On the Off-Label Use of Outer Approximations: An External Active Set Strategy

Hoam Chung
Elijah Polak
S. Shankar Sastry

Electrical Engineering and Computer Sciences
University of California at Berkeley

Technical Report No. UCB/EECS-2007-127
http://www.eecs.berkeley.edu/Pubs/TechRpts/2007/EECS-2007-127.html

October 26, 2007

Acknowledgement

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 Marcus Edvall at TOMLAB Optimization Inc. for his advice on our codes. This work is supported by ARO SWARMS (W911NF-0510219) and ARO Phase II STTR (W911NF-06-C-0192)

On the Off-Label Use of Outer Approximations: An External Active Set Strategy

H. Chung, E. Polak, and S. Sastry [1]

Abstract

Outer approximations are a well known technique for solving semi-infinite 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. As our numerical results show, the effect of this external active-set strategy can be spectacular, with reductions in computing time by a factor that can exceed 500.

Key Words. Outer approximations, inequality constrained optimization, active set strategies.

[1]Authors are with the Department of Electrical Engineering and Computer Science, University of California, Berkeley, CA, 94720-1770 USA, hachung/polak/sastry@eecs.berkeley.edu.

1 Introduction

Outer approximations algorithms are used to solve semi-infinite optimization problems of the form

$$\mathbf{P}_Y \qquad \min_{x \in \mathbb{R}^n} \{f(x) | \phi(x, y) \leq 0, y \in Y\}, \qquad (1)$$

where $Y \subset \mathbb{R}^m$ is compact (with $m = 1$ most frequently) and $f : \mathbb{R}^n \to \mathbb{R}$ and $\phi : \mathbb{R}^n \times \mathbb{R}^m \to q$ continuously differentiable. In the most elementary form, these algorithms construct a sequence of finitely constrained problems of the form (see [8])

$$\mathbf{P}_{Y_i} \qquad \min_{x \in \mathbb{R}^n} \{f(x) | \phi(x, y) \leq 0, y \in Y_i\}, \qquad (2)$$

$i = 0, 1, 2, \ldots$, where $Y_0 \subset Y$ is arbitrary, and for $i \geq 1$,

$$Y_{i+1} = Y_i \cup \hat{Y}_{i-1}, \qquad (3)$$

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 $\mathbf{P}_{Y_{i-1}}$. It is straightforward to show that any accumulation point $\hat{x}$ of the optimizers $\hat{x}_i$ is an optimizer of $\mathbf{P}_Y$. Since the constraint set of $\mathbf{P}_Y$ is a subset of the constraint set of $\mathbf{P}_{Y_i}$, for all i, the naming of this approach "method of outer approximations" is obvious.

A characteristic of methods of outer approximations is that they, eventually, generate optimization problems with very large numbers of inequality constraints. Unfortunately, standard nonlinear programming packages, with the exception of CFSQP [5] including the excellent set found in TOMLAB [2], including SNOPT [7], NPSOL [1], Schittkowski SQP [10], and KNITRO[2] [3], do not incorporate effective active-set strategies that can alleviate the computational burden caused by large numbers of inequality constraints.

In this paper, we show that 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. Our numerical examples, drawn from real world applications, show that computing times can be reduced by a factor that can exceed 500, over the use of "raw" optimization packages. The effectiveness of our new active-set strategy increases as the number of constraints that are active at a solution

[2]KNITRO is a collection of optimization algorithms, and we use the algorithm option 'Interior/Direct' with quasi-Newton symmetric rank one updates in this paper.

decreases. Our new strategy is particularly effective when used with nonlinear programming solvers that allow warm starts. It may be useful to observe that a related strategy [9] for solving semi-infinite minimax problems using log-sum-exponential smoothing has proved to be equally effective.

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

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).* □

2 The Active-Set Strategy

Consider the inequality constrained minimization problem:

$$\mathbf{P_q} \qquad \min\{f^0(x) \mid f^j(x) \leq 0, j \in \mathbf{q}\}, \tag{4}$$

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

Next we define the index set $\mathbf{q}_\epsilon(x)$ with $\epsilon \geq 0$ by

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

where

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

and

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

Definition 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 of a sequence $\{x_i\}_{i=0}^{\infty}$, constructed according to the recursion* (8)*, is a feasible stationary point for $\mathbf{P_q}$.* □

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. **Data:** x_0, $\epsilon \geq 0$, $N_{iter} \in \mathbb{N}$

```
Step 0: Set $i = 0$, $\mathbf{Q}_0 = \mathbf{q}_\epsilon(x_0)$.
Step 1: Set $\xi_0 = x_i$ and perform $N_{iter}$ iterations of the form $\xi_{k+1} = A(\xi_k)$
on the problem
$\mathbf{P}_{\mathbf{Q}_i}$
        $\min\{f^0(\xi) | f^j(\xi) \leq 0,\ j \in \mathbf{Q}_i\}$        (9)
to obtain $\xi_{N_{iter}}$ and set $x_{i+1} = \xi_{N_{iter}}$.
Step 2: Compute $\psi(x_{i+1})$.
if $\xi_{N_{iter}}$ is returned as a global, local, or stationary solution of $\mathbf{P}_{\mathbf{Q}_i}$ and
$\psi(x_{i+1}) \leq 0$, then
    STOP,
else
    Compute
        $\mathbf{Q}_{i+1} = \mathbf{Q}_i \cup \mathbf{q}_\epsilon(x_{i+1})$,        (10)
    and set $i = i + 1$, and go to Step 1.
end if
```

Lemma 3. *Suppose that $\epsilon \geq 0$ and 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 $\mathbf{Q} = \cup_{i=0}^{\infty} \mathbf{q}_\epsilon(x_i) \subset \mathbf{q}$. For any $x \in \mathbb{R}^n$, let*

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

If $\psi_{\mathbf{Q}}(\hat{x}) \leq 0$, then $\psi(\hat{x}) \leq 0$.

Proof. Since the set $\mathbf{q}$ is finite, there must exist an i_0 such that $Q = \cup_{i=0}^{i_0} \mathbf{q}_\epsilon(x_i)$. Since $\mathbf{q}_0(x_i) \subset \mathbf{Q}$ for all $i \geq i_0$, it follows that $\psi(x_i) = \psi_{\mathbf{Q}}(x_i)$ for all $i \geq i_0$. Now, both $\psi(\cdot)$ and $\psi_{\mathbf{Q}}(\cdot)$ are continuous, and hence $\psi(\hat{x}) = \lim \psi(x_i) = \lim \psi_{\mathbf{Q}}(x_i) = \psi_{\mathbf{Q}}(\hat{x})$. The desired result now follows directly. □

Lemma 4. *Suppose that $\mathbf{Q} \subset \mathbf{q}$ and consider the problem*

$\mathbf{P}_{\mathbf{Q}}$
$$\min\{f^0(x) | f^j(x) \leq 0, j \in \mathbf{Q}\}. \tag{12}$$

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

(a) *If $\hat{x}$ is a global minimizer for $\mathbf{P}_{\mathbf{Q}}$, then it is also a global minimizer for $\mathbf{P}_{\mathbf{q}}$.*

(b) *If $\hat{x}$ is a local minimizer for $\mathbf{P}_{\mathbf{Q}}$, then it is also a local minimizer for $\mathbf{P}_{\mathbf{q}}$.*

(c) *If $\hat{x}$ is a stationary point for $\mathbf{P_Q}$, i.e., it satisfies the F. John conditions [4] (or Theorem 2.2.4, p. 188 in [8]), then it is also a stationary point for $\mathbf{P_q}$.*

Proof. Clearly, since $\hat{x}$ is feasible for $\mathbf{P_q}$ it is also feasible for $\mathbf{P_Q}$.
(a) Suppose that $\hat{x}$ is not a global minimizer for $\mathbf{P_q}$. Then there exists an x^* such that $f^j(x^*) \leq 0$ for all $j \in \mathbf{q}$ and $f^0(x^*) < f^0(\hat{x})$. Now, x^* is also feasible for $\mathbf{P_Q}$ and hence $\hat{x}$ cannot be a global minimizer for $\mathbf{P_Q}$, a contradiction.
(b) Suppose that $\hat{x}$ is not a local minimizer for $\mathbf{P_q}$. 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 $f^j(x_i) \leq 0$ for all i and $j \in \mathbf{q}$. But this contradicts the assumption that $\hat{x}$ is a local minimizer for $\mathbf{P_Q}$.
(c) Since $\hat{x}$ satisfies the F. John conditions for $\mathbf{P_Q}$, there exist multipliers $\mu^0 \geq 0$, $\mu^j \geq 0$, $j \in \mathbf{Q}$, 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 $\mathbf{P_q}$ with multipliers $\mu^j = 0$ for all $j \notin \mathbf{Q}$ and otherwise as for $\mathbf{P_Q}$. □

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

Theorem 5. *Suppose that the problem $\mathbf{P_q}$ has feasible solutions, i.e., there exist vectors x^* such that $f^j(x^*) \leq 0$ for all $j \in \mathbf{q}$.*

(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 $\mathbf{P_q}$, 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 $\mathbf{P_q}$. Then any accumulation point[3] $\hat{x}$ of this sequence is feasible and stationary for $\mathbf{P_q}$.*

[3]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 $i \xrightarrow{K} \infty$

Proof. **(a)** If sequence $\{x_i\}_{i=0}^{k}$ is finite, then, by the exit rule, it is feasible for $\mathbf{P_q}$ and it is a global, local, or stationary solution for $\mathbf{P}_{\mathbf{Q}_i}$. It now follows from Lemma 4, that it is also a global, local, or stationary solution for $\mathbf{P_q}$.
(b) Since the sets $\mathbf{Q}_i$ grow monotonically, and since $\mathbf{q}$ is finite, there must exist an i_0 and a set $\mathbf{Q} \subset \mathbf{q}$, such that $\mathbf{Q}_i = \mathbf{Q}$ for all $i \geq i_0$. Next, it follows from the fact that $A(\cdot)$ is convergent, that for any accumulation point $\hat{x}$, $\psi_{\mathbf{Q}}(\hat{x}) \leq 0$ and hence, from Lemma 3 that $\psi(\hat{x}) \leq 0$, i.e., that $\hat{x}$ is a feasible point for $\mathbf{P_q}$. It now follows from the fact that $A(\cdot)$ is convergent and Lemma 4 that any accumulation point $\hat{x}$ is stationary for $\mathbf{P_q}$. $\square$

3 Numerical Results

We will now present three numerical examples. One involving the control of drones (also known as unmanned aerial vehicles, or UAV's), one arising in the development of integration formulas on the sphere, and one involving a Kautz filter design.

All numerical experiments were performed using MATLAB V7.2 and TOMLAB V5.5 [2] in Windows XP, on a desktop computer with an Intel Xeon 3.2GHz processor with 3GB RAM. Optimization solvers tested in this paper were the Schittkowski SQP algorithm with cubic line search [10], NPSOL 5.02 [1], SNOPT 6.2 [7], and KNITRO [3].

It should be clear from the form of Algorithm 2, that our strategy benefits considerably from warm starts of the nonlinear programming solvers that are to be used after constructing the active set $\mathbf{Q}_i$. 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 $\mathbf{P}_{\mathbf{Q}_i}$ as initial conditions for solving the problem $\mathbf{P}_{\mathbf{Q}_{i+1}}$. SNOPT allows the user to provide initial variables and 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 an initial solution vector and initial Hessian matrix approximation. KNITRO allows the user to provide only the initial solution vector. For maximum efficiency, this data must be saved at the end of the $i-$th run and transmitted as initial data for the $i+1$-th run of the solver.

3.1 Control of Eight UAV's

This is a problem of controlling 8 identical UAV's which are required to fly inside a circle in a horizontal plane, without incurring a collision. Their controls are to be determined by a centralized computer, in a scheme known as Receding Horizon Control (see [6]).

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 yaw rate. The cost function for this problem is the sum of the energy used by the UAV's over the interval $[0, T]$, i.e.,

$$\sum_{i=1}^{8} \int_0^T u_i^2(\tau)d\tau. \tag{15}$$

The constraints, to be made explicit shortly, are those of staying inside a 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 (x_i^1, x_i^2, x_i^3) with a fourth component, x_i^4,

$$x_i^4(t) \triangleq \int_0^t \frac{T}{2} u_i^2(\tau)d\tau, \tag{16}$$

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

$$\frac{dx_i}{dt} = \begin{bmatrix} Tv\cos x_i^3 \\ Tv\sin x_i^3 \\ Tu_i \\ \frac{T}{2}u_i^2 \end{bmatrix} \triangleq h(x_i(t), u_i(t)) \tag{17}$$

with the initial state $x_i(0)$ given. We will denote the solution of the dynamic equation (17) by $x_i(t, u_i)$, with $t \in [0, 1]$. Let $u = (u_1, u_2, \ldots, u_8)$. The optimal control problem we need to solve is of the form

$$\min_{u \in L_\infty^8[0,1]} f^0(u) \triangleq \sum_{i=1}^{8} x_i^4(1, u_i) \tag{18}$$

subject to two sets of constraints:

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

$$f^i_{\text{bnd}}(t,u_i) \overset{\triangle}{=} x^1_i(t,u_i)^2 + x^2_i(t,u^i)^2 \leq r^2_{\text{bnd}},\ \forall t \in [0,1],\ i = 1,2,\ldots,8, \tag{19}$$

and
(b) Pairwise collision avoidance constraints:

$$\begin{aligned} f^{(i,j)}_{\text{ca}}(t,u_i,u_j) \overset{\triangle}{=} (x^1_i(t,u_i) - x^1_j(t,u^j))^2 + (x^2_i(t,u_i) - x^2_j(t,u_j))^2 \geq r^2_{\text{ca}}, \\ \forall t \in [0,1], i \neq j, i,j = 1,2,\ldots,8. \end{aligned} \tag{20}$$

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

$$\bar{x}_i(t_{k+1}) - \bar{x}_i(t_k) = \Delta h(\bar{x}_i(t_k),\bar{u}_i(t_k)),\ \bar{x}_i(0) = x_i(0),\ i = 0,1,2,\ldots,8 \tag{21}$$

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

$$\bar{u}_i \overset{\triangle}{=} (\bar{u}_i(t_0),\bar{u}(t_1),\ldots,\bar{u}_i(t_{N-1})), \tag{22}$$

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

$$\min_{\bar{u}_i \in \mathbb{R}^N,\ i\in\{1,\ldots,N_a\}} \bar{f}^0(\bar{u}) \overset{\triangle}{=} \sum_{i=1}^{8} \bar{x}^4_i(1,\bar{u}_i) \tag{23}$$

subject to $\bar{u}_i(t_k)| \leq b$ for $k = 0,1,\ldots,N-1$ and $i = 1,2,\ldots,8$, the discretized dynamics of each UAV (21) and the discretized stay-in-a-circle and non-collision constraints

$$\bar{f}^k_{\text{bnd},i}(\bar{u}^i) \overset{\triangle}{=} \bar{x}^1_i(t_k,\bar{u}_i)^2 + \bar{x}^2_i(t_k,\bar{u}_i)^2 \leq r^2_{\text{bnd}},\ k \in \{1,\ldots,N\}, \tag{24}$$

and

$$\begin{aligned} f^k_{\text{ca},(i,j)}(\bar{u}_i,\bar{u}_j) \overset{\triangle}{=} (\bar{x}^1_i(t_k,\bar{u}_i) - \bar{x}^1_j(t_k,\bar{u}_j))^2 + (\bar{x}^2_i(t_k,\bar{u}_i) - \bar{x}^2_j(t_k,\bar{u}_j))^2 \geq r^2_{\text{ca}}, \\ k \in \{1,\ldots,N\}, i \neq j,\ i,j = 1,2,\ldots,8. \end{aligned} \tag{25}$$

The total number of inequality constraints in this problem is $8N(8-1)/2 + 8N$. Clearly, (23), (24), and (25) is a mathematical programming problem which 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 $r_{\text{bnd}} = 4$, $r_{\text{ca}} = 1$, $T = 25$ and $N = 64$, resulting in 2304 nonlinear inequality constraints. The initial conditions and initial controls for each UAV are set as

$$\begin{aligned}
x_0^1 &= (2.5, 2.5, \pi, 0), & \bar{u}_0^1 &= -1.25 \times 10^{-1}\mathbf{1}_{1\times N} \\
x_0^2 &= (-2.5, 2, -\pi/2, 0), & \bar{u}_0^2 &= 1.25 \times 10^{-1}\mathbf{1}_{1\times N} \\
x_0^3 &= (-2.5, -2.5, -\pi/4, 0), & \bar{u}_0^3 &= 1.25 \times 10^{-1}\mathbf{1}_{1\times N} \\
x_0^4 &= (2, -2.5, \pi/2, 0), & \bar{u}_0^4 &= 2.50 \times 10^{-1}\mathbf{1}_{1\times N} \\
x_0^5 &= (2.5, 0, \pi/2, 0), & \bar{u}_0^5 &= 2.50 \times 10^{-1}\mathbf{1}_{1\times N} \\
x_0^6 &= (-2.5, 0, -\pi/2, 0), & \bar{u}_0^6 &= 1.25 \times 10^{-1}\mathbf{1}_{1\times N} \\
x_0^7 &= (0, 3, -3\pi/4, 0), & \bar{u}_0^7 &= 1.25 \times 10^{-1}\mathbf{1}_{1\times N} \\
x_0^8 &= (0, -3, \pi/4, 0), & \bar{u}_0^8 &= -2.50 \times 10^{-1}\mathbf{1}_{1\times N}.
\end{aligned} \tag{26}$$

The numerical results are summarized in the Tables 1–4. 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, are defined as follows:

$$\begin{aligned}
N_{grad} &= \sum_{i=0}^{i_T} |Q_i| \times \text{number of gradient function calls during } i\text{-th inner iteration} \\
t_{CPU} &= \sum_{i=0}^{i_T} \Big[\text{CPU time spent for } i\text{-th inner iteration} \\
&\qquad\qquad + \text{CPU time spent for setting up } i\text{-th inner iteration}\Big].
\end{aligned} \tag{27}$$

In the above, and in the tables, i_T is the value of the iteration index i at which Algorithm 2 is terminated by the termination tests incorporated in the optimization solver used, and i_{stab} is the value of index i at which $|\mathbf{Q}|$ is stabilized. Also, $\%_{RAW}$, the percentage of t_{CPU} with respect to the computation time with the *raw* algorithm, i.e. using the solver with the full set of constraints (shown in the last row of each table), is used in tables.

Fig. 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.

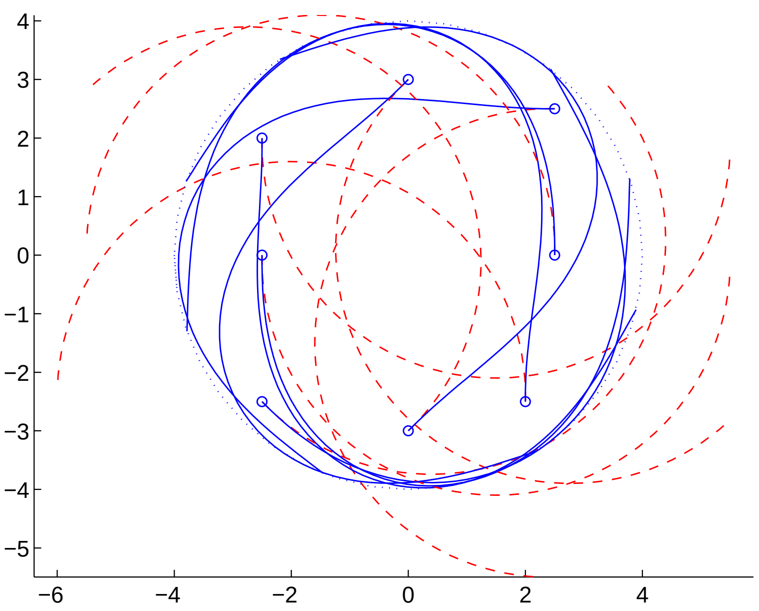

Figure 1: Initial trajectories (dashed red) and optimal trajectories (solid blue). Bounding circular region is represented by the dotted blue circle.

Table 1: External active-set strategy with Schittkowski SQP, eight-UAV example. The best result is marked with a '⋄' after the data number. '*' indicates that no meaningful data is available since the algorithm returns without an optimum after 100 iterations.

Data #	ϵ	N_{iter}	i_T	f^0	N_{grad}	$\lvert\mathbf{Q}\rvert$	i_{stab}	t_{CPU}	$\%_{RAW}$
01	1	10	74	10.635	1107859	1196	65	16573	55.1
02	1	20	22	2.0452	102133	258	22	1841.3	6.12
03	1	30	19	3.0388	112905	299	19	2192.4	7.28
04	0.1	10	71	3.2144	223197	334	70	3932.0	13.1
05	0.1	20	53	3.4327	221146	244	53	4212.3	14.0
06	0.1	30	42	2.9046	171308	190	42	3695.9	12.3
07⋄	0.01	10	64	2.8008	51016	91	64	1429.5	4.75
08	0.01	20	82	4.2251	266843	144	82	5687.7	18.9
09	0.01	30	100	*	*	*	*	*	*
Raw				4.1973	2642688	2304		30106	100

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

Data #	ϵ	N_{iter}	i_T	f^0	N_{grad}	$\lvert\mathbf{Q}\rvert$	i_{stab}	t_{CPU}	$\%_{RAW}$
01	1	10	17	2.1366	28392	214	16	397.3	0.50
02	1	20	14	2.1601	46012	229	14	716.7	0.90
03	1	30	12	2.1256	66952	239	12	1072	1.34
04	0.1	10	21	1.7028	11549	64	20	223.2	0.28
05	0.1	20	17	1.7028	18001	64	17	336.0	0.42
06	0.1	30	14	1.7028	19698	60	14	374.7	0.47
07$\diamond$	0.01	10	20	1.7028	5654	31	18	137.3	0.17
08	0.01	20	19	1.7028	11888	34	19	268.3	0.34
09	0.01	30	19	1.7028	15199	34	18	339.9	0.43
Raw				2.6809	7128576	2304		79817	100

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

Data #	ϵ	N_{iter}	i_T	f^0	N_{grad}	$\lvert\mathbf{Q}\rvert$	i_{stab}	t_{CPU}	$\%_{RAW}$
01	1	10	10	2.0874	10840	177	10	156.8	0.48
02	1	20	10	2.0897	12503	162	10	179.9	0.55
03	1	30	10	2.0840	14923	165	10	212.0	0.65
04	0.1	10	20	2.0838	12165	81	20	205.0	0.63
05	0.1	20	19	2.0838	13458	81	19	228.3	0.70
06	0.1	30	20	2.0838	14471	81	20	239.8	0.74
07$\diamond$	0.01	10	18	1.7028	3142	34	18	70.04	0.22
08	0.01	20	18	1.7028	3189	34	18	70.68	0.22
09	0.01	30	18	1.7028	3189	34	18	70.74	0.22
Raw				4.1381	3220992	2304		32425.7	100

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

Data #	ϵ	N_{iter}	i_T	f^0	N_{grad}	$\lvert\mathbf{Q}\rvert$	i_{stab}	t_{CPU}	$\%_{RAW}$
01	1	100	100	*	*	*	*	*	*
02	1	200	100	*	*	*	*	*	*
03	1	300	100	*	*	*	*	*	*
04	0.1	100	100	*	*	*	*	*	*
05	0.1	200	23	1.7028	183058	71	18	3494.8	7.81
06	0.1	300	15	1.7028	74569	67	15	1636.4	3.66
07	0.01	100	100	*	*	*	*	*	*
08$\diamond$	0.01	200	17	1.7028	32389	34	17	893.76	2.00
09	0.01	300	17	1.7028	32389	34	17	898.75	2.01
Raw				3.7896	4343040	2304		44728	100

These are all associated with staying in the circle; there are no active non-collision constraints. When properly adjusted, Algorithm 2 accumulates fewer than 40 constraints. Consequently, the reduction in the number of gradient computations is huge.

In Table 1, the best result using Algorithm 2, with the Schittkowski SQP defining the map $A(\cdot)$, was achieved with data set 7, which required about 1/20 of the CPU time used by the raw Schittkowski SQP algorithm. With NPSOL, the reduction was gigantic, and a locally optimal solution was obtained using about 1/500 of the CPU time used by NPSOL with the full constraint set (data set 7 in Table 2). When SNOPT was used as the map $A(\cdot)$ in Algorithm 2, the reduction about 1/400 was achieved with data set 7 in Table 3. Since the KNITRO TOMLAB interface does not support any warm start, Algorithm 2 with a small N_{iter} did not perform well. As shown in Table 4, when KNITRO was used as the internal solver, we used larger numbers of N_{iter}'s. With KNITRO, if N_{iter} is too small (data set 4 and 7), or ϵ is too big (data set 1–3), Algorithm 2 returned without a feasible stationary point after 100 outer iterations ($i_T = 100$). However, with proper values of N_{iter} and ϵ, reduction up to 1/20 in CPU time could be achieved.

To summarize, our overall fastest solution of the optimal control problem was obtained using data set 7 with the SNOPT algorithm. This computing time is 1/429 of the computing time of Schittkowski SQP, 1/462 of SNOPT, 1/1139 of NPSOL, and 1/638 of KNITRO, without using Algorithm 2.

3.2 Polynomial Interpolation on The Unit Sphere

The next numerical example is a semi-infinite minimax problem which arises in the polynomial interpolation on the unit sphere $\mathbb{S}^2$ [9]. The objective of the problem is to find sets of points $\eta_j \in \mathbb{S}^2,\ j = 1, \ldots, m$ that are good for polynomial interpolation and cubature. Let $x \in \mathbb{R}^n$ be the normalized spherical parameterization of η with $n = 2m - 3$. After discretization using nested icosahedral grids over $\mathbb{S}^2$, the fundamental system η can be found by solving the minimax problem involving Lagrangian sum of squares:

$$\min_{x \in \mathbb{R}^n} \max_{k \in K} \|G(\eta(x))^{-1} g(y_k; \eta(x))\|_2^2, \tag{28}$$

where $\eta \triangleq \{\eta_1, \eta_2, \ldots, \eta_m\}$ is the fundamental system of points on the unit sphere $\mathbb{S}^2$, $G(\eta) \in \mathbb{R}^{m \times m}$ is symmetric positive semi-definite basis matrix, $g : \mathbb{S}^2 \to \mathbb{R}$ is the reproducing kernel basis function.

In order to convert the minimax problem into a constrained nonlinear programming problem, we introduce the slack variable $x^{n+1} \in \mathbb{R}$ and define

$$\begin{aligned} f^0(\bar{x}) &\triangleq x^{n+1}, \quad \bar{x} = \begin{bmatrix} x \\ x^{n+1} \end{bmatrix} \\ f^k(\bar{x}) &\triangleq \|G(\eta(x))^{-1} g(y_k; \eta(x))\|_2^2 - x^{n+1}, \quad k \in K. \end{aligned} \tag{29}$$

The equivalent constrained minimization problem is now defined by

$$\min_{\bar{x} \in \mathbb{R}^{n+1}} x^{n+1} \tag{30}$$

subject to

$$f^k(\bar{x}) \leq 0, \quad k \in K. \tag{31}$$

We use $n = 29$ and the decision variable x is initialized with points obtained by maximizing $\log\det(G(\cdot))$ [11]. When our algorithm is applied to minimax type problems, the initial value of x^{n+1} should be carefully chosen so that the ϵ-active set of the equivalent nonlinear programming problem is nonempty in the beginning. Otherwise, the solver makes $x^{n+1} \to -\infty$ in the first inner iteration, and the solver gets stuck at wrong solution. In this paper, we set up the initial value of x^{n+1} such that

$$x_0^{n+1} = \max_{k \in K} \|G(\eta(x_0))^{-1} g(y_k; \eta(x_0))\|_2^2, \tag{32}$$

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

Tables 5–8 show the results with 2562 mesh points, i.e., $K = \{1, 2, \ldots, 2562\}$. Since many inequality constraints are in the ϵ-active set at a solution, the reduction in CPU time is much less than that in the previous example. In the cases of Schittkowski SQP and KNITRO, no performance enhancement was achieved by using Algorithm 2. However, with NPSOL, a locally optimal solution was obtained using about 1/5 of the CPU time used by NPSOL with the full set of constraints (data set 4 in Table 6). In Table 7, the best reduction achieved by using SNOPT as the map $A(\cdot)$ is about 40% (data set 3).

As the number of inequality constraints increased from 2562 to 10242, the Schittkowski SQP and SNOPT could not solve the problem with the full set of constraints due to a memory fault. Therefore, in Tables 9–12, the data field $\%_{RAW}$ is available only for NPSOL and KNITRO. Even though 10242 inequality constraints caused memory faults in Schittkowski SQP and SNOPT, Algorithm 2 enabled these solvers to find a solution, since, under our algorithm, only a certain portion of the full constraint set is considered, which results in a reduction of the total amount of memory required to solve the problem. NPSOL still achieved 3% - 60% reductions in computation time with certain parameter sets (data set 1 – 5 in Table 10). KNITRO did not show any enhancement in this case. Our conclusion is that Algorithm 2 may not work well on a problem with a large fraction of constraints active at a solution, if the internal solver does not provide a warm start scheme.

In brief, data set 7 with NPSOL (Table 6) achieved the fastest solution for the case with 2562 inequality constraints, and data set 1 with SNOPT (Table 11) for the case with 10242 inequality constraints.

3.3 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 semi-infinite minimax problem

$$\min_x \max_y \left| \log_{10}(|H(x,y)|) - \log_{10}(\tilde{F}(y)) \right|, \; x \in \mathbb{R}^{2N}, \; y \in [\varepsilon_1, 1 - \varepsilon_1], \quad (33)$$

Table 5: External active-set strategy with Schittkowski SQP, polynomial interpolation on a sphere, with 2562 inequality constraints

Data #	ϵ	N_{iter}	i_T	f^0	N_{grad}	$\lvert\mathbf{Q}\rvert$	i_{stab}	t_{CPU}	$\%_{RAW}$
01	1	10	100	*	*	*	*	*	*
02	1	20	4	3.5449	360280	1665	3	100.3	186.1
03	1	30	2	3.5449	207001	1663	2	57.99	107.8
04	0.1	10	26	3.5474	178619	439	24	63.68	118.3
05	0.1	20	25	3.5445	301395	423	25	109.9	204.3
06	0.1	30	15	3.545	224835	333	15	85.04	158.0
07	0.01	10	100	*	*	*	*	*	*
08	0.01	20	75	3.5484	318130	145	75	176.5	328.0
09	0.01	30	85	3.5466	553489	155	85	295.3	548.8
Raw$\diamond$				3.5449	179340	2562		53.81	100

Table 6: External active-set strategy with NPSOL, polynomial interpolation on a sphere, with 2562 inequality constraints

Data #	ϵ	N_{iter}	i_T	f^0	N_{grad}	$\lvert\mathbf{Q}\rvert$	i_{stab}	t_{CPU}	$\%_{RAW}$
01	1	10	3	3.5449	44734	1682	3	13.20	54.5
02	1	20	2	3.5449	47715	1662	2	12.94	53.4
03	1	30	1	3.5449	44109	1521	1	11.34	46.9
04$\diamond$	0.1	10	5	3.5449	11136	205	5	4.84	20.0
05	0.1	20	29	3.5445	258654	455	29	90.45	373.7
06	0.1	30	32	3.5445	416286	437	32	145.6	601.7
07	0.01	10	100	*	*	*	*	*	*
08	0.01	20	92	3.5450	244278	166	92	125.1	517.0
09	0.01	30	100	*	*	*	*	*	*
Raw				3.5449	97356	2562		24.20	100

Table 7: External active-set strategy with SNOPT, polynomial interpolation on a sphere, with 2562 inequality constraints

Data #	ϵ	N_{iter}	i_T	f^0	N_{grad}	$\|\mathbf{Q}\|$	i_{stab}	t_{CPU}	$\%_{RAW}$
01	1	10	3	3.5449	55992	1675	3	16.4	68.0
02	1	20	2	3.5449	64349	1663	2	18.1	74.8
03$\diamond$	1	30	1	3.5449	50193	1521	1	13.7	56.8
04	0.1	10	31	3.5448	117975	502	31	43.4	179.9
05	0.1	20	28	3.5448	208670	419	28	73.0	302.5
06	0.1	30	32	3.5447	368526	488	32	125.4	519.4
07	0.01	10	100	*	*	*	*	*	*
08	0.01	20	100	*	*	*	*	*	*
09	0.01	30	100	*	*	*	*	*	*
Raw				3.5449	87108	2562		24.14	100

Table 8: External active-set strategy with KNITRO, polynomial interpolation on a sphere, with 2562 inequality constraints

Data #	ϵ	N_{iter}	i_T	f^0	N_{grad}	$\|\mathbf{Q}\|$	i_{stab}	t_{CPU}	$\%_{RAW}$
01	1	100	18	3.5449	4126019	2504	17	1750	594.2
02	1	200	6	3.5446	2077923	2419	4	900.0	305.6
03	1	300	6	3.5441	3634181	2433	5	1560	529.8
04	0.1	100	100	*	*	*	*	*	*
05	0.1	200	100	*	*	*	*	*	*
06	0.1	300	95	3.5465	16117359	1551	95	7571	2570
07	0.01	100	100	*	*	*	*	*	*
08	0.01	200	100	*	*	*	*	*	*
09	0.01	300	100	*	*	*	*	*	*
Raw$\diamond$				3.5446	773724	2562		294.5	100

Table 9: External active-set strategy with Schittkowski SQP, polynomial interpolation on a sphere, with 10242 inequality constraints (Results with $\epsilon = 1$ are not available because of memory fault)

Data #	ϵ	N_{iter}	i_T	f^0	N_{grad}	$\lvert\mathbf{Q}\rvert$	i_{stab}	t_{CPU}	$\%_{RAW}$
01$\diamond$	0.1	10	9	3.5531	139065	957	9	66.1	
02	0.1	20	14	3.5542	552806	1421	14	247	
03	0.1	30	26	3.5540	1822533	1562	26	787	
04	0.01	10	100	*	*	*	*	*	
05	0.01	20	97	3.5533	996360	427	96	674.0	
06	0.01	30	67	3.5536	795553	278	67	626.2	
Raw						10242			

Table 10: External active-set strategy with NPSOL, polynomial interpolation on a sphere, with 10242 inequality constraints

Data #	ϵ	N_{iter}	i_T	f^0	N_{grad}	$\lvert\mathbf{Q}\rvert$	i_{stab}	t_{CPU}	$\%_{RAW}$
01	1	10	4	3.5542	391000	6746	4	167.8	93.6
02	1	20	2	3.5542	222742	6669	2	94.54	52.7
03	1	30	1	3.5542	261053	6071	1	108.2	60.4
04$\diamond$	0.1	10	13	3.5547	127068	1410	13	68.29	38.1
05	0.1	20	15	3.5542	342837	1393	15	173.2	96.6
06	0.1	30	29	3.5531	1275241	1550	29	620.2	346
07	0.01	10	93	3.5531	275429	408	93	226.3	126
08	0.01	20	97	3.5534	654975	410	97	486.0	271
09	0.01	30	89	3.5529	695385	380	89	555.9	310
Raw				3.5542	542826	10242		179.3	100

Table 11: External active-set strategy with SNOPT, polynomial interpolation on a sphere, with 10242 inequality constraints (Results with $\epsilon = 1$ are not available because of memory fault)

Data #	ϵ	N_{iter}	i_T	f^0	N_{grad}	$\lvert\mathbf{Q}\rvert$	i_{stab}	t_{CPU}	$\%_{RAW}$
01◇	0.1	10	6	3.5551	44772	785	6	22.87	
02	0.1	20	30	3.5532	909132	1646	30	394.1	
03	0.1	30	28	3.5537	1294214	1662	28	552.5	
04	0.01	10	100	*	*	*	*	*	
05	0.01	20	100	*	*	*	*	*	
06	0.01	30	96	3.5536	900507	385	96	629.0	
Raw						10242			

Table 12: External active-set strategy with KNITRO, polynomial interpolation on a sphere, with 10242 inequality constraints

Data #	ϵ	N_{iter}	i_T	f^0	N_{grad}	$\lvert\mathbf{Q}\rvert$	i_{stab}	t_{CPU}	$\%_{RAW}$
01	1	100	100	*	*	*	*	*	*
02	1	200	31	3.5530	56142701	9946	22	36705	759
03	1	300	15	3.5536	36236311	9146	6	22998	476
04	0.1	100	100	*	*	*	*	*	*
05	0.1	200	100	*	*	*	*	*	*
06	0.1	300	99	3.5536	40554982	3489	98	27447	568
07	0.01	100	100	*	*	*	*	*	*
08	0.01	200	100	*	*	*	*	*	*
09	0.01	300	100	*	*	*	*	*	*
Raw◇				3.5446	7712226	10242		4834.3	100

subject to the constraints

$$\begin{aligned} x^{2k-1} - \frac{x^{2k}}{1-\varepsilon_2} - (1-\varepsilon_2) &\leq 0, \\ -x^{2k-1} - \frac{x^{2k}}{1-\varepsilon_2} - (1-\varepsilon_2) &\leq 0, \quad k = 1, \ldots, N, \varepsilon_2 \in (0,1), \\ x^{2k} - (1-\varepsilon_2)^2 &\leq 0, \end{aligned} \tag{34}$$

where

$$\begin{aligned} H(x,y) = \sum_{k=1}^{N} \Big[&(x^{N+2k+1}\ p^k (e^{i2\pi y} - 1) + x^{N+2k}\ q^k (e^{i2\pi y} + 1)) \\ &\cdot \frac{1}{1 + x^1 e^{i2\pi y} + x^2 e^{i4\pi y}} \prod_{l=1}^{k} \frac{x^{2l} + x^{2l-1} e^{i2\pi y} + e^{i4\pi y}}{1 + x^{2l+1} e^{i2\pi y} + x^{2l+2} e^{i4\pi y}} \Big], \end{aligned} \tag{35}$$

$$\begin{aligned} p^k &= \sqrt{\frac{(1-x^{2k})(1+x^{2k}-x^{2k-1})}{2}} \\ q^k &= \sqrt{\frac{(1-x^{2k})(1+x^{2k}+x^{2k-1})}{2}} \end{aligned}\ , \tag{36}$$

and

$$\tilde{F}(y) = \begin{cases} \frac{1}{\sqrt{2\pi y}} & y \in [\varepsilon_1, \frac{1}{2}] \\ \frac{1}{\sqrt{2\pi(1-y)}} & y \in [\frac{1}{2}, 1-\varepsilon_1]. \end{cases} \tag{37}$$

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

$$\bar{x} \triangleq \begin{bmatrix} x \\ x^{2N+1} \end{bmatrix}. \tag{38}$$

Then the original minimax problem becomes the nonlinear programming problem

$$\min_{\bar{x}} x^{2N+1} \tag{39}$$

subject to constraints (34), and

$$-x^{2N+1} \leq \log_{10}(|H(x,y_i)|) - \log_{10}(\tilde{F}(y_i)) \leq x^{2N+1}, i = 1, 2, \ldots, N_d. \tag{40}$$

The above nonlinear inequality constraints are mostly active at a minimum.

For numerical experiments, we set $\epsilon_1 = \epsilon_2 = 0.01$, $N = 20$, and $N_d = 63$, which results in $3 \times N + 2 \times N_d + 1 = 187$ inequality constraints. The initial condition is set as

$$x_0 = 0.7 \ \mathbf{1}_{2N\times 1}. \tag{41}$$

x_0^{2N+1} is set using same manner in (32), i.e.,

$$x_0^{2N+1} = \max_{i=1,\ldots,N_d} |\log_{10}(|H(x_0, y_i)|) - \log_{10}(\tilde{F}(y_i))|. \tag{42}$$

This is a highly ill-conditioned problem which proved to be a challenge for the solvers we were using. When solvers were required to solve the problem with the full set of constraints, Schittkowski SQP reported an error message, 'Too large penalty', and NPSOL 'Current point cannot be improved on'. SNOPT returned a solution after consuming a large amount of time ('raw' data set in Table 15) in comparison with KNITRO ('raw' data set in Table 16). Using Algorithm 2, with Schittkowski SQP and NPSOL as the map $A(\cdot)$, optimal solutions could be found, as shown in Table 13 and Table 14. In Table 15, SNOPT achieved a maximum 95% reduction in computation time. KNITRO was compatible with Algorithm 2 on this example, but no enhancement was obtained.

To summarize, 'raw' use of KNITRO achieved the fastest solution, but Algorithm 2 allowed Schittkowski SQP and NPSOL to find a solution, and enhanced the performance of SNOPT so that its computation time is comparable to the best case of KNITRO.

4 Conclusion

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 problems, using nonlinear programming solvers. Our numerical results show that this strategy results in considerable savings in computer time, with time reductions depending on the fraction of constraints active at a solution. Reductions of computer time well over 500 have been observed.

Table 13: External active-set strategy with Schittkowski SQP Algorithm, Kautz filter design example

Data #	ϵ	N_{iter}	i_T	f^0	N_{grad}	$\lvert\mathbf{Q}\rvert$	i_{stab}	t_{CPU}	$\%_{RAW}$
01	1	10	25	1.5357e-4	104408	173	16	155.8	
02	1	20	29	2.0326e-4	242488	172	20	344.7	
03	1	30	100	*	*	*	*	*	
04$\diamond$	0.1	10	30	1.1731e-4	66071	140	17	118.0	
05	0.1	20	61	2.9738e-4	296798	163	45	445.2	
06	0.1	30	75	1.7893e-3	568494	167	52	772.9	
07	0.01	10	100	*	*	*	*	*	
08	0.01	20	100	*	*	*	*	*	
09	0.01	30	100	*	*	*	*	*	
Raw						187			100

Table 14: External active-set strategy with NPSOL, Kautz filter design example

Data #	ϵ	N_{iter}	i_T	f^0	N_{grad}	$\lvert\mathbf{Q}\rvert$	i_{stab}	t_{CPU}	$\%_{RAW}$
01	1	10	100	*	*	*	*	*	
02	1	20	100	*	*	*	*	*	
03	1	30	100	*	*	*	*	*	
04	0.1	10	100	*	*	*	*	*	
05	0.1	20	100	*	*	*	*	*	
06	0.1	30	100	*	*	*	*	*	
07	0.01	10	100	*	*	*	*	*	
08$\diamond$	0.01	20	54	1.3408e-7	289889	130	23	491.4	
09	0.01	30	54	5.2361e-8	429368	135	23	697.9	
Raw						187			100

Table 15: External active-set strategy with SNOPT, Kautz filter design example

Data #	ϵ	N_{iter}	i_T	f^0	N_{grad}	$\lvert\mathbf{Q}\rvert$	i_{stab}	t_{CPU}	$\%_{RAW}$
01	1	10	15	1.442e-5	82337	171	14	117.8	4.77
02	1	20	11	2.242e-5	116207	170	9	162.7	6.59
03	1	30	17	3.010e-6	329064	178	10	429.1	17.4
04◇	0.1	10	36	4.638e-6	63408	153	29	94.19	3.81
05	0.1	20	29	1.701e-6	63213	149	28	98.80	4.00
06	0.1	30	40	2.664e-7	203058	156	39	286.2	11.6
07	0.01	10	57	1.666e-6	114975	142	40	185.6	7.52
08	0.01	20	61	3.024e-6	142929	144	42	231.0	9.35
09	0.01	30	67	4.482e-6	548091	150	57	823.7	33.4
Raw				1.934e-06	2102172	187		2469	100

Table 16: External active-set strategy with KNITRO, Kautz filter design example

Data #	ϵ	N_{iter}	i_T	f^0	N_{grad}	$\lvert\mathbf{Q}\rvert$	i_{stab}	t_{CPU}	$\%_{RAW}$
01	1	100	100	*	*	*	*	*	*
02	1	200	26	1.0781e-4	661792	186	25	936.6	1516
03	1	300	100	*	*	*	*	*	*
04	0.1	100	100	*	*	*	*	*	*
05	0.1	200	100	*	*	*	*	*	*
06	0.1	300	100	*	*	*	*	*	*
07	0.01	100	100	*	*	*	*	*	*
08	0.01	200	100	*	*	*	*	*	*
09	0.01	300	100	*	*	*	*	*	*
Raw◇				1.0471e-4	48546	187		61.78	100

Acknowledgment

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 Marcus Edvall at TOMLAB Optimization Inc. for his advice on our codes. This work is supported by ARO SWARMS (W911NF-0510219) and ARO Phase II STTR (W911NF-06-C-0192).

References

[1] P. E. Gill, W. Murray, Michael A. Saunders, and Margaret H. Wright. User's guide for NPSOL 5.0: A fortran package for nonlinear programming. Technical Report SOL 86-2, Systems Optimization Laboratory, Department of Operations Research, Stanford University, 1998.

[2] K. Holmström, A. O. Göran, and Marcus M. Edvall. *User's Guide for TOMLAB*. Tomlab Optimization Inc., December 2006.

[3] K. Holmström, A. O. Göran, and M. M. Edvall. *User's Guide for TOMLAB/KNITRO v5.1*. Tomlab Optimization Inc., April 2007.

[4] F. John. Extremum problems with inequlaities as side condtions. In K. O. Friedrichs, O. W. Neugebauer, and J. J. Stoker, editors, *Studies and Essays: Courant Anniversary Volume*, pages 187–204. Interscience Publishers, Inc., New York, 1948.

[5] C. T. Lawrence, J. L. Zhou, and A. L. Tits. 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. Technical Report TR-94-16r1, Institute for Systems Research, University of Maryland, 1997.

[6] D. Q. Mayne, J. B. Rawlings, C. V. Rao, and P. O. M. Scokaert. Survey paper: Constrained model predictive control: Stability and optimality. *Automatica*, Vol. 36, pp. 789–814, 2000.

[7] W. Murray, P. E. Gill, and M. A. Saunders. SNOPT: An SQP algorithm for large-scale constrained optimization. *SIAM Journal on Optimization*, 12:979–1006, 2002.

[8] E. Polak. *Optimization: Algorithms and Consistent Approximations*, Volume 124 of *Applied Mathematical Sciences.* Springer, 1997.

[9] E. Polak, R. S. Womersley, and H. X. Yin. An algorithm based on active sets and smoothing for discretized semi-infinite minimax problems. *Journal of Optimization Theory and Applications*, 2007. in press.

[10] K. Schittkowski. On the convergence of a sequential quadratic programming method with an augmented lagrangian line search function. Technical report, Systems Optimization laboratory, Stanford University, 1982.

[11] Robert S. Womersley and Ian H. Sloan. How good can polynomial interpolation on the sphere be? *Advances in Computational Mathematics*, 14:195–226, 2001.