times \frac{1}{1 + x^1 e^{i2\pi y} + x^2 e^{i4\pi y}} \prod_{l=1}^{k} \frac{x^{2l} + x^{2l-1} e^{i2\pi y} + e^{i4\pi y}}{1 + x^{2l+1} e^{i2\pi y} + x^{2l+2} e^{i4\pi y}} \Bigg],
\end{aligned} \tag{31}$$

$$\begin{aligned}
p^k &= \sqrt{\frac{(1-x^{2k})(1+x^{2k}-x^{2k-1})}{2}},\\
q^k &= \sqrt{\frac{(1-x^{2k})(1+x^{2k}+x^{2k-1})}{2}},
\end{aligned} \tag{32}$$

and

$$\tilde{F}(y) = \begin{cases} \frac{1}{\sqrt{2\pi y}}, & y \in [\varepsilon_1, \frac{1}{2}], \\ \frac{1}{\sqrt{2\pi(1-y)}}, & y \in [\frac{1}{2}, 1-\varepsilon_1]. \end{cases} \tag{33}$$

In order to transcribe this semiinfinite minimax problem into an ordinary minimax problem, we discretize the set Y, and set it to be $Y = \{y_1, y_2, \dots, y_{N_d}\}$. Note that the discretization should be logarithmically spaced, since we are looking for the best fit in the frequency domain. Next, to convert this minimax problem into a constrained nonlinear programming problem, we introduce the slack variable $x^{2N+1} \geq 0$, and define

$$\bar{x} \triangleq \begin{bmatrix} x \\ x^{2N+1} \end{bmatrix}. \tag{34}$$

Then the original minimax problem becomes the nonlinear programming problem

$$\min_{\bar{x}} x^{2N+1} \tag{35}$$

subject to constraints (30), and

$$-x^{2N+1} \leq \log_{10}(|H(x, y_i)|) - \log_{10}(\tilde{F}(y_i)) \leq x^{2N+1}, \quad i = 1, 2, \dots, N_d. \tag{36}$$

Most of the above nonlinear inequality constraints are active at a minimum.

For numerical experiments, we applied our algorithm only to nonlinear inequality constraints (36), and took into account linear constraints (30) all the time. We set $\varepsilon_1 = \varepsilon_2 = 0.01$, $N = 20$, and $N_d = 63$, which results in $2 \times N_d = 126$ nonlinear inequality constraints. The initial condition is set as

$$x_0 = 0.7\ 1_{2N\times 1}. \tag{37}$$

When our algorithm is applied to nonlinear programming problems obtained by transcription of minimax problems via the addition of an additional variable, say x^{n+1}, the initial value of x^{n+1} must be chosen so that the ϵ-active set of the equivalent nonlinear programming problem is nonempty in the beginning. Otherwise, the solver causes $x^{n+1} \to -\infty$ in the first inner iteration, which is useless.

For this example, we set up the initial value of the additional variable as follows:

$$x_0^{2N+1} = \max_{i=1,\dots,N_d} |\log_{10}(|H(x_0, y_i)|) - \log_{10}(\tilde{F}(y_i))|, \tag{38}$$

which ensures that the ϵ-active set is not empty for any $\epsilon > 0$.

The Kautz filter design is a highly ill-conditioned problem with many local minima, which proved to be impossible to solve with two of the four solvers that we were using. The Schittkowski SQP, failed and reported an error message, "Too large penalty", NPSOL failed and reported the error message "Current point cannot be improved on." SNOPT and IPOPT were able to solve the problem, but the computing time was very long. KNITRO posted very good computing times, but converged to a much inferior local minimum.

Table 6 External active-set strategy with Schittkowski SQP, Kautz filter design example

Data #	N_{iter}	i_T	f^0	N_{grad}	$\lvert Q\rvert$	i_{stab}	t_{CPU}	$\%_{\text{nat}}$	θ^*
1	10	17	1.073e-03	42157	122	11	57.3		−1.11e-06
2	20	40	1.026e-03	200497	120	16	256.2		−1.06e-07
3	30	19	1.104e-03	109642	114	11	139.5		−2.39e-07
Native					126				

Table 7 External active-set strategy with NPSOL, Kautz filter design example

Data #	N_{iter}	i_T	f^0	N_{grad}	$\lvert Q\rvert$	i_{stab}	t_{CPU}	$\%_{\text{nat}}$	θ^*
1	10	118	5.133e-08	483171	126	22	601.6		−4.77e-09
2	20	109	3.713e-07	938074	126	17	1125.0		−8.21e-09
3	30	63	1.663e-06	727929	124	53	864.0		−2.43e-10
Native					126				

Table 8 External active-set strategy with SNOPT, Kautz filter design example

Data #	N_{iter}	i_T	f^0	N_{grad}	$\lvert Q\rvert$	i_{stab}	t_{CPU}	$\%_{\text{nat}}$	θ^*
1	10	33	6.116e-07	155033	126	9	188.2	16.2	−4.03e-07
2	20	13	1.360e-05	111460	125	7	132.5	11.4	−4.25e-07
3	30	10	1.008e-05	119415	126	6	142.3	12.3	−3.19e-07
Native			2.413e-07	1011906	126		1158.5	100	−4.40e-06

Table 9 External active-set strategy with IPOPT, Kautz filter design example

Data #	N_{iter}	i_T	f^0	N_{grad}	$\lvert Q\rvert$	i_{stab}	t_{CPU}	$\%_{\text{nat}}$	θ^*
1	10	94	4.174e-08	125064	126	41	276.9	2.0	−5.02e-09
2	20	415	4.183e-08	1074624	126	414	2016.0	14.8	−5.12e-09
3	30	14	3.147e-07	47097	126	5	74.9	0.6	−4.98e-09
Native			5.136e-06	1581804	126		13585.8	100	−4.98e-09

Using Algorithm 2.1, with Schittkowski SQP and NPSOL as the map $A(\cdot)$, optimal solutions were found, as shown in Tables 6 and 7. Referring to Table 8, we see that when used with SNOPT, Algorithm 2.1 achieved reductions in computation time ranging from 84% to 89%, depending on the choice of the parameter N_{iter}. The reductions in computation time achieved by IPOPT were dramatic. With IPOPT, a locally optimal solution was obtained using about $1/166$ of the CPU time used by IPOPT with the full constraint set as shown in Table 9.

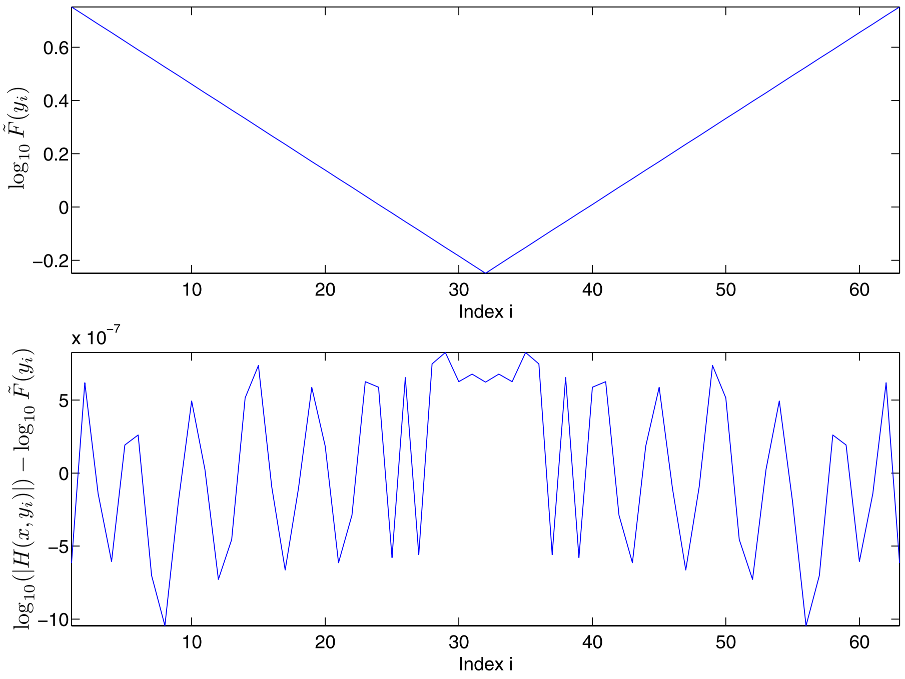

Fig. 2 Desired profile $\tilde{F}(y_i)$ (*upper plot*) and the error $\log_{10}(|H(x, y_i)|) - \log_{10}(\tilde{F}(y_i))$ at a minimizer (data set #1 in Table 8)

In the Kautz filter design, most of the constraints are eventually included in the active set Q_i (see the values of $|Q|$ in the tables). In spite of this, when used with SNOPT and IPOPT, our algorithm was able to reduce computation time, because only a few constraints were active in the early iterations.

To summarize, Algorithm 2.1 enabled the Schittkowski SQP and NPSOL to find a solution, and enhanced the performance of SNOPT and IPOPT. Finally, referring to Fig. 2, which is a plot of $\log_{10} \tilde{F}(y_i)$, we see that from a practical point of view all the designs obtained are very good, since the maximum errors, which are of the order of 10^{-3} to 10^{-8} are very small relative to the desired reference filter values, which are in the range of -0.2 to 0.7. The best accuracy of fit was obtained using enhanced IPOPT.

The results obtained by KNITRO were fastest among the native algorithms, but the quality of the solution was not as good.

3.3 Minimum-Time Trajectory Planning for a 2-Link Manipulator

Our last numerical example involves the computation of a minimum-time trajectory for an 2-link (and two joint) manipulator [3], see Fig. 3. The end tip of the manipulator is required to pass through 4 via points in the Cartesian 2-dimensional plane to which the motion is confined. Using inverse kinematics, each via point can be converted into a vector in the 2-dimensional joint-variable space. Following the convention in

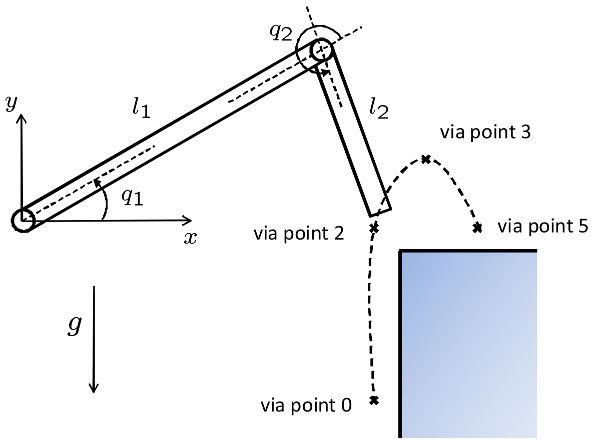

Fig. 3 Configuration of a two-link robot arm

[3], we introduce six via points in the 2-dimensional joint variable space:

$$q^i \triangleq [q_1^i, q_2^i]^T, \quad i = 0, 1, \dots, 5, \tag{39}$$

where q^0, q^2, q^3, and q^5 are the image, under inverse kinematics, of the given via points in Cartesian space, while the via points q^1 and q^4 are free variables, which must be added in order to guarantee continuity in velocity and acceleration, see [3].

We will denote the joint vector motion by $q(t) \in \mathbb{R}^2$. Let $\delta > 0$ and let $t_i \in [\delta, \infty)$ denote the time required for the manipulator to move from the $i-1$-th via point to the i-th via point. We define the interval vector $T \triangleq [t_1 \ \cdots \ t_5]^T \in S$, where $S \triangleq \{T \in \mathbb{R}^5 | t_i \in [\delta, \infty),\ i = 1, 2, \dots, 5\}$.

For $i = 1, 2, \dots, 5$, let

$$\sigma_i = \sum_{j=1}^{i} t_j \tag{40}$$

and we set $\sigma_0 = 0$.

Our objective is to minimize σ_5.

We define the position, velocity, and acceleration vectors at i-th via point, $i = 0, 1, \dots, 5$, by

$$q(\sigma_i) \triangleq \begin{bmatrix} q_1^i \\ q_2^i \end{bmatrix} = q^i, \qquad \dot{q}(\sigma_i) \triangleq \dot{q}^i = \begin{bmatrix} \dot{q}_1^i \\ \dot{q}_2^i \end{bmatrix}, \qquad \ddot{q}(\sigma_i) \triangleq \ddot{q}^i = \begin{bmatrix} \ddot{q}_1^i \\ \ddot{q}_2^i \end{bmatrix}. \tag{41}$$

We assume that $\dot{q}^0$, $\dot{q}^5$, $\ddot{q}^0$, and $\ddot{q}^5$ are given.

We parameterize the joint trajectory, $q(t)$, using ten cubic splines, two for each time interval where the robot arm travels from via point q^i to via point q^{i+1}, defined on the intervals $[0, t_i]$, of the following form

$$p_k^i(t) \triangleq q_k^{i-1} + \dot{q}_k^{i-1} t + \left[\frac{3}{t_i^2}(q_k^i - q_k^{i-1}) - \frac{1}{t_i}(\dot{q}_k^i + 2\dot{q}_k^{i-1})\right] t^2 + \left[-\frac{2}{t_i^3}(q_k^i - q_k^{i-1}) + \frac{1}{t_i^2}(\dot{q}_k^i + \dot{q}_k^{i-1})\right] t^3, \quad (42)$$

where $i = 1, 2, \dots, 5$, $k = 1, 2$, and $t \in [0, t_i]$. For $t \in [\sigma_{i-1}, \sigma_i]$, $i = 1, 2, \dots, 5$, we set

$$q(t) = p^i(t - \sigma_{i-1}) = [p_1^i(t - \sigma_{i-1}), p_2^i(t - \sigma_{i-1})]^T. \quad (43)$$

To obtain a semiinfinite programming formulation in standard form, we need to normalize (42) in order to define the problem on a fixed set [0, 1], rather than variable sets $[0, t_i]$. Setting $t = st_i$ for $s \in [0, 1]$, (42) becomes

$$p_k^i(st_i) \triangleq q_k^{i-1} + \dot{q}_k^{i-1} t_i s + \left[3(q_k^i - q_k^{i-1}) - t_i(\dot{q}_k^i + 2\dot{q}_k^{i-1})\right] s^2 + \left[-2(q_k^i - q_k^{i-1}) + t_i(\dot{q}_k^i + \dot{q}_k^{i-1})\right] s^3, \quad (44)$$

where $i = 1, 2, \dots, 5$, and $k = 1, 2$.

The continuity in joint accelerations is enforced by the relations

$$\ddot{p}_k^1(0) = \ddot{q}_k^0, \ \ \ddot{p}_k^5(t_5) = \ddot{q}_k^5, \quad (45)$$

$$\ddot{p}_k^{i+1}(0) = \ddot{p}_k^i(t_i), \quad (46)$$

for all $i = 1, \dots, 5$ and $k = 1, 2$. From (45), we can define the free variables q_k^1 and q_k^5 as functions of unknown variables $\dot{q}_k^1$ and $\dot{q}_k^5$. The remaining unknowns, $\dot{q}_k^1, \dots, \dot{q}_k^5$ can be obtained by solving a linear system of equations defined by (46), in terms of the t_i's. Therefore, $p_k^i(\cdot)$ is parameterized in terms of T, i.e., $p_k^i(\cdot)$ is now a function of s and T. In [3], it was shown that the system of equations defining the $p_k^i(\cdot)$ always has a solution for any $T \in S$.

For $i = 1, 2, \dots, 5$ and $k = 1, 2$, with the time origin shifted to σ_{i-1}, the torque at k-th joint in the i-th time interval, induced by the manipulator motion has the form

$$\tau_k^i(t) = M_k(p(t), \dot{p}(t), \ddot{p}(t)), \quad t \in [0, t_i], \quad (47)$$

where $p(t)$, $\dot{p}(t)$, and $\ddot{p}(t)$ are the joint position vector, the joint velocity vector, and the joint acceleration vector respectively. $\tau_k^i(t)$ is defined by (dependencies on t is omitted below for simplicity)

$$\begin{aligned}
\tau_1^i &= m_2 l_2^2(\ddot{p}_1^i + \ddot{p}_2^i) + m_2 l_1 l_2 \cos(p_2^i)(2\ddot{p}_1^i + \ddot{p}_2^i) \\
&\quad + (m_1 + m_2) l_1^2 \ddot{p}_1^i - m_2 l_1 l_2 \sin(p_2^i)(\dot{p}_2^i)^2 \\
&\quad - 2 m_2 l_1 l_2 \sin(p_2^i) \dot{p}_1^i \dot{p}_2^i + m_2 l_2 \cos(p_1^i + p_2^i) \\
&\quad + (m_1 + m_2) l_1 g \cos(p_1^i), \\
\tau_1^i &= m_2 l_1 l_2 \cos(p_2^i) \ddot{p}_1^i + m_2 l_1 l_2 \sin(p_2^i)(\dot{p}_1^i)^2 \\
&\quad + m_2 l_2 g \cos(p_1^i + p_2^i) + m_2 l_2^2(\ddot{p}_1^i + \ddot{p}_2^i),
\end{aligned} \quad (48)$$

where m_k and l_k denote the mass and the length of k-th link, and g is the gravitational acceleration.

Since we have defined $t = st_i$, it follows that $dt = t_i ds$, and hence we have

$$\dot{p}(t) = \frac{dp(t)}{dt} = \frac{1}{t_i}\frac{dp(st_i)}{ds},$$
$$\ddot{p}(t) = \frac{d\dot{p}(t)}{dt} = \frac{1}{t_i^2}\frac{d^2p(st_i)}{d^2s}. \tag{49}$$

If we assume that $p(t)$ is expressed in terms of normalized cubic splines $p_k^i(s;T)$, then we obtain that

$$\tau_k^i(s;T) = M_k\left(p^i(s;T), \frac{1}{t_i}\frac{dp^i(s;T)}{ds}, \frac{1}{t_i^2}\frac{d^2p^i(s;T)}{d^2s}\right), \quad s \in [0,1], \tag{50}$$

where $p^i(s;T) = [p_1^i(s;T),\ p_2^i(s;T)]^T$. Also,

$$\frac{d\tau_k^i(s;T)}{dt} = \frac{1}{t_i}\frac{d\tau_k^i(s;T)}{ds} = \frac{1}{t_i}\frac{dM_k(\cdot)}{ds}, \quad s \in [0,1]. \tag{51}$$

The minimum-time trajectory planning problem can now be formulated as a semi-infinite optimization problem, as follows:

$$\min_{T\in S}\sum_{i=1}^{5} t_i \tag{52}$$

subject to

$$-a_k \le \tau_k^i(s;T) \le a_k, \quad \forall s \in [0,1], \tag{53}$$
$$-b_k \le \dot{\tau}_k^i(s;T) \le b_k, \quad \forall s \in [0,1], \tag{54}$$

for all $k = 1, 2$, and all $i = 0, 1, \dots, 5$. Constraints in (53) represent actuator torque limits at each joints, and in (54) 'jerk' limits at each joints.

In [3], a hybrid algorithm specially designed to compute the global optimal solution of the above problem was used. In this paper, we discretize the above semiinfinite programming problem and use nonlinear programming packages to obtain an approximate solution.

We discretize the interval [0, 1] into N subintervals, and define

$$s_j = \frac{j}{N}. \tag{55}$$

Finally, the constraints involving the continuous variable s in (53) and (54) are replaced by the set of discrete constraints:

$$-a_k \le \tau_k^i(s_j;T) \le a_k,$$
$$-b_k \le \dot{\tau}_k^i(s_j;T) \le b_k, \tag{56}$$

for all $i = 1, 2, \ldots, 5$, $j = 1, 2, \ldots, N$, and $k = 1, 2$.

For numerical experiments, we use the parameter values in [3] ($m_1 = 15$, $m_2 = 7$, $l_1 = 1$, $l_2 = 0.5$, $a_1 = 260$, $a_2 = 50$, $b_1 = 300$, $b_2 = 200$) except for the number of given via points. We use only four via points ($n = 4$), and therefore the number of variables is five. We set $N = 8$, and the number of nonlinear inequality constraints is 320 ($= N \times (5) \times$ number of nonlinear functional inequality constraints $\times 2$). The initial condition T_0 is set to $5 \cdot 1_{5\times 1}$.

Tables 11–14 show our numerical results. Note that NPSOL failed to solve the trajectory planning problem with the full set of constraints, hence $\%_{\text{nat}}$ is not available. However, enhanced by our procedure, NPSOL found an optimal solution. With the enhanced Schittkowski SQP algorithm, the best result was achieved with data set 2, which required about 1/7 of the CPU time used by the native Schittkowski SQP algorithm. With the enhanced SNOPT algorithm, the best result was achieved using data set 3, which required about 1/3 of the CPU time used by the native SNOPT algorithm. Note the poor result for SNOPT for $N_{\text{iter}} = 10$, in Table 13, which is caused by the large setup cost for this problem. In Table 14, the best result using Algorithm 2.1, with IPOPT, was achieved using data set 1, which required about 1/4 of the CPU time used by the native IPOPT.

Table 10 Result using the native KNITRO, Kautz filter design example

Solver	f^0	N_{grad}	t_{CPU}	θ^*
KNITRO	1.032e-04	122220	156.18	−4.42e-05

Table 11 External active-set strategy with Schittkowski SQP, robot arm example

Data #	N_{iter}	i_T	f^0	N_{grad}	$\lvert Q\rvert$	i_{stab}	t_{CPU}	$\%_{\text{nat}}$	θ^*
1	10	10	2.2830	693	10	10	67.2	19.6	−2.15e-13
2	20	11	2.2830	924	11	11	44.6	13.0	−8.55e-14
3	30	11	2.2830	1011	11	11	49.6	14.5	−2.75e-14
Native			2.2830	28800	320		341.8	100	−4.45e-014

Table 12 External active-set strategy with NPSOL, robot arm example

Data #	N_{iter}	i_T	f^0	N_{grad}	$\lvert Q\rvert$	i_{stab}	t_{CPU}	$\%_{\text{nat}}$	θ^*
1	10	13	2.2830	811	12	13	35.8		−2.77e-13
2	20	10	2.2830	644	10	10	30.0		−2.51e-13
3	30	10	2.2830	698	10	10	32.4		−2.19e-13
Native					320				

Table 13 External active-set strategy with SNOPT, robot arm example

Data #	N_{iter}	i_T	f^0	N_{grad}	$\lvert Q\rvert$	i_{stab}	t_{CPU}	$\%_{\text{nat}}$	θ^*
1	10	152	2.2830	14773	15	152	622.1	380.1	−4.52e-09
2	20	17	2.2830	2609	10	17	103.5	63.2	−5.72e-09
3	30	10	2.2830	1203	10	10	51.9	31.7	−5.72e-09
Native			2.2830	14720	320		163.6	100	−1.47e-13

Table 14 External active-set strategy with IPOPT, robot arm example

Data #	N_{iter}	i_T	f^0	N_{grad}	$\lvert Q\rvert$	i_{stab}	t_{CPU}	$\%_{\text{nat}}$	θ^*
1	10	11	2.2830	415	10	11	24.2	34.7	−2.94e-09
2	20	11	2.2830	593	10	11	29.6	42.4	−2.94e-09
3	30	11	2.2830	718	10	11	35.5	50.9	−2.94e-09
Native			2.2830	6080	320		69.7	100	−8.92e-09

3.4 Optimal Control of a Quadratically Constrained Particle

This is a convex problem, which was inspired by the Problem 11.5.7 of [25]. The problem is to control the motion of a particle in a plane under a constant force field, subject to control and trajectory constraints. The motion of the particle is described by the following differential equation.

$$\frac{d}{dt}\begin{bmatrix} x_1(t) \\ x_2(t) \\ x_3(t) \\ x_4(t) \end{bmatrix} = \begin{bmatrix} x_2(t) \\ u_1(t)+d_1 \\ x_4(t) \\ u_2(t)+d_2 \end{bmatrix}, \tag{57}$$

where (x_1, x_3) denotes the position coordinates of the particle, (x_2, x_4) the velocity vector, u_1 and u_2 are control inputs, and d_1 and d_2 are constant disturbance force elements in x_1 and x_2 directions as shown in Fig. 4. The cost functions $f^0(\cdot)$ is defined by

$$f^0(u) = \frac{1}{2}\int_0^T [u_1^2(\tau) + u_2^2(\tau)]d\tau, \tag{58}$$

where $u \overset{\triangle}{=} (u_1, u_2)$. The constraints are

$$x_2^2(t) + x_4^2(t) \le v_{\max}^2, \tag{59}$$

$$u_1^l \le u_1(t) \le u_1^u,$$

$$u_2^l \le u_2(t) \le u_2^u, \tag{60}$$

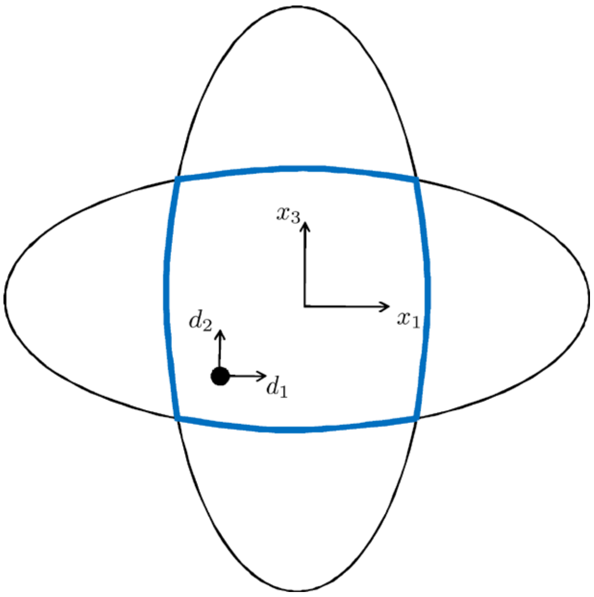

Fig. 4 Control of a particle on a plane under a constant force field

Table 15 Result using the native KNITRO, robot arm example

Solver	f^0	N_{grad}	t_{CPU}	θ^*
KNITRO	2.2830	5440	80.5	−1.62e-05

and

$$\frac{x_1^2(t)}{a^2} + \frac{x_3^2(t)}{b^2} \le 1,$$
$$\frac{x_1^2(t)}{b^2} + \frac{x_3^2(t)}{a^2} \le 1 \tag{61}$$

for all $t \in [0, T]$, where $a > b > 0$. Note that this is a linear-quadratic optimal control problem with linear and quadratic constraints, and hence it is convex.

The continuous-time optimal control problem described above was converted to a discrete-time, finite-dimensional optimal control problem using the discretization procedure used in Sect. 3.1. In our numerical experiments, we set $d_1 = 0.3$, $d_2 = 0.3$, $u_1^l = u_2^l = -2$, $v_{\max} = 2$, $u_1^u = u_2^u = 2$, $a = 5$, $b = 3$, and $T = 10$. The time $0 \le t \le T$ was discretized into $N = 128$ intervals. In the numerical experiments, we kept the box constraints (60) in the computations all the time, and applied our algorithm only to nonlinear constraints, discretized versions of (59) and (61), since our main interest is to reduce the computation time by excluding computationally expensive inactive constraint gradients from the computation. Therefore $N \times 3 = 384$ nonlinear constraints were considered in the following numerical experiments. As initial values, we used $x_0 = (0, 0, 0, 0)$ and $u_0 = -0.5 \times 1_{2\times N}$.

In Table 16, the best result using the enhanced Schittkowski SQP was achieved with data set 1, which required about 1/7 of the CPU time used by the native Schit-

Table 16 External active-set strategy with Schittkowski SQP, particle example

Data #	N_{iter}	i_T	f^0	N_{grad}	$\lvert Q\rvert$	i_{stab}	t_{CPU}	$\%_{\text{nat}}$	θ^*
1	10	2	3.1198e-01	190	12	2	3.6	14.1	−4.68e-10
1	20	2	3.1198e-01	421	13	2	6.6	25.9	−2.76e-10
1	30	2	3.1198e-01	517	13	2	8.9	34.9	−2.09e-09
Native			3.1198e-01	7296	384		25.5	100	−1.70e-10

Table 17 External active-set strategy with NPSOL, particle example

Data #	N_{iter}	i_T	f^0	N_{grad}	$\lvert Q\rvert$	i_{stab}	t_{CPU}	$\%_{\text{nat}}$	θ^*
1	10	2	3.1198e-01	209	11	2	2.1	3.1	−1.84e-12
2	20	2	3.1198e-01	268	13	2	2.5	3.8	−5.16e-16
3	30	2	3.1198e-01	268	13	2	2.5	3.7	−5.16e-16
Native			3.1198e-01	24192	384		67.4	100	−6.47e-12

Table 18 External active-set strategy with SNOPT, paricle example

Data #	N_{iter}	i_T	f^0	N_{grad}	$\lvert Q\rvert$	i_{stab}	t_{CPU}	$\%_{\text{nat}}$	θ^*
1	10	2	3.1198e-01	276	13	2	2.3	3.6	−6.17e-08
1	20	2	3.1198e-01	276	13	2	2.3	3.6	−6.17e-08
1	30	2	3.1198e-01	276	13	2	2.3	3.6	−6.17e-08
Native			3.1198e-01	21888	384		62.4	100	−6.42e-10

Table 19 External active-set strategy with IPOPT, particle example

Data #	N_{iter}	i_T	f^0	N_{grad}	$\lvert Q\rvert$	i_{stab}	t_{CPU}	$\%_{\text{nat}}$	θ^*
1	10	2	3.1198e-01	126	13	2	1.4	7.1	−9.11e-09
2	20	2	3.1198e-01	190	13	2	2.0	9.9	−9.14e-09
3	30	2	3.1198e-01	190	13	2	2.0	10.0	−9.14e-09
Native			3.1198e-01	5760	384		19.8	100	−9.14e-09

tkowski SQP algorithm. With enhanced NPSOL, the reduction was more significant, and a locally optimal solution was obtained using about $1/33$ of the CPU time used by the native NPSOL (data set 1 in Table 17). With enhanced SNOPT, about $1/27$ of the time used by the native SNOPT was required, irrespective of the choice on the parameter N_{iter} in Table 18. With enhanced IPOPT, about $1/14$ of the time used by the native SNOPT was used, with data set 1 in Table 19.

Table 20 Result using the native KNITRO, particle example

Solver	f^0	N_{grad}	t_{CPU}	θ^*
KNITRO	3.1198e-01	15744	45.8	−1.86e-08

Table 21 Computation time comparison of solvers: C = conSolve, N = NPSOL, S = SNOPT, I = IPOPT, K = KNITRO, NAT = Native, ENH = Enhanced

	Example 3.1		Example 3.2		Example 3.3		Example 3.4	
RANK	NAT	ENH	NAT	ENH	NAT	ENH	NAT	ENH
1	S	I	K	C	I	I	I	I
2	I	S	S	I	K	N	C	N
3	K	N	I	S	S	C	K	S
4	N	C		N	C	S	S	C
5	C						N	

Finally, in Table 20, we display the results for the native form of KNITRO. We see that on this problem, it is faster than native NPSOL and SNOPT. But slower than all of our enhanced versions, and also slower than native Schittkowski SQP and IPOPT.

4 Conclusions

We have presented an external active-set strategy for solving nonlinear programming problems with large numbers of inequality constraints, such as discretized semi-infinite optimization and optimal control problems, using nonlinear programming solvers. Our numerical results show that this strategy can result in considerable savings in computing time, with the computing time reduction depending on the fraction of constraints active at a solution, as well as the choice of algorithm parameters. We have tested the use of our strategy with four solvers and in the process produced a comparative evaluation of these solvers, which we present in Table 21. IPOPT appears to give the best results with and without our strategy and would be our first choice. However, the other algorithms that we tested are easily available via TOMLAB for use with MATLAB, while IPOPT has to be compiled from available blocks of code for use with MATLAB.

To conclude, the advantages of our active-set strategy are that it is very simple to implement and, at least as far as our numerical experiments indicate, it leads to a considerable reduction in computing time over the use of standard optimization code in "native" form.

References

1. Bhatti, M.A., Pister, K.S., Polak, E.: Optimization of control devices in base isolation systems for aseismic design. In: Leipholz, H.H.E. (ed.) Structural Control: Proceedings of the International Iutam Symposium On Structural Control, pp. 127–138. North-Holland, Amsterdam (1980)
2. Polak, E., Meeker, G., Yamada, K., Kurata, N.: Evaluation of an active variable-damping-structure. Earthquake Eng. Struct. Dyn. **23**, 1259–1274 (1994)
3. Bianco, C.G.L., Piazzi, A.: Minimum-time trajectory planning of mechanical manipulators under dynamic constraints. Int. J. Control **75**(13), 967–980 (2002)
4. Görner, S., Potchinkov, A., Reemtsen, R.: The direct solution of nonconvex nonlinear fir filter design problems by a SIP method. Optim. Eng. **1**, 123–154 (2000)
5. Durrant-Whyte, H., Pagac, D., Rogers, B., Stevens, M., Nelmes, G.: Field and service applications—an autonomous straddle carrier for movement of shipping containers—from research to operational autonomous systems. IEEE Robot. Autom. Mag. **14**(3), 14–23 (2007)
6. Rembold, U., Lueth, T., Ogasawara, T.: From autonomous assembly robots to service robots for factories. In: Proceedings of the IEEE/RSJ/GI International Conference on Intelligent Robots and Systems, vol. 3, pp. 2160–2167 (1994)
7. Shim, D.H., Chung, H., Sastry, S.: Conflict-free navigation in unknown urban environments. IEEE Robot. Autom. Soc. Mag. **13**, 27–33 (2006)
8. Upcroft, B., Moser, M., Makarenko, A., Johnson, D., Donikian, A., Alempijevic, A., Fitch, R., Uther, W., Grøtli, E.I., Biermeyer, J., Gonzalez, H., Templeton, T., Srini, V.P., Sprinkle, J.: Darpa urban challenge technical paper: Sydney-Berkeley driving team. Tech. rep., University of Sydney; University of Technology, Sydney; University of California, Berkeley (June 2007)
9. Murray, W., Gill, P.E., Saunders, M.A.: SNOPT: An SQP algorithm for large-scale constrained optimization. SIAM J. Optim. **12**, 979–1006 (2002)
10. Gill, P.E., Murray, W., Saunders, M.A., Wright, M.H.: User's guide for NPSOL 5.0: A Fortran package for nonlinear programming. Tech. Rep. SOL 86-2, Systems Optimization Laboratory, Department of Operations Research, Stanford University (1998)
11. Byrd, R.H., Nocedal, J., Waltz, R.A.: KNITRO: An integrated package for nonlinear optimization. In: di Pillo, G., Roma, M. (eds.) Large-Scale Nonlinear Optimization, pp. 35–59. Springer, Berlin (2006)
12. Wächter, A., Biegler, L.T.: On the implementation of a primal-dual interior point filter line search algorithm for large-scale nonlinear programming. Math. Program. **106**(1), 25–57 (2006)
13. Schittkowski, K.: On the convergence of a sequential quadratic programming method with an augmented Lagrangian line search function. Tech. rep., Systems Optimization Laboratory, Stanford University (1982)
14. Lawrence, C.T., Zhou, J.L., Tits, A.L.: User's guide for CFSQP version 2.5: A C code for solving (large scale) constrained nonlinear (minimax) optimization problems, generating iterates satisfying all inequality constraints. Tech. Rep. TR-94-16r1, Institute for Systems Research, University of Maryland (1997)
15. Cheney, E.W., Goldstein, A.A.: Newton's method for convex programming and Tchebycheff approximation. Numer. Math. **1**, 253–268 (1959)
16. Polak, E.: Optimization: Algorithms and Consistent Approximations. Applied Mathematical Sciences, vol. 124. Springer, Berlin (1997)
17. Leineweber, D.: Efficient Reduced SQP Methods for the Optimization of Chemical Processes Described by Large Sparse DAE Models. Fortschritt-Berichte VDI Reihe 3, vol. 613. VDI Verlag, Düsseldorf (1999)
18. Polak, E., Womersley, R.S., Yin, H.X.: An algorithm based on active sets and smoothing for discretized semiinfinite minimax problems. J. Optim. Theory Appl. **138**(2), 311–328 (2008)
19. Chung, H., Polak, E., Sastry, S.S.: An external active-set strategy for solving optimal control problems. IEEE Trans. Automat. Contr. **54**(5), 1129–1133 (2009)
20. John, F.: Extremum problems with inequalities as side conditions. In: Friedrichs, K.O., Neugebauer, O.W., Stoker, J.J. (eds.) Studies and Essays: Courant Anniversary Volume, pp. 187–204. Interscience Publishers, New York (1948)
21. Polak, E., Chung, H., Sastry, S.S.: An external active-set strategy for solving optimal control problems. Tech. Rep. UCB/EECS-2007-90, EECS Department, University of California, Berkeley (Jul 2007)
22. Holmström, K., Göran, A.O., Edvall, M.M.: User's Guide for TOMLAB. Tomlab Optimization Inc. (December 2006)

23. Schenk, O., Gärtner, K.: Solving unsymmetric sparse systems of linear equations with PARDISO. J. Future Gener. Comput. Syst. **20**(3), 475–487 (2004)
24. Mayne, D.Q., Rawlings, J.B., Rao, C.V., Scokaert, P.O.M.: Survey paper: Constrained model predictive control: Stability and optimality. Automatica **36**, 789–814 (2000)
25. Lewin, J.: Differential Games: Theory and Methods for Solving Game Problem with Singular Surfaces. Springer, Berlin (1994)