A Regularized Inexact Penalty Decomposition Algorithm for Multidisciplinary Design Optimization Problems With Complementarity Constraints

Shen Lu
Department of Industrial and Enterprise Systems Engineering, University of Illinois, Urbana, IL 61801
e-mail: shenu2@illinois.edu

Harrison M. Kim
Department of Industrial and Enterprise Systems Engineering, University of Illinois, Urbana, IL 61801
e-mail: hmkim@illinois.edu

Economic and physical considerations often lead to equilibrium problems in multidisciplinary design optimization (MDO), which can be captured by MDO problems with complementarity constraints (MDO-CC)—a newly emerging class of problem. Due to the ill-posedness associated with the complementarity constraints, many existing MDO methods may have numerical difficulties solving this class of problem. In this paper, we propose a new decomposition algorithm for the MDO-CC based on the regularization technique and inexact penalty decomposition. The algorithm is presented such that existing proofs can be extended, under certain assumptions, to show that it converges to stationary points of the original problem and that it converges locally at a superlinear rate. Numerical computation with an engineering design example and several analytical example problems shows promising results with convergence to the all-in-one solution.

[DOI: 10.1115/1.4001206]

1 Introduction

The research area of multidisciplinary design optimization (MDO) has been intensively investigated during the last several decades. Particularly, rich contribution has been made concerning two fundamental issues in this research area: (1) the decomposition modeling of the multidisciplinary system and (2) the coordination among individual subsystems to efficiently achieve the overall optimum. Previous research has addressed the core components of static, single multidisciplinary system design with models in systems engineering, nonlinear programming, and multistage programming. Yet, the ever increasing intricacy and complexity of engineering design often requires generalization to more complex models to facilitate the notion of dynamic design and operations, such as system-of-systems as well as competitive agents. Specifically, this paper considers a generalization of multilevel MDO in which its lower-level constraints represent the solution sets of optimization or game-theoretic problems. Such settings include, for example, (1) iterative and interactive design decision making processes captured by hierarchical frameworks or game protocols [1–4]; (2) multimode system design where a system may undergo multiple operating conditions represented as lower-level constraints, but not necessarily satisfy all of them, i.e., switch among different modes; and (3) biddings among multiple teams or suppliers given upper-level system requirements [5,6]. These problems would fall under the setting of multidisciplinary design optimization problem with complementarity constraints (MDO-CC) [7]—a newly emerging class of problem, which relates the MDO with the areas of mathematical program with complementarity constraints (MPCC) [8].

1.1 The Relevance of Complementarity Constraints to Mechanical and Multidisciplinary Design Optimization. Although few works are reported introducing complementarity constraints (CCs) into mechanical and multidisciplinary design optimization, CC can actually be related to mechanical and multidisciplinary design optimization in many scenarios. The most straightforward scenario addresses complementarities in physical feasibility. For example, complementarity can be used to "switch" a system from one possible working mode, e.g., weather condition, to another; it also models the cases in structural design, where Hooke’s law is turned on/off beyond control of designer [9]. While these scenarios could as well be handled by introducing discrete variables, CC provides a useful alternative with potentially less computation under continuous setting. A potentially more significant application of CC in mechanical and multiple disciplinary design optimizations involves the economic aspects of design decision making. For instance, in many settings, a designer may need to consider the competition among a collection of convex agents (e.g., market players). This can be modeled by adding the sufficient first-order optimality conditions, in the format of CCs, as design constraints, leading to an MPCC or MDO-CC. Such a framework ensures that strategic concerns can be better captured within the design phase and further enables the promising research on integrating enterprise decision and engineering design [5,10–13].

The need to consider CC in mechanical and multidisciplinary design is further motivated by the procedural or philosophical aspects of design. Design decision making, especially product design in a complex enterprise context, may involve interaction among different parties, e.g., collaborative design teams, manufacturing teams and maintenance teams. In some settings, these parties are often competing for some resources, implying that the problem could be modeled as a noncooperative game [2,14,15]. In some settings, there is a definite notion of a leader and a follower; in particular, a designer makes a decision subject to subsystems competing for resources contingent on such a decision—a single-leader multifollower game, referred to as a Stackelberg game [3,16,17]. The equilibrium conditions of many games such as Nash game, Stackelberg game, etc., can be reformulated into...
MPCCs. For example, the equilibrium conditions of a Nash game, if exists, can be captured by an aggregation of the optimality conditions of all the competing agents, which can be represented as a set of CCs. Due to this reason, MPCC has been traditionally used as a numerical solver of these games [8]. With the aid of CC, these procedural aspects of design processes can be captured and included in mechanical and multidisciplinary design optimization.

1.2 Literature Review. MDO problems can be solved directly with so-called all-in-one (AIO) approaches, which handle all the variables in a single optimization problem. The implementation of the AIO approach is straightforward; however, it may become impractical, undesirable, or even impossible, as the complexity of the problem increases. As an alternative to the AIO approaches, decomposition-based MDO methods have been investigated extensively over the past few decades—for example, monotonicity-based decomposition method (MBDM) [18], concurrent subspace optimization [19], bilevel integrated system synthesis [20], collaborative optimization (CO) [21,22], the constraint margin approach [23], analytical target cascading (ATC) [24–26], penalty decomposition (PD) [27], and augmented Lagrangian decomposition (ALD) [28]. Solving MDO problems with decomposition approaches could be advantageous for many reasons: computationally, it breaks the AIO problems into smaller subproblems usually easier to solve; it also allows specialized algorithms to be applied to each subproblems. Organizationally, it keeps the individual disciplinary design optimizations as independent as possible with minimum amount of communication, making it possible to integrate existing disciplinary analysis codes at small expense.

Among all the variants of MDO, the quasi-separable MDO problem has gained particular attention during recent years. Many of the above-mentioned MDO methods (e.g., MBDM, CO, constraint margin approach, PD, and ALD) can be considered as the quasi-separable MDO problem. In addition to these, Lu et al. [29] proposed an ATC variant with local objectives under the context of multimode design optimization. Among these approaches, PD, ATC, and ALD have been shown to have formulations whose solutions satisfy the Karush–Kuhn–Tucker (KKT) conditions of the original problems under certain assumptions. Additionally, it is also shown that the PD algorithms (inexact PD and exact PD) converge locally at a superlinear rate.

MPCCs represent an active research area, which is not well connected to the MDO. An example of the complementarity constraint is given as follows:

\[ 0 \leq F(x) \perp G(x) \geq 0 \]  

where \( x \) represents the variables and \( F \) and \( G \) are multifunctions in \( \mathbb{R}^n \). Particularly, the symbol \( \perp \) indicates the following inequalities:

\[ F \geq 0, \quad G \geq 0, \quad G(x) \circ F(x, y) = 0 \]

where the symbol \( \circ \) represents the Hadamard product, i.e., the term-by-term product operation between two vectors: \( a \circ b = [a_1, a_2, \ldots, a_n] \circ [b_1, b_2, \ldots, b_n] = [a_1 b_1, a_2 b_2, \ldots, a_n b_n] \).

In order to solve an MPCC, one intuitive approach is to reformulate it into a nonlinear programming problem through replacing the CCs (Eq. (1)) with its equivalent inequality constraints (Eq. (2)). However, the resulting nonlinear program (NLP) usually fails to satisfy the linear independence constraint qualification (LICQ) [30] and the weaker Mangasarian–Fromovitz constraint qualification (MFCQ) [30] at every feasible point. The failure of these constraint qualifications may have important negative numerical implications: the multiplier set may be unbounded; the active constraint normals may be linearly dependent; and a linear relaxation of the reformulated nonlinear programming problem can become inconsistent arbitrarily close to a solution to the MPCC [31]. As a consequence, existing nonlinear programming techniques may have difficulties solving this type of problem.
regularized IPD formulation. In addition to the formulation manipulation, the connection among solutions of these formulations is also presented.

2.1 AIO Formulation. We consider a general quasi-separable MDO-CC with \( n \) subsystems, whose AIO formulation is given as follows:

\[
P_{\text{MDO-CC-AIO}}: \min_{y_{i1}, \ldots, y_{in}} \sum_{i=1}^{n} f_i(x_i, y_i) \]

subject to \( g_i(x_i, y_i) \leq 0, \quad \forall \ i = 1, \ldots, n \)
\( h_i(x_i, y_i) = 0, \quad \forall \ i = 1, \ldots, n \)
\( 0 \leq G_i(x_i, y_i) \perp F_i(x_i, y_i) \leq 0, \quad \forall \ i = 1, \ldots, n \)

where \( y \) represents a vector of linking variables shared by all the \( n \) subsystems; and \( x_i \) represents the vector of local variables only relevant to subsystem \( i \) (\( i = 1, \ldots, n \)). The problem is quasi-separable in that the system objective is the summation of the subsystem objectives \( f_i \), and that the subsystem objectives and constraints \( g_i, h_i, G_i, F_i \) are associated exclusively with subsystem \( i \), depending only on the linking variables and the subsystem’s local variables. The local complementarity constraint (\( \perp \) symbol) indicates that \( G_i \) and \( F_i \) are non-negative, and that \( [G_i][\perp F_i]=0 \) (shown in Fig. 2(a)) for \( j = 1, \ldots, p \), where \( G_i, F_i \) are multifunctions in \( \mathbb{R}^p \). We assume throughout this paper that the objective and constraint functions are three times continuously-differentiable. This assumption is required to derive the proposed RIPD algorithm (refer to Theorem 1).

In order to avoid the numerical difficulties associated with the CCs, we follow the regularization methods \([39]\), which replace a CC with non-negative constraints on both of the two components and an inequality constraint on the product of the two. Additionally, the constraint on component product is further relaxed by a positive scalar so that the relaxed problem satisfies the LICQ. The regularization scheme is illustrated in Fig. 2(a) shows the feasible space of the original CC, which coincides with the two non-negative axes; such a feasible space fail to satisfy the MFCQ in that none of its points satisfies inequalities (Eq. (2)) strictly \([40]\).

Figure 2(b) shows the feasible space of the regularized constraints, which resides between the two non-negative axes and the solid curve; its strict feasible region is nonempty for any positive scalar \( \tau \). The regularized AIO formulation is as follows:

\[
P_{\text{MDO-RegularAIO}(\tau)}: \min_{y_{i1}, \ldots, y_{in}} \sum_{i=1}^{n} f_i(x_i, y_i) \]

subject to \( g_i(x_i, y_i) \leq 0, \quad \forall \ i = 1, \ldots, n \)
\( h_i(x_i, y_i) = 0, \quad \forall \ i = 1, \ldots, n \)
\( G_i(x_i, y_i) \geq 0, \quad \forall \ i = 1, \ldots, n \)
\( F_i(x_i, y_i) \geq 0, \quad \forall \ i = 1, \ldots, n \)

where \( G_i(x_i, y_i) \circ F_i(x_i, y_i) \leq \tau \epsilon_i \), \( \forall \ i = 1, \ldots, n \)

where the symbol \( \circ \) represents the Hadamard product, i.e., the term-by-term product operation between two vectors: \( a \circ b = [a_1, \ldots, a_d]^T [b_1, \ldots, b_d]^T = [a_1 b_1, \ldots, a_d b_d]^T \). Additionally, \( \{t_k\} \) is a positive descent sequence that converges to zero, and \( \epsilon_i \) is a vector of unit elements. Note that as \( \{t_k\} \) approaches zero, the feasible space of \( P_{\text{MDO-RegularAIO}(\tau)} \) generally converges to that of the original AIO problem in Eq. (3).

2.2 Bilevel Decomposed Formulation. Implementation of the AIO problem is straightforward in general, but obtaining its solution is usually computationally difficult due to the problem size. An alternative to the AIO approach is the decomposition-based approaches \([41,42]\), where the original AIO problem is decomposed into a set of interrelated subproblems and solved through an iterative process of subproblem optimization and coordination among them. Using decomposition-based approach can be advantageous, as it reduces the size of individual problems by decomposition while limiting the communication among subproblems only to where necessary via linking variables.

As an initial step to decomposing (Eq. (4)), we introduce duplicated copies of the linking variables \( y \) in each subsystem \( i \) to separate local constraints. Also, additional constraints are introduced to ensure consistency among \( y \)’s. The resulted regularized, modified AIO problem is given as follows:

\[
P_{\text{MDO-ModAIO}(\gamma)}: \min_{y_{i1}, \ldots, y_{in}} \sum_{i=1}^{n} f_i(x_i, y_i) \]

subject to \( g_i(x_i, y_i) \leq 0, \quad \forall \ i = 1, \ldots, n \)
\( h_i(x_i, y_i) = 0, \quad \forall \ i = 1, \ldots, n \)
\( G_i(x_i, y_i) \geq 0, \quad \forall \ i = 1, \ldots, n \)
\( F_i(x_i, y_i) \geq 0, \quad \forall \ i = 1, \ldots, n \)
\( G_i(x_i, y_i) \circ F_i(x_i, y_i) \leq \tau \epsilon_i, \quad \forall \ i = 1, \ldots, n \)
\( c_i = y - y_i = 0, \quad \forall \ i = 1, \ldots, n \)

On the basis of Eq. (5), a second manipulation is applied so that the constraint consistent given as \( c_i = 0 \) relaxed, and the corresponding violations are penalized in the format of a quadratic penalty function. As a result of these manipulations, the regularized, relaxed AIO formulation (or simply relaxed AIO for convenience) is given as follows:

\[
P_{\text{MDO-RegularAIO}(\gamma_m)}: \min_{y_{i1}, \ldots, y_{in}} \sum_{i=1}^{n} \left( f_i(x_i, y_i) + \gamma_i \| y - y_i \|^2 \right) \]

subject to \( g_i(x_i, y_i) \leq 0, \quad \forall \ i = 1, \ldots, n \)
\( h_i(x_i, y_i) = 0, \quad \forall \ i = 1, \ldots, n \)
\( G_i(x_i, y_i) \geq 0, \quad \forall \ i = 1, \ldots, n \)
\( F_i(x_i, y_i) \geq 0, \quad \forall \ i = 1, \ldots, n \)
\( G_i(x_i, y_i) \circ F_i(x_i, y_i) \leq \tau \epsilon_i, \quad \forall \ i = 1, \ldots, n \)

where \( \gamma_m \) denotes the penalty parameter satisfying \( \{\gamma\} \rightarrow \infty \).

We note that Eq. (6) consists of \( n \) subsystems coupled through the linking variables \( y \) only. Therefore, it can be decomposed into a bilevel program through holding \( y \) constant at each subsystem. The resulted subsystems are presented in Eq. (8). In addition, the optimal subsystem objective values under given \( y \) setting are used to define a master problem, as shown in Eq. (7)
\[ P_{\text{RIPD-Master}}(t_k, \gamma_m) : \min_{y} \sum_{i=1}^{n} F_i'(y) \]  

where \( F_i'(y) \) is the optimal subsystem objective value given \( y \)

\[ P_{\text{RIPD-Sub}}(t_k, \gamma_m) : F_i'(y) = \min f_j(x, y) + \gamma_m\|y - y^0\|_2^2 \]

subject to \( g_i(x, y) \leq 0 \)

\[ h(x, y) = 0 \]

\[ G_i(x, y) \geq 0 \]

\[ F_i(x, y) \geq 0 \]

We call the above formulation (Eqs. (7) and (8)) the RIPD formulation to be differentiated from the IPD formulation [27].

2.3 Connection Among the Proposed Formulations. This subsection is devoted to mapping the stationarity conditions of the RIPD formulation (Eqs. (7) and (8)) to those of the AIO formulation (Eq. (3)). We employ the regularized, relaxed AIO problem (Eq. (6)) and the regularized AIO problem (Eq. (4)) as intermediate steps to facilitate this mapping. The flow of stationary point mapping is indicated by the dotted and dashed arrows in Fig. 1: an accumulative stationary point of Eqs. (7) and (8) is first mapped to a stationary point of Eq. (6) through Theorem 1; then a limit of these stationary points is mapped to a stationary point of Eq. (4) following Theorem 2; finally a limit of solutions to Eq. (4) is mapped to a strongly-stationary point of Eq. (3) through Theorem 3. While these theorems are already well-established in their respective areas of research, our contribution is to show that they can be combined in the context of a new type of problem, namely, the MDO-CC, to effectively derive convergence results. Before proceeding, we provide definitions for several concepts related to MPCC.

Definition 1 (strong-stationarity conditions for an MPCC)

\[ P_{\text{MPCC}} : \min f(x) \]

subject to \( g(x) \leq 0 \)

\[ h(x) = 0 \]

\[ F(x) - s = 0 \]

\[ G(x) - t = 0 \]

\[ 0 \leq s \perp t \geq 0 \]

a point \( z = (x, s, t) \) is strongly-stationary if and only if there exist multipliers \( (\mu, \lambda, \sigma_1, \sigma_2, v_1, v_2) \) satisfying

\[
\begin{pmatrix}
\nabla f & \nabla g & \nabla h & \nabla F & \nabla G
\end{pmatrix} 
\begin{pmatrix}
\mu \\
\lambda \\
\sigma_1 \\
\sigma_2 \\
v_1 \\
v_2
\end{pmatrix} \]

\[ = 0 \]

\[ 0 \leq \mu \perp -g(x) \geq 0 \]

\[ h(x) = 0 \]

\[ F(x) - s = 0 \]

\[ G(x) - t = 0 \]

\[ 0 \leq s \perp t \geq 0 \]

\[ \text{if } [s_j] = [t_j] = 0, \text{ then } [v_1] \geq 0 \text{ and } [v_2] \geq 0, \forall j \]

where \( \nabla \) denotes the Jacobian for a vector function.

Let \( A_1, A_2 \subseteq \{1, \ldots, m\} \) be the sets of indices corresponding to \( s \) and \( t \) respectively, which satisfy \( A_1 \cup A_2 = \{1, \ldots, m\} \). These sets can be employed to construct a relaxed nonlinear program (relaxed NLP)

\[ P_{\text{MPCC-RNL}} : \min_{x} f(x) \]

subject to \( g(x) \leq 0 \)

\[ h(x) = 0 \]

\[ F(x) - s = 0 \]

\[ G(x) - t = 0 \]

\[ 0 \leq s \perp t \geq 0 \]

\[ \text{if } [s_j] = [t_j] = 0, \text{ then } [v_1] \geq 0 \text{ and } [v_2] \geq 0, \forall j \]

The notion of strong-stationarity is intimately related to the relaxed NLP in that a point is a strongly-stationary solution of Eq. (9) if and only if it is a stationary point of Eq. (11) (see Proposition 4.1 of Ref. [31]).

In addition to the stationarity conditions, we also provide the definition of the following constraint qualification.

Definition 2 (MPCC-LICQ). The MPCC (Eq. (9)) satisfies MPCC linear independence constraint qualification if the relaxed NLP (Eq. (11)) satisfies LICQ.

The stationarity conditions of the RIPD formulation (Eqs. (7) and (8)) are mapped to those of the regularized, relaxed AIO formulation (Eq. (6)) following Theorems 4.9 and 4.8 of Ref. [27]. We provide these theorems in the context of the MDO-CC.

Theorem 1. Assume that \( f, g, h, G, F \) are three times continuously differentiable. For any \( t_k, \gamma_m \), let \((y(t_k, \gamma_m), y_1(t_k, \gamma_m), \ldots, y_n(t_k, \gamma_m), x_1(t_k, \gamma_m), \ldots, x_m(t_k, \gamma_m))\) be a local minimizer of Eqs. (7) and (8), which satisfies LICQ, strict complementarity slackness (SCS) and second-order sufficient conditions (SOSCs) for Eq. (8). Then \((y_1(t_k, \gamma_m), \ldots, y_n(t_k, \gamma_m), x_1(t_k, \gamma_m), \ldots, x_m(t_k, \gamma_m))\) is a first-order KKT point of Eq. (6) satisfying LICQ.

If, in addition, SCS and SOSC hold at \((y_1(t_k, \gamma_m), \ldots, y_n(t_k, \gamma_m), x_1(t_k, \gamma_m), \ldots, x_m(t_k, \gamma_m))\) for Eq. (6), then the objective of Eq. (7) \( F(y) \) is locally twice continuously-differentiable with respect to \( y \) in a neighborhood of \( y(t_k, \gamma_m) \). Also, \( y(t_k, \gamma_m) \) is a minimizer of \( F(y) \) satisfying SOSC.

The next theorem maps the limit of stationary solutions of the relaxed AIO formulation (Eq. (6)) to a stationary solution of the regularized AIO formulation (Eq. (4)). It follows from well-established results of penalty methods, for example, Theorem 17.1 in Ref. [30] and Theorem 17.2 in Ref. [43].

Theorem 2. For any \( t_0 \), let \((y_1(t_0), \ldots, y_n(t_0), x_1(t_0), \ldots, x_m(t_0))\) be a sequence of KKT points of Eq. (6) corresponding to a sequence of penalty parameters \( \gamma_0 \) with \( \gamma_0 \xrightarrow{} \infty \). If LICQ holds for each KKT point in the sequence, then any of its limit point \((y_1(t_2), \ldots, y_n(t_2), x_1(t_2), \ldots, x_m(t_2))\), at which LICQ holds for the equality constraints \( y(t_2) = y(t_2) \) and all
the active local constraints is a KKT point of Eq. (4) satisfying LICQ.

The following theorem maps the limit of stationary solutions of the regularized AIO formulation (Eq. (4)) to a strongly-stationary solution of the AIO formulation (Eq. (3)). It follows Theorem 3.1 of Ref. [39]. The original theorem derived so-called B-stationarity\(^1\) for Eq. (3), but it also made clear that the B-stationarity is equivalent to strong-stationarity under MPCC-LICQ.

Theorem 3. Let \(\{y(t), x_1(t), \ldots, x_n(t)\}_k\) be a sequence of KKT points of Eq. (4) corresponding to a sequence of regularization parameters \(\{\gamma_k\}\) with \(t_k \to 0\). Assume that the sequence converges to \(\{y, x_1, \ldots, x_n\}\), which satisfies MPCC-LICQ. Let

\[
P^k = \left\{ j \mid [G_j(x(t_k), y(t_k))]_i [F_i(x(t_k), y(t_k))]_j = 0 \right\}
\]

\[
P^0 = \left\{ j \mid [G_j(x, y)]_i = 0 \right\}
\]

Then \(\{y, x_1, \ldots, x_n\}\) is a strongly-stationary point of Eq. (3) if and only if for each subsystem \(i\), the limits of the Lagrange multipliers corresponding to \([G_j]_i\) and \([F_i]_j\), \([v_{i,j}]_i\) and \([v_{j,i}]_j\) are zero for \(j\) such that \(j \in P^0 \cap \overline{P}^k\).

As a summary of the results presented in this subsection, we present the following proposition, which establishes the connection between stationarity conditions of the RIPD problems (Eqs. (5) and (6)) to those of the AIO problem (Eq. (1)).

Proposition 1. Assume that \(f_2, g_3, h_4, G_5\), and \(F_6\) are three times continuously differentiable. For any \(t_k\), let

(i) \(\{y(t_k), y_1(t_k), \ldots, y_n(t_k), y_1(t_k), \gamma_2(t_k), \ldots, y_n(t_k), \gamma_n(t_k)\}\) be a sequence of local minimizers of Eqs. (7) and (8) corresponding to a sequence of penalty parameters \(\{\gamma_m\}\) with \(\gamma_m \to \infty\) such that LICQ, SCS, and SOSC are satisfied for Eq. (8) at any point of the sequence.

(ii) \(\{y(t_k), y_1(t_k), \ldots, y_n(t_k), x_1(t_k), \ldots, x_n(t_k)\}\) be a limit point of \(\{y(t_k), y_1(t_k), y_2(t_k), \ldots, y_n(t_k), x_1(t_k), \ldots, x_n(t_k)\}\) at which LICQ holds for the equality constraints \(y(t_k, \gamma) = y(t_k, \gamma)\) and all the active local constraints.

If the sequence of \(\{y(t_k), x_1(t_k), \ldots, x_n(t_k)\}\) corresponding to a sequence of regularization parameters \(\{t_k\}\) with \(t_k \to 0\) converges to \(\{y, x_1, \ldots, x_n\}\), which satisfies the assumptions of Theorem 3, then \(\{y, x_1, \ldots, x_n\}\) is a strongly-stationary point of Eq. (3).

Proof. For any \(t_k\), Theorem 1 indicates that any point in \(\{y(t_k, \gamma) y_1(t_k, \gamma), \ldots, y_n(t_k, \gamma), x_1(t_k, \gamma), \ldots, x_n(t_k, \gamma)\}\) is a KKT point of Eq. (6) satisfying LICQ. Therefore, Theorem 2 guarantees that \(\{y(t_k), y_1(t_k), \ldots, y_n(t_k), x_1(t_k), \ldots, x_n(t_k), y(t_k)\}\) is a KKT point of Eq. (4) under regularization parameter \(t_k\). Following this result, \(\{y, x_1, \ldots, x_n\}\) is a strongly-stationary point of Eq. (3) due to Theorem 3.

3 Solution Algorithm

In this section, we present a solution algorithm for the MDO-CC following the regularization approach and the IPD method.

The framework of the presented bilevel decomposed algorithm is given as a nested loop shown in Fig. 3: in the outer loop, the regularization approach is followed so that the regularization parameter \(t_k\) is gradually reduced; in the inner loop, the IPD algorithm is applied to solve the regularized problem under each \(t_k\)

\(^1\)A feasible point \(x\) of an MPCC is Bouligand stationary (B-stationary) if it is a local minimizer of the linearized MPCC, which is obtained by linearizing all data functions at point \(x\) [40].

**Fig. 3. The nested loop framework of the regularized IPD algorithm**

**Fig. 4. The single loop framework of the regularized IPD algorithm**

setting specified by the outer. Specifically, the IPD algorithm increases the penalty parameter \(\gamma_m\) for the deviation terms at the beginning of each inner loop iteration, and solves the master problem (Eq. (7)) under the specified \(\gamma_m\) value. This procedure directly follows Proposition 1. Therefore, it converges to the solution if the assumptions of Proposition 1 are satisfied.

While supported theoretically by Proposition 1, the nested loop framework may incur excessive computation due to costly inner loop iterations. In order to address this, an alternative strategy could be to employ looser inner loop termination criteria. Several approaches has been proposed to reduce the number of inner loop iterations in the context of MDO: Tosserams et al. [44] proposed to follow alternate direction methods of multipliers to reduce the number of inner loop iteration in ATC approach. Additionally, Li et al. [45] presented a truncated diagonal quadratic approximation of the ATC, which is closely related to Tosserams’ method. Note that the nested loop computation framework in these papers refers to the decomposition algorithm for MDO problems, which is actually the inner loop of our presented nested loop framework. As an extreme case of reducing the inner loop iterations, we present a single loop procedure in which the inner loop is terminated after just a single RIPD problem solution. In other words, the regularization parameter and the penalty parameter are updated together. The analytical convergence property of the presented single loop procedure is not yet established. However, numerical computation shows promising results with convergence to AIO solutions. The procedure of the single loop framework is presented in Fig. 4; and the results are presented in Sec. 4.
Modified BFGS algorithm

**BEGIN**

Step 0: Given $x_0$, convergence tolerance $\varepsilon_{\text{grad}}$ and initial Hessian approximation $B_0$; $k = 0$  

**Step 1**: **REPEAT**

1. Evaluate function $F(y)$ and gradient $\nabla F(y)$.

2. Calculate search direction: solve $B_k \Delta y_k = -\nabla F(y_k)$.

3. **WHILE** $F(y_k + a_k \Delta y_k) - F(y_k) > \alpha \Delta F(y_k)^T \Delta y_k$  

4. $a_k = \alpha k / 2$; Calculate $F(y_k + a_k \Delta y_k)$

**ENDWHILE**

4. $s_k = a_k \Delta y_k$, $\phi_k = \nabla F(y_k + a_k \Delta y_k)$  

5. $r_k = \phi_k - a_k B_k \phi_k$, $y_{k+1} = y_k + a_k \Delta y_k$.

6. $\alpha = 1 + \max \{ \frac{2 \phi_k^T s_k}{s_k^T r_k} \}$

7. **UNTIL** $\| \nabla F(y_k) \|_2 + F(y_k) < \varepsilon_{\text{grad}}$

**END**

**Fig. 5** The BFGS procedure

Goldfarb-Shanno (BFGS) method with backtracking line search [46], which builds an approximation of $F(y)$'s second derivative using its first gradient. A pseudocode for the algorithm is provided for the reader in Fig. 5: a numerical example is also included in the Appendix to illustrate the approach.

Note that the gradient of $F(y)$ is calculated with [27]

$$\nabla F(y) = 2 \gamma y - y$$

Although the gradient in Eq. (12) looks as if it is derived through directly differentiating $F(y)$ with respect to $y$, it actually considers the implicit functions such as $y_i(y)$ and $x_j(y)$, etc. The reader is referred to Ref. [27] for its detailed derivation.

3.1 The Convergence Procedure and Local Convergence Analysis

The convergence procedure of the nested loop framework is illustrated in Figs. 6(a) and 6(b). For demonstration purposes, we consider a simple example with two linking variables $y=[y_1, y_2]^T$ and present the convergence procedure of the linking variables only.

In Figs. 6(a) and 6(b), the shaded dots, denoted as $(y(t_0), y(t_1), ... , y(t_{L-1})), ...$, represent a sequence of stationary solutions to Eq. (4), corresponding to $(t_k)$. For each adjacent pair of points in this sequence $y(t_k)$ and $y(t_{k+1})$, the inner loop takes a sequence of stationary solutions to Eq. (6), corresponding to $(\gamma(t_i))$, to move from $y(t_k)$ to $y(t_{k+1})$. This sequence, denoted as $(y(t_{k+1,0}), y(t_{k+1,1}), ... , y(t_{k+1, m}), ...)$, is illustrated by the gray dots in Fig. 6(a).

Furthermore, for each adjacent pair of points in the sequence of stationary solutions to Eq. (6) $y(t_{k+1}, \gamma_{m})$ and $y(t_{k+1}, \gamma_{m+1})$, the BFGS algorithm takes a sequence of steps to converge from $y(t_{k+1}, \gamma_{m})$ to $y(t_{k+1}, \gamma_{m+1})$. This sequence, denoted as $(y^{(0)}(t_{k+1}, \gamma_{m}), y^{(1)}(t_{k+1}, \gamma_{m+1}), ... , y^{(L)}(t_{k+1}, \gamma_{m+1}))$, is illustrated by the solid dots in Fig. 6(b).

As discussed previously, we employ the BFGS method and the SQP as the master and subproblem optimizer respectively. Following this setting, standard local convergence property can be expected. For any $t_k$, we note that given certain assumptions, the implicit function theorem and Theorem 6 of Ref. [43] indicate that there exists a locally unique twice continuously-differentiable trajectory of stationary solutions to Eq. (6) $y(t_k, \gamma)$, which converges to $y(t_k)$. The following theorem shows that for each parameter setting (regularization and penalty parameter combined), the BFGS master problem solver converges locally at a superlinear rate to a stationary solution to Eq. (6). Also presented is a restatement of the superlinear convergence parameter of the SQP solver for each subproblem.

**Theorem 4** Assume that the assumptions of Proposition 1 hold. If for any $t_k$ $(y(t_k), x_1(t_k), ..., x_n(t_k))$ satisfies LICQ, SCS, and SOSC, then there exists $\gamma(t_k) > 0$ satisfying for each $\gamma > \gamma(t_k)$

1. There exists $\epsilon(t_k) > 0$ such that the BFGS iterates $(y^{(0)}(t_k, \gamma), y^{(1)}(t_k, \gamma), ... , y^{(L)}(t_k, \gamma), ...)$ converge locally and superlinearly to $y(t_k, \gamma)$, if $\| y^{(0)}(t_k, \gamma) - y(t_k, \gamma) \|_2 \leq \epsilon(t_k)$.

2. Assume that $(x(t_k), y^{(0)}(t_k, \gamma), y^{(1)}(t_k, \gamma), ...)$ is a KKT solution satisfying the LICQ and SOSC corresponding to $(y^{(0)}(t_k, \gamma))$, there exists $\epsilon(t_k) > 0$ such that the SQP iterates converge locally and superlinearly to $(x(t_k), y^{(0)}(t_k, \gamma), y^{(1)}(t_k, \gamma), ...)$, if the SQP starts sufficiently close to $(x(t_k), y^{(0)}(t_k, \gamma), y^{(1)}(t_k, \gamma), ...)$, and $\| y^{(0)}(t_k, \gamma) - y(t_k, \gamma) \|_2 \leq \epsilon(t_k)$.

4 Numerical Results

In this section, we present a numerical study to illustrate the proposed regularized inexact penalty decomposition algorithm. Two classes of problems are tested: a variant of the Golinski's problem, and SQPESgen, a problem set of quasi-separable quadratic programs with complementarity constraints (QPCPs). In addition, another simple numerical example is presented in the Appendix to provide step-by-step demonstration of the proposed approach.

4.1 A Variant of Golinski’s Problem

Golinski’s speed reducer design problem [47] is originally presented as a regular MDO problem. Its AIO formulation is given as follows:

$$\text{min } f(x_1, ..., x_7) = 0.7854 x_1^2 (3.3333 x_7^2 + 14.9333 x_3^2$$

subject to

$$g_1 = \frac{1}{110 x_3} \sqrt{745 x_5^2} + 1.69 \cdot 10^7 - 1 = 0$$

$$g_2 = \frac{1}{87 x_7} \sqrt{745 x_5^2} + 1.575 \cdot 10^8 - 1 = 0$$

$$g_3 = \frac{1.5 x_6 + 1.9}{x_4} - 1 \leq 0, \quad g_4 = \frac{1.1 x_3 + 1.9}{x_5} - 1 \leq 0, \quad g_5 = \frac{27}{x_1 x_2 x_3} - 1 \leq 0$$

$$g_6 = \frac{397.5}{x_1 x_2 x_5} - 1 \leq 0, \quad g_7 = \frac{1.93 x_6^3}{x_2 x_3 x_7^2} - 1 \leq 0, \quad g_8 = \frac{1.93 x_6^3}{x_2 x_3 x_7^2} - 1 \leq 0$$

(13)
While the Golinski’s problem was not originally presented with linking variables only, the original AIO problem can be decomposed into the RIPD formulation consisting the following master (Eq. (17)) and subproblems (Eqs. (18)–(20)):  
\[
\text{Gol_{master}}: \min F^*(x_1^{U}, x_2^{U}, x_3^{U}) = \sum_{i=1}^{n} F_i^*(x_1^{U}, x_2^{U}, x_3^{U})
\]

\[
F_i^*(x_1^{U}, x_2^{U}, x_3^{U}) = \min_{x_1^{(1)}, x_2^{(1)}, x_3^{(1)}, x_4^{(1)}, x_5^{(1)}, x_6^{(1)}, x_7^{(1)}, x_8^{(1)}, x_9^{(1)}, x_{10}^{(1)}, x_{11}^{(1)}, x_{12}^{(1)} = 0} \left(-1.5079 x_1^{(1)} + 7.477 x_3^{(1)} + 0.7854 x_4^{(1)} + \gamma_0 \sum_{i=1}^{3} (x_1^{(i)} - x_1^{(2)})^2\right)
\]

subject to \([g_1^{T}, g_3^{T}, g_{12}^{T}]^{T} \leq 0, \quad g_1 g_{12} \leq t_k\)

\[
F_i^*(x_1^{U}, x_2^{U}, x_3^{U}) = \min_{x_1^{(2)}, x_2^{(2)}, x_3^{(2)}, x_4^{(2)}, x_5^{(2)}, x_6^{(2)}, x_7^{(2)}, x_8^{(2)}, x_9^{(2)}, x_{10}^{(2)}, x_{11}^{(2)}, x_{12}^{(2)} = 0} \left(-1.5079 x_1^{(2)} + 7.477 x_3^{(2)} + 0.7854 x_4^{(2)} + \gamma_0 \sum_{i=1}^{3} (x_1^{(i)} - x_1^{(2)})^2\right)
\]

subject to \([g_2^{T}, g_4^{T}, g_{13}^{T}]^{T} \leq 0, \quad g_4 g_{13} \leq t_k\)

\[
F_i^*(x_1^{U}, x_2^{U}, x_3^{U}) = \min_{x_1^{(3)}, x_2^{(3)}, x_3^{(3)}, x_4^{(3)}, x_5^{(3)}, x_6^{(3)}, x_7^{(3)}, x_8^{(3)}, x_9^{(3)}, x_{10}^{(3)}, x_{11}^{(3)}, x_{12}^{(3)} = 0} \left(0.7854 x_1^{(3)} + 3.333 x_3^{(3)} + 14.93 x_3^{(3)} + 43.0934) + \gamma_0 \sum_{i=1}^{3} (x_1^{(i)} - x_1^{(3)})^2\right)
\]

subject to \([g_5^{T}, g_6^{T}, g_{10}^{T}, g_{11}^{T}]^{T} \leq 0, \quad g_6 g_{11} \leq t_k\)

We applied the proposed RIPD algorithm to the MDO-CC variant of Golinski’s problem. Both the nested loop and the single loop framework are tested. The numerical results are presented in Table 1 with a comparison between the RIPD solutions and the numerical AIO solution. It can be noted from Table 1 that both the nested loop RIPD approach and the single loop RIPD approach obtained solutions identical to the numerical AIO solution to MDO-CC variant of the Golinski’s problem. All of the three solutions are obtained with KNTRO [30] solver in the MATLAB environment. The corresponding parameter settings are \(t_0=0.05, \beta_1=8, \gamma_0=1, \beta_2=2, \epsilon_{\alpha} \approx 1 \times 10^{-6}, \epsilon_{\text{outer}} = 1 \times 10^{-6}\), and \(\epsilon_{\text{grad}}=5 \times 10^{-4}\). The active constraints at the solution are \(g_2, g_4, g_{10}, \text{ and } g_{12}\).

### 4.2 SQPECgen: A Test Problem Set for Quadratically Separable Programs With Complementarity Constraints.

SQPECgen [37] is a quasi-separable QPCC test problem set based on a QPCC generator proposed by Jiang and Ralph [48]. It features quasi-separable structure, which differentiates linking variables from local variables. Specifically, the problems it generates possess a quadratic objective function with polyhedral first level (relevant to linking variables only) constraints and complementarity second level (relevant to both linking and local variables) constraints, as shown below.

<table>
<thead>
<tr>
<th>Approach</th>
<th>Initial solution</th>
<th>Final solution</th>
</tr>
</thead>
<tbody>
<tr>
<td>AIO</td>
<td>[3.6,0.8,28,8.3,3.8,3.9,5.5]</td>
<td>[3.500000,0.700000,17.000000,7.300000,7.670397,3.542421,5.245814]</td>
</tr>
<tr>
<td>Nested loop RIPD</td>
<td>[3.6,0.8,28,8.3,3.8,3.9,5.5]</td>
<td>[3.500013,0.700004,17.000000,7.300000,7.670396,3.542421,5.245814]</td>
</tr>
<tr>
<td>Single loop RIPD</td>
<td>[3.6,0.8,28,8.3,3.8,3.9,5.5]</td>
<td>[3.500029,0.700005,17.000000,7.300000,7.670396,3.542421,5.245814]</td>
</tr>
</tbody>
</table>
system has five local variables. As shown in Fig. 8, the single loop RIPD converged to a solution identical to a numerical AIO solution from random generated initial solutions. The corresponding number of function evaluations (the average of two test problems with the same number of subsystems, five runs for each test problem) is plotted in Fig. 8. Additionally, the same metric corresponding to the ALD method, is also presented for a comparison purpose. Both algorithms converge to the same solution for each quasi-separable QPCC in the test problem set.

Table 2 presents the numerical results obtained through the RIPD algorithms. Each number in the table is the average number of function evaluations needed to converge from five random initial solutions. We note that for each initial solution the nested loop framework converged to a solution similar to a numerical AIO solution for all the problems tested. All of these solutions are generated with KNTTR0® 5.0 solver in MATLAB® 7.2. The corresponding parameter settings are τ0=0.05, β=4, γ0=1, βγ=2, εinner=1×10⁻⁶, εouter=5×10⁻⁵, and εgrad=5×10⁻³.

4.2.2 The Single Loop Framework. Although our numerical results show that the nested loop framework converges to AIO solutions, it usually incurs intensive computation due to excessive inner loop iterations, as shown in Table 2. In this subsection, we demonstrate the performance of the more practical single loop framework (Fig. 4) with a set of quasi-separable QPCC problems with varying number of subsystems. For each of the test problems, the single loop RIPD converged to a solution identical to a numerical AIO solution from five randomly generated initial solutions. The corresponding number of function evaluations (the average of two test problems with the same number of subsystems, five runs for each test problem) is plotted in Fig. 8. Additionally, the same metric corresponding to the ALD method, is also presented for a comparison purpose. Both algorithms converge to the same solution for each quasi-separable QPCC in the test problem set.

In Fig. 8, each data dot represents the average number of function evaluations for two test problems under the same subsystem settings. Here, each test problem has five linking variables, and each subsystem has five local variables. As shown in Fig. 8, the single loop RIPD approach takes approximately 14 times as many function evaluations to solve the 40-subsystem case as it takes for the five-subsystem case. The increase in computation cost results from both the increase in subsystem number and the increase in major iterations needed to generate a consistent solution. This ratio is approximately 75 for the case of ALD algorithm.

Table 2 presents the numerical results obtained through the RIPD algorithms. Each number in the table is the average number of function evaluations needed to converge from five random initial solutions. We note that for each initial solution the nested loop framework converged to a solution similar to a numerical AIO solution for all the problems tested. All of these solutions are generated with KNTTR0® 5.0 solver in MATLAB® 7.2. The corresponding parameter settings are τ0=0.05, β=4, γ0=1, βγ=2, εinner=1×10⁻⁶, εouter=5×10⁻⁵, and εgrad=5×10⁻³.

4.2.2 The Single Loop Framework. Although our numerical results show that the nested loop framework converges to AIO solutions, it usually incurs intensive computation due to excessive inner loop iterations, as shown in Table 2. In this subsection, we demonstrate the performance of the more practical single loop framework (Fig. 4) with a set of quasi-separable QPCC problems with varying number of subsystems. For each of the test problems, the single loop RIPD converged to a solution identical to a numerical AIO solution from five randomly generated initial solutions. The corresponding number of function evaluations (the average of two test problems with the same number of subsystems, five runs for each test problem) is plotted in Fig. 8. Additionally, the same metric corresponding to the ALD method, is also presented for a comparison purpose. Both algorithms converge to the same solution for each quasi-separable QPCC in the test problem set.

In Fig. 8, each data dot represents the average number of function evaluations for two test problems under the same subsystem settings. Here, each test problem has five linking variables, and each subsystem has five local variables. As shown in Fig. 8, the single loop RIPD approach takes approximately 14 times as many function evaluations to solve the 40-subsystem case as it takes for the five-subsystem case. The increase in computation cost results from both the increase in subsystem number and the increase in major iterations needed to generate a consistent solution. This ratio is approximately 75 for the case of ALD algorithm.

Both algorithms in this study are implemented with KNTTR0® solver in the MATLAB® environment. The corresponding parameter settings for the five- and ten-subsystem cases are τ0=0.05, β=4, γ0=1, βγ=2, εinner=1×10⁻⁶, εouter=5×10⁻⁵, and εgrad=1×10⁻³; for the 20–40-subsystem case τ0=0.05, β=3, γ0=1, βγ=1.5, εinner=1×10⁻⁶, εouter=5×10⁻⁵, and εgrad=1×10⁻³. Additionally, the deviation tolerance of the ALD is set as 1×10⁻⁶, the same value as εinner in RIPD.

In addition to SQPECgen problems with varying number of subsystem, we have also tested the single loop framework on a set of SQPECgen problems with fixed number of linking variables and varying number local variables. Specifically, each problem tested has two linking variables and ten subsystems with the number of local variables per subsystem ranging from 30 to 50. The
According to our numerical experience, both the nested loop and single loop framework appears to be more sensitive to convergence. SQPECgen test problems, 2.5 is recommended to speedup the single loop RIPD approach with a setting.

Table 3 shows the numerical results obtained through both the RIPD and the regularized AIO approach. The numbers presented are the average numbers of function evaluations and computation times of five runs, for each of which the two algorithms obtained a same solution from a randomly generated initial solution. According to numerical results, the computation time of the RIPD algorithm is approximately four times as much as that of the regularized AIO approach for the 30-local variable case; while the former is approximately a third of the latter for the 50-local variable case. This indicates that the computational effect of decomposition is more remarkable when the coupling among the subsystems are looser. Additionally, we notice in Table 3 that function evaluations of the regularized AIO problem take more time than those of the RIPD subproblems due to the difference in dimensionality. Note that the RIPD computation time is measured under serial implementation, i.e., the computation time is the summation of all the subsystem computation time. If the RIPD is implemented in parallel, the computation time will be much shorter.

Both algorithms in this study are implemented with KNITRO® solver in the MATLAB® environment. The corresponding parameter setting is \( t_0=0.05, \beta_3=3, \gamma_0=1, \beta_y=1.5, \epsilon_{inner}=1 \times 10^{-4}, \epsilon_{outer}=1 \times 10^{-5}, \) and \( \epsilon_{grad}=1 \times 10^{-3}. \) The same \( t_0, \beta_3, \) and \( \epsilon_{outer} \) are applied to the regularized AIO approach.

4.2.3 Recommendation for Parameter Selection. Finally, we provide some recommendations for the selection of \( \beta_3 \) and \( \beta_y. \)

According to our numerical experience, both the nested loop and the single loop framework converge with a \( \beta_y \in (1, 10]. \) For the SQPECgen test problems, \( 2 \leq \beta_3 \leq 4 \) is recommended to speedup convergence.

Additionally, our numerical experience indicates that the nested loop framework converges with a \( \beta_y \in (1, 4], \) while \( 2 \leq \beta_y \leq 3 \) is recommended to speedup convergence. On the other hand, the single loop framework appears to be more sensitive to \( \beta_y \) setting. A \( \beta_y \in (1, 2.5] \) worked for all the problems we tested; and \( 1 \leq \beta_y \leq 2 \) is recommended for convergence speed.

5 Conclusion and Future Work

The presence of complementarity constraints in a MDO problem poses a numerical challenge, which existing MDO approaches usually could not handle. This paper presents a nested decomposition formulation for the MDO-CC based on regularization and inexact penalty decomposition technique. As an important contribution, we showed that existing theories could be adapted to map a limit point of stationary solutions of the parameterized decomposition formulation to a strongly-stationary solution of the AIO formulation. Following this result, a solution algorithms for the MDO-CC were developed with potential implementations of a nested loop framework and a single loop framework. Additionally, we showed that superlinear convergence rate could be expected for the proposed algorithm following the local convergence results of standard master and subproblem solvers. Numerical results showed that both the two presented algorithms converge to solutions identical to the AIO solutions.

Future work includes an in-depth study regarding the differences between nested loop versus single loop solution process. Currently, the numerical results show that the single loop cases converge to AIO solutions, which is identical to the nested loop case. An extensive study will be useful in terms of comparing efficiency and robustness (from ill-conditioning) of solution process, which can provide further guidelines for setting \( \beta \) coefficients.

Acknowledgment

This work is supported by the National Science Foundation under Grant No. 0726934 and by Caterpillar, Inc. under Project No. 2007-03101. Any opinions, findings, and conclusions or recommendations expressed in this work are those of the authors and do not necessarily reflect the views of the National Science Foundation and Caterpillar, Inc. The authors acknowledge Uday Shanbhag for the suggestions in the initial stage.

Appendix: Illustrative Example

A simple numerical example is provided in this Appendix, for illustrative purpose. The example problem is a quasi-separable QPCC with two linking variables and two subsystems with two local variables each. The formulation of the problem is given as follows:

\[
\begin{align*}
\min_{x=(y', z', x', z')^T} & \quad \frac{1}{2} z^T P z + \begin{bmatrix} 1 & \frac{1}{2} \end{bmatrix} z
\end{align*}
\]

\[
\begin{bmatrix}
2.0009 & -0.3070 & -0.0916 & 0.6246 & -0.0916 & 0.6246 \\
-0.3070 & 0.8408 & -0.7418 & -0.7411 & -0.7418 & -0.7411 \\
-0.0916 & -0.7418 & 2.6815 & -0.1375 & 0 & 0 \\
0.6246 & -0.7411 & -0.1375 & 2.6477 & 0 & 0 \\
-0.0916 & -0.7418 & 0 & 0 & 2.6815 & -0.1375 \\
0.6246 & -0.7411 & 0 & 0 & -0.1375 & 2.6477 \\
\end{bmatrix}
\begin{bmatrix}
-1.4486 \\
2.9529 \\
1.4343 \\
-0.8599 \\
-2.2608 \\
3.5206 \\
\end{bmatrix}
\]
Following the technique presented in Sec. 2, the decomposition formulations of the above QPCC are:

\[
\min_{y} F^{\ast}(y) = \sum_{i=1}^{m} F_i^*(y) \quad (A2)
\]

\[
F_1^*(y) = \min_{z_1} \frac{1}{2} z_1^T \begin{pmatrix}
0 & 0 & -0.0916 & 0.6246 \\
0 & 0 & -0.7418 & -0.7411 \\
0.6246 & -0.7411 & -0.1375 & 2.6477
\end{pmatrix} z_1 + 0.4343 \\
+ \begin{pmatrix} 0 \
0 \
-0.8599 \end{pmatrix}^T y \quad (A3)
\]

subject to \( x_1 \triangleq 0 \begin{pmatrix}
-0.0136 & -0.0067 \\
-0.5144 & -0.6580
\end{pmatrix} y + \begin{pmatrix} 4.2209 & 2.2445 \
2.2445 & 1.5515 \end{pmatrix} x_1 + \begin{pmatrix} -5.9478 \
-3.3817 \end{pmatrix} \geq 0 \)

\[
F_2^*(y) = \min_{z_2} \frac{1}{2} z_2^T \begin{pmatrix}
0 & 0 & -0.0916 & 0.6246 \\
0 & 0 & -0.7418 & -0.7411 \\
0.6246 & -0.7411 & -0.1375 & 2.6477
\end{pmatrix} z_2 + \begin{pmatrix} 0 \
0 \
-2.2608 \
-3.5206 \end{pmatrix}
\]

subject to \( x_2 \triangleq 0 \begin{pmatrix}
-0.0136 & -0.0067 \\
-0.5144 & -0.6580
\end{pmatrix} y + \begin{pmatrix} 4.2209 & 2.2445 \
2.2445 & 1.5515 \end{pmatrix} x_2 + \begin{pmatrix} -5.9478 \
-3.3817 \end{pmatrix} \geq 0 \quad (A4)

\[
F_3^*(y) = \min_{y_3} \frac{1}{2} y_3^T \begin{pmatrix}
2.0009 & -0.3070 & 1.4486 \\
-0.3070 & 0.8408 & 2.9529
\end{pmatrix} y_3 + \begin{pmatrix} 0 \end{pmatrix}^T y_3 + \gamma_3 \left\| y - y_3 \right\|_2^2
\]

subject to \( y_3 \triangleq 0 \begin{pmatrix}
-0.1243 & -0.1101
\end{pmatrix} y \leq 0.2481 \quad (A5)\)

We take an initial solution of \( y^0 = (-2.2, 0, 0, 0, 0) \), with an initial \( t_1 = 0.05 \) and an initial \( y_3 \) of 1. Following step 1.1 of Fig. 5, we start with solving the three subsystems (Eqs. (A3)–(A5)), respectively, which yields:

\[
\begin{pmatrix}
\tilde{y}_1^{(0)}(t_1, y_3) \\
\tilde{y}_2^{(0)}(t_1, y_3) \\
\tilde{y}_3^{(0)}(t_1, y_3)
\end{pmatrix} = \begin{pmatrix}
-2.6223 \\
2.7791 \\
0.4257
\end{pmatrix} \cdot \begin{pmatrix}
\tilde{y}_1^{(0)}(t_1, y_3) \\
\tilde{y}_2^{(0)}(t_1, y_3) \\
\tilde{y}_3^{(0)}(t_1, y_3)
\end{pmatrix}
\]

Following Eqs. (A2) and (12), the master objective and its gradient are:

\[
F^*(y^{(0)}(t_1, y_3)) = -3.0511, \nabla F^*(y^{(0)}(t_1, y_3)) = \begin{pmatrix}
-1.3280 \\
0.5640
\end{pmatrix}
\]

With the initial Hessian approximation \( B_0 \) as the identity matrix, this generates a descent search direction (step 1.2 of Fig. 5)

\[
\Delta y^{(0)}(t_1, y_3) = \begin{pmatrix}
1.3280 \\
0.5640
\end{pmatrix}
\]

Following step 1.3 of Fig. 5, let \( y = y^{(0)}(t_1, y_3) + \alpha_0 \Delta y^{(0)}(t_1, y_3) = (-0.6720, 2.5640)^T \) be a tentative solution (where \( \alpha_0 = 1 \)), the corresponding subsystem solutions and master objective are:

\[
\begin{pmatrix}
\tilde{y}_1(y^{(0)}(t_1, y_3)) \\
\tilde{y}_2(y^{(0)}(t_1, y_3)) \\
\tilde{y}_3(y^{(0)}(t_1, y_3))
\end{pmatrix} = \begin{pmatrix}
2.2445 \\
1.5515 \\
2.2445 \\
1.5515 \\
2.2445 \\
1.5515
\end{pmatrix}
\]

\[
\begin{pmatrix}
\tilde{y}_1^{(0)}(t_1, y_3) \\
\tilde{y}_2^{(0)}(t_1, y_3) \\
\tilde{y}_3^{(0)}(t_1, y_3)
\end{pmatrix} = \begin{pmatrix}
-2.0427 \\
2.5272 \\
3.0225
\end{pmatrix} \cdot \begin{pmatrix}
\tilde{y}_1^{(0)}(t_1, y_3) \\
\tilde{y}_2^{(0)}(t_1, y_3) \\
\tilde{y}_3^{(0)}(t_1, y_3)
\end{pmatrix}
\]

\[
\begin{pmatrix}
\tilde{y}_1(y, t_1, y_3) \\
\tilde{y}_2(y, t_1, y_3) \\
\tilde{y}_3(y, t_1, y_3)
\end{pmatrix} = \begin{pmatrix}
2.0777 \\
2.1077 \\
2.2036
\end{pmatrix}
\]

\[
\begin{pmatrix}
\tilde{y}_1^{(0)}(t_1, y_3) \\
\tilde{y}_2^{(0)}(t_1, y_3) \\
\tilde{y}_3^{(0)}(t_1, y_3)
\end{pmatrix} = \begin{pmatrix}
0.0856 \\
0.0224 \\
0.7749
\end{pmatrix}
\]

\[
F^*(y) = -2.1618
\]

Assume that \( \sigma = 0.05 \), since \( F^*(y) - F^*(y^{(0)}(t_1, y_3)) > \sigma \), \( \nabla F^*(y^{(0)}(t_1, y_3))^T \Delta y^{(0)}(t_1, y_3) \) is not satisfied, the tentative solution \( y \) is rejected, and the step length \( \alpha_0 \) is diminished by a factor of 0.5. With this updated step length, a new tentative solution is calculated, given as \( y^* = (-1.3360, 2.2820)^T \). The corresponding subsystem solutions and master objective are calculated as follows:


\[
\begin{align*}
\left( \tilde{y}_1(t_1, \gamma_1) \right) & = \left( \begin{array}{c}
-2.3260 \\
2.6545 \\
0.3854 \\
1.9768 \\
0.3391 \\
0.0293 \\
1.9566
\end{array} \right), \\
\left( \tilde{y}_2(t_1, \gamma_1) \right) & = \left( \begin{array}{c}
0.4316 \\
1.8839 \\
0.3854 \\
0.3529 \\
0.3854 \\
0.3529
\end{array} \right)
\end{align*}
\]

Because

\[
F^*(\tilde{y}) = -2.1618
\]

Since

\[
F^*(\tilde{y}) - F^*(y^{(1)}(t_1, \gamma_1)) > \sigma F^*(y^{(1)}(t_1, \gamma_1))^T \Delta y^{(1)}(t_1, \gamma_1),
\]

the tentative solution \( \tilde{y} \) is accepted

\[
y^{(1)}(t_1, \gamma_1) = \tilde{y}
\]

\[
\left( \begin{array}{c}
y_1^{(1)}(t_1, \gamma_1) \\
x_1^{(1)}(t_1, \gamma_1)
\end{array} \right) = \left( \begin{array}{c}
\tilde{y}_1(t_1, \gamma_1) \\
\tilde{x}_1(t_1, \gamma_1)
\end{array} \right), \\
\left( \begin{array}{c}
y_2^{(1)}(t_1, \gamma_1) \\
x_2^{(1)}(t_1, \gamma_1)
\end{array} \right) = \left( \begin{array}{c}
\tilde{y}_2(t_1, \gamma_1) \\
\tilde{x}_2(t_1, \gamma_1)
\end{array} \right)
\]

Additionally, the Hessian approximation is updated.

\[
F^*(\tilde{y}) = -2.1618
\]

The algorithm will keep iterating until the BFGS termination criterion is met. If, for example, \( \epsilon_{grad} = 1 \times 10^{-3} \), it terminates at \((-1.2737, 1.7736)^T\) after the sixth iterations. At this point, the penalty parameter \( \gamma \) will be updated if the nested framework is followed, while if the single loop framework is followed, not only the penalty parameter \( \gamma \) but also the regularization parameter \( t \) will be updated. After this update, the algorithm will start a new set of BFGS iterations in a similar fashion as presented above.

References


