Revisiting Individual Discipline Feasible using matrix-free Inexact-Newton-Krylov

Alp Dener[*] and Jason E. Hicken[†]

Rensselaer Polytechnic Institute, Troy, New York, 12180

The individual-discipline-feasible (IDF) formulation was proposed to simplify the implementation of MDO problems. The IDF formulation introduces coupling variables into the optimization problem that eliminate the need for a full multidisciplinary analysis at each optimization iteration; this simplifies the solution of MDO problems by maintaining modularity of the discipline software. Historically, the MDO community has used conventional optimization algorithms to solve IDF-formulated problems. Conventional optimizers are not well suited to IDF, because they use limited-memory quasi-Newton methods (linear convergence) and require the constraint Jacobian explicitly. The cost of computing the coupling-variable constraint Jacobian is prohibitively expensive for high-fidelity IDF problems. Matrix-free Reduced-Space inexact-Newton-Krylov (RSNK) algorithms overcome these issues, because they scale superlinearly and do not require the constraint Jacobian explicitly. Therefore, this class of algorithm has great potential to solve IDF-formulated MDO problems in a scalable and efficient manner. In this paper, we describe the application of RSNK to the IDF formulation and compare its performance to the multidisciplinary feasible architecture.

I. Introduction

Multidisciplinary design optimization (MDO) is a powerful methodology that can be used to inform the design of complex engineering systems. In brief, MDO uses numerical optimization to improve the design of a product and/or process that is mathematically modeled. Provided the mathematical model is sufficiently accurate, MDO can provide valuable insight into the trade-offs between different potential solutions.

In this paper, we are specifically interested in MDO problems that involve two or more coupled partial-differential equations (PDEs). These high-fidelity MDO problems arise when simulations offer the only way to accurately capture the complex physics involved. Examples of high-fidelity MDO problems include the aerostructural optimization of a helicopter rotor, the noise minimization of aircraft flaps, and the minimization of greenhouse gas emissions from an engine.

High-fidelity MDO problems are computationally intense; for example, state-of-the-art aerostructural optimizations require hundreds of simulations on hundreds of processors [1]. This motivates highly efficient optimization algorithms. Efficiency, however, is not the only requirement. Modular algorithms are also important, because the simulations often involve disparate PDE solvers for each of the disciplines.

The individual discipline feasible (IDF) formulation was proposed as means to maintain modularity when solving MDO problems [2, 3]. The IDF formulation, described in more detail later, introduces coupling variables that allow the optimization to proceed without requiring a full multidisciplinary analysis at each iteration. Thus, the individual PDE solvers can be kept independent during the optimization. This independence is a significant advantage of IDF from the perspective of implementation. Moreover, since the disciplines remain feasible with respect to their PDEs, the IDF formulation is more robust than the full-space methods often used for single-discipline PDE-constrained optimization problems (e.g. LNKS [4, 5]).

Despite its advantages, IDF has not become the de facto formulation for solving MDO problems. The broad adoption of IDF has been hindered for two reasons. First, the coupling variables introduced by IDF increase the size of the optimization problem by several orders of magnitude. Gradient-based algorithms

[*]Graduate Student, Department of Mechanical, Aerospace, and Nuclear Engineering, Student Member AIAA
[†]Assistant Professor, Department of Mechanical, Aerospace, and Nuclear Engineering, Member AIAA

frequently resort to limited-memory BFGS for such large-scale problems [6–8], and this leads to linear convergence rates [9, 10] and many iterations for IDF-based formulations. The second problem posed by IDF is the Jacobian of the coupling-variable constraints, which is explicitly required by most conventional optimization algorithms. The coupling-constraint Jacobian is prohibitively expensive to form, because it amounts to computing a total derivative for each of the coupling variables.

The two problems identified above are limitations of the optimization algorithms used to solve IDF problems and not of the IDF formulation itself. This observation suggests that alternative, state-of-the-art optimization algorithms may avoid these limitations. The hypothesis pursued here is that matrix-free Reduced-Space inexact-Newton-Krylov (RSNK) is such an algorithm.

The paper is organized as follows. We begin by reviewing the IDF formulation and MDO problems in general. In Section III, we present the reduced-space inexact-Newton-Krylov algorithm, with a particular focus on computing the KKT-vector products needed by the method. Subsequently, we introduce a novel preconditioner for the IDF problem in Section IV. Results comparing MDF with IDF can be found in Section V. Finally, conclusions and future work are discussed in Section VI.

II. Individual Discipline Feasible

To simplify the description of the proposed approach, we will consider a two-discipline problem. We begin by defining the multi-disciplinary analysis (MDA) problem: for a fixed set of design variables $\mathbf{x} \in \mathbb{R}^n$, solve

$$\boldsymbol{\mathcal{C}}^{(\mathbf{u})}(\mathbf{u}, R^{(\mathbf{v})}\mathbf{v}, \mathbf{x}) = \mathbf{0}, \tag{1}$$

$$\boldsymbol{\mathcal{C}}^{(\mathbf{v})}(\mathbf{v}, R^{(\mathbf{u})}\mathbf{u}, \mathbf{x}) = \mathbf{0}, \tag{2}$$

for $\mathbf{u} \in \mathbb{R}^p$ and $\mathbf{v} \in \mathbb{R}^q$. The variables $\mathbf{u}$ and $\mathbf{v}$ are called the state variables, and (1) and (2) are the state equations. In general, $\boldsymbol{\mathcal{C}}^{(\mathbf{u})} \rightarrow \mathbb{R}^p \times \mathbb{R}^s \times \mathbb{R}^n$ and $\boldsymbol{\mathcal{C}}^{(\mathbf{v})} \rightarrow \mathbb{R}^q \times \mathbb{R}^r \times \mathbb{R}^n$ are nonlinear algebraic operators that model the physics of the two disciplines. In the present work, (1) and (2) represent discretizations of partial-differential equations (PDEs). We assume that the PDEs are well posed at $\mathbf{x}$, so that a solution to (1) and (2) exists.

The state equations are coupled via $R^{(\mathbf{u})}\mathbf{u}$ and $R^{(\mathbf{v})}\mathbf{v}$, where $R^{(\mathbf{u})} \in \mathbb{R}^{r\times p}$ and $R^{(\mathbf{v})} \in \mathbb{R}^{s\times q}$ are binary matrices that select a subset of $\mathbf{u}$ and $\mathbf{v}$, respectively. Typically, these matrices are rectangular, and often $r \ll p$ and $s \ll q$.

An aeroelastic-wing problem will help illustrate the notation above. In this case, $\boldsymbol{\mathcal{C}}^{(\mathbf{u})} = \mathbf{0}$ may represent a computational-fluid-dynamics (CFD) model of the aerodynamics and $\boldsymbol{\mathcal{C}}^{(\mathbf{v})} = \mathbf{0}$ a computational-structural-mechanics (CSM) model of the structure. Thus, the state variable $\mathbf{u}$ is a vector of flow variables (e.g. ρ, $\rho\vec{u}$, e) for each degree of freedom in the CFD mesh. Likewise, $\mathbf{v}$ is a vector of the displacements for each nodal degree of freedom in the CSM mesh. Furthermore, $R^{(\mathbf{u})}\mathbf{u}$ selects those CFD degrees of freedom that determine the aerodynamic forces on the wing, and $R^{(\mathbf{v})}\mathbf{v}$ is the subset of CSM variables that determine the displacement of the surface of the wing.

Next, we introduce the multi-disciplinary design optimization (MDO) problem based on the MDA (1)–(2). Let $\mathcal{J} : \mathbb{R}^n \times \mathbb{R}^p \times \mathbb{R}^q$ denote the objective function of interest. The generic two-discipline MDO problem is

$$\begin{aligned} \underset{\mathbf{x},\mathbf{u},\mathbf{v}}{\text{minimize}} \quad & \mathcal{J}(\mathbf{x}, \mathbf{u}, \mathbf{v}), \\ \text{subject to} \quad & \boldsymbol{\mathcal{C}}^{(\mathbf{u})}(\mathbf{u}, R^{(\mathbf{v})}\mathbf{v}, \mathbf{x}) = \mathbf{0}, \\ & \boldsymbol{\mathcal{C}}^{(\mathbf{v})}(\mathbf{v}, R^{(\mathbf{u})}\mathbf{u}, \mathbf{x}) = \mathbf{0}. \end{aligned} \tag{3}$$

In the aeroelastic-wing example, $\mathcal{J}$ may be the Breguet range equation, which accounts for both structural weight and aerodynamic performance. The design variables may be aerodynamic shape parameters (twist, sweep, etc.) and structural-member sizing (skin gauge, rib thickness, etc.).

Before arriving at our final formulation, we introduce the coupling variables $\bar{\mathbf{u}} \in \mathbb{R}^r$ and $\bar{\mathbf{v}} \in \mathbb{R}^s$ into the

MDO problem (3):

$$\begin{aligned} \underset{\mathbf{x},\mathbf{u},\mathbf{v},\bar{\mathbf{u}},\bar{\mathbf{v}}}{\text{minimize}} \quad & \mathcal{J}(\mathbf{x},\mathbf{u},\mathbf{v}), \\ \text{subject to} \quad & \boldsymbol{\mathcal{C}}^{(\mathbf{u})}(\mathbf{u},\bar{\mathbf{v}},\mathbf{x}) = \mathbf{0}, \\ & \boldsymbol{\mathcal{C}}^{(\mathbf{v})}(\mathbf{v},\bar{\mathbf{u}},\mathbf{x}) = \mathbf{0}, \\ & \bar{\mathbf{u}} - R^{(\mathbf{u})}\mathbf{u} = \mathbf{0}, \\ & \bar{\mathbf{v}} - R^{(\mathbf{v})}\mathbf{v} = \mathbf{0}. \end{aligned} \tag{4}$$

This is called the "most general formulation" of MDO in [3] and the "all-at-once" (AAO) problem statement in [11]. Although this formulation is not used in practice — $\bar{\mathbf{u}}$ and $\bar{\mathbf{v}}$ can easily be eliminated — it provides a bridge between different MDO formulations.

In practice, the state variables are often considered implicit functions of the design variables. That is, for a given $\mathbf{x}$, we solve for $\mathbf{u}$ and $\mathbf{v}$ using the state equations. This is called a reduced-space formulation, and it is motivated by several concerns. First, there is usually efficient legacy software available to solve the state equations. Moreover, these existing software libraries include specialized globalization strategies for the particular nonlinearities that appear in the state equations. An example in the aeroelastic example would be a CFD solver that has been tailored to handle highly nonlinear turbulence models. A general optimization algorithm that seeks to solve the full-space problem, e.g. problem (3), will not have access to these specialized solution strategies and may fail to converge.

Another shortcoming of the full-space approach is the lack of optimization software that can accommodate distributed memory. When the state constraints represent PDE models, they may need 10^6–10^9 degrees of freedom to accurately capture the physics. Such large problems demand parallel solution methods, yet few parallel optimization libraries exist. Therefore, to use a full-space approach practitioners must build an optimization algorithm on top of their PDE solver(s). This represents a significant software-engineering task and does not promote software reuse.

Having motivated reduced-space formulations, we now consider how they might be incorporated in the two-discipline MDO formulation. We could consider a reduced-space version of (3), that is

$$\underset{\mathbf{x}}{\text{minimize}} \quad \mathcal{J}(\mathbf{x},\mathbf{u}(\mathbf{x}),\mathbf{v}(\mathbf{x})). \tag{5}$$

Here, we invoke the implicit function theorem to express $\mathbf{u}$ and $\mathbf{v}$ as functions of $\mathbf{x}$ using the MDA (1)–(2). In the MDO literature, (5) is called the multidisciplinary feasible (MDF) formulation. The MDF formulation requires a full MDA for each evaluation of $\mathcal{J}$, e.g. a coupled CFD-CSM simulation in the aeroelastic example. Furthermore, if we use a gradient-based optimization algorithm, MDF will require either expensive finite-difference approximations or a coupled-adjoint approach [12].

Despite the challenges, there has been considerable success using MDF in aeroelastic optimization problems [1,12–14]; nevertheless, the fact remains that the approach demands a monolithic solution method (the MDA) and, perhaps, a monolithic coupled adjoint. In many applications it is desirable to keep the solution of the individual state equations separate. For example, we might have a legacy CFD library and a commercial CSM library, but no direct means of coupling these to solve aeroelastic problems. Even if two particular libraries have been coupled, it may be difficult to replace one or both of the libraries should the need arise.

The above observations motivate the individual-discipline-feasible (IDF) formulation [2,3]. The IDF formulation is a reduced-space version of (4) in which the state variables have been eliminated, but the coupling variables remain.

$$\begin{aligned} \underset{\mathbf{x},\bar{\mathbf{u}},\bar{\mathbf{v}}}{\text{minimize}} \quad & \mathcal{J}\left(\mathbf{x},\mathbf{u}(\mathbf{x},\bar{\mathbf{v}}),\mathbf{v}(\mathbf{x},\bar{\mathbf{u}})\right), \\ \text{subject to} \quad & \bar{\mathbf{u}} - R^{(\mathbf{u})}\mathbf{u}(\mathbf{x},\bar{\mathbf{v}}) = \mathbf{0}, \\ & \bar{\mathbf{v}} - R^{(\mathbf{v})}\mathbf{v}(\mathbf{x},\bar{\mathbf{u}}) = \mathbf{0}. \end{aligned} \tag{6}$$

Notice that the state variables are now implicit functions of the design variables *and* the coupling variables. Consequently, the linear equations that previously defined the coupling variables in (4) have become nonlinear constraints in (6).

Because the IDF formulation uses coupling variables, the state equations can be solved independently at each optimization iteration. This is a significant advantage of the formulation. Unfortunately, the IDF optimization problem can be considerably larger than the corresponding MDF problem; indeed, the coupling variables may add between 10^3 to 10^4 degrees of freedom in our aeroelastic example. Another issue with

IDF, discussed below, is that the cost of computing the coupling-constraint Jacobian grows with the number of coupling variables. Thus, to make the IDF formulation practical, we need a large-scale optimization algorithm that scales independently of the number of coupling variables.

III. Reduced-Space Inexact-Newton-Krylov Approach

A local optimum for IDF must satisfy the first-order optimality conditions (i.e. the KKT conditions). These necessary optimality conditions can be derived by differentiating the Lagrangian of (6):

$$\mathcal{L} \equiv \mathcal{J} + \boldsymbol{\lambda}^T(\bar{\mathbf{u}} - R^{(\mathbf{u})}\mathbf{u}) + \boldsymbol{\mu}^T(\bar{\mathbf{v}} - R^{(\mathbf{v})}\mathbf{v}) + \boldsymbol{\Psi}_1^T \boldsymbol{\mathcal{C}}^{(\mathbf{u})} + \boldsymbol{\Phi}_1^T \boldsymbol{\mathcal{C}}^{(\mathbf{v})}, \tag{7}$$

where $\boldsymbol{\lambda} \in \mathbb{R}^r$ and $\boldsymbol{\mu} \in \mathbb{R}^s$ are the Lagrange multipliers associated with the coupling-variable equations. The vectors $\boldsymbol{\Psi}_1 \in \mathbb{R}^p$ and $\boldsymbol{\Phi}_1 \in \mathbb{R}^q$ are Lagrange multipliers associated with the state equations; they are the adjoint variables and the meaning of the subscript "1" will become clear below. Introducing these adjoint variables is not strictly necessary, since the state equations are satisfied at each iteration of IDF; however, their introduction greatly simplifies the sensitivity analysis.

Thus, differentiating $\mathcal{L}$, we find that the first-order conditions for an optimum of IDF are given by

$$\begin{aligned}
\frac{\partial \mathcal{L}}{\partial \mathbf{x}} &= \frac{\partial \mathcal{J}}{\partial \mathbf{x}} + \boldsymbol{\Psi}_1^T \frac{\partial \boldsymbol{\mathcal{C}}^{(\mathbf{u})}}{\partial \mathbf{x}} + \boldsymbol{\Phi}_1^T \frac{\partial \boldsymbol{\mathcal{C}}^{(\mathbf{v})}}{\partial \mathbf{x}} = \mathbf{0}^T, \\
\frac{\partial \mathcal{L}}{\partial \bar{\mathbf{u}}} &= \boldsymbol{\lambda}^T + \boldsymbol{\Phi}_1^T \frac{\partial \boldsymbol{\mathcal{C}}^{(\mathbf{v})}}{\partial \bar{\mathbf{u}}} = \mathbf{0}^T, \\
\frac{\partial \mathcal{L}}{\partial \bar{\mathbf{v}}} &= \boldsymbol{\mu}^T + \boldsymbol{\Psi}_1^T \frac{\partial \boldsymbol{\mathcal{C}}^{(\mathbf{u})}}{\partial \bar{\mathbf{v}}} = \mathbf{0}^T, \\
\frac{\partial \mathcal{L}}{\partial \boldsymbol{\lambda}} &= \bar{\mathbf{u}} - R^{(\mathbf{u})}\mathbf{u} = \mathbf{0}, \\
\frac{\partial \mathcal{L}}{\partial \boldsymbol{\mu}} &= \bar{\mathbf{v}} - R^{(\mathbf{v})}\mathbf{v} = \mathbf{0}.
\end{aligned} \tag{8}$$

The adjoint variables $\boldsymbol{\Psi}_1$ and $\boldsymbol{\Phi}_1$ are found by differentiating $\mathcal{L}$ with respect to $\mathbf{u}$ and $\mathbf{v}$, respectively, and solving the resulting linear equations. Below, we define the residuals of these adjoint linear systems.

$$\boldsymbol{\mathcal{S}}^{(\mathbf{u})} \equiv \left(\frac{\partial \mathcal{J}}{\partial \mathbf{u}}\right)^T - \left(R^{(\mathbf{u})}\right)^T \boldsymbol{\lambda} + \left(\frac{\partial \boldsymbol{\mathcal{C}}^{(\mathbf{u})}}{\partial \mathbf{u}}\right)^T \boldsymbol{\Psi}_1 = \mathbf{0}, \tag{9}$$

$$\boldsymbol{\mathcal{S}}^{(\mathbf{v})} \equiv \left(\frac{\partial \mathcal{J}}{\partial \mathbf{v}}\right)^T - \left(R^{(\mathbf{v})}\right)^T \boldsymbol{\mu} + \left(\frac{\partial \boldsymbol{\mathcal{C}}^{(\mathbf{v})}}{\partial \mathbf{v}}\right)^T \boldsymbol{\Phi}_1 = \mathbf{0}. \tag{10}$$

Like the state equations, the adjoint equations are solved at each optimization iteration[a].

Because $\boldsymbol{\Psi}_1$ and $\boldsymbol{\Phi}_1$ are used to compute first-derivative information, they are sometimes referred to as first-order adjoints; hence, the subscript 1. Similarly, (9)–(10) are called the first-order adjoint equations. The solution of first-order adjoint equations is well understood and developed in the literature. Indeed, even commercial analysis codes are beginning to provide first-order adjoint analyses; e.g., ANSYS 14.5 (`http://www.ansys.com/Products/ANSYS+14.5+Release+Highlights/`). Therefore, we will not discuss the practical implementation of equations (9)–(10) further.

The KKT conditions are a set of nonlinear coupled equations; thus, when the problem is sufficiently smooth, the most efficient methods for solving (8) are based on Newton's method. Let $\mathbf{y}^T = (\mathbf{x}^T, \bar{\mathbf{u}}^T, \bar{\mathbf{v}}^T)$ denote the vector of all free variables, and let $\boldsymbol{\sigma}^T = (\boldsymbol{\lambda}^T, \boldsymbol{\mu}^T)$ denote the vector of all Lagrange multipliers. Then Newton's method applied to (8) yields the KKT system

$$\begin{bmatrix} H & A^T \\ A & 0 \end{bmatrix} \begin{pmatrix} \Delta \mathbf{y} \\ \Delta \boldsymbol{\sigma} \end{pmatrix} = -\begin{pmatrix} \frac{\partial \mathcal{L}}{\partial \mathbf{y}} \\ \frac{\partial \mathcal{L}}{\partial \boldsymbol{\sigma}} \end{pmatrix}. \tag{11}$$

where

$$H \equiv \frac{\partial^2 \mathcal{L}}{\partial \mathbf{y}^2}, \qquad \text{and} \qquad A \equiv \frac{\partial^2 \mathcal{L}}{\partial \mathbf{y} \partial \boldsymbol{\sigma}},$$

[a]More precisely, they are solved whenever the iteration requires a gradient.

are the Hessian of the Lagrangian and the coupling-constraint Jacobian, respectively.

Most Newton-based optimization algorithms require routines that compute H and A explicitly. This is unacceptable for problems where these matrices are expense or impractical to compute. The IDF formulation (6) is one such problem. The coupling constraint Jacobian involves sensitivities of $\mathbf{u}$ and $\mathbf{v}$ with respect to the design variables and the coupling variables. To compute each of these sensitivities would require solving $r+s$ first-order adjoint problems. Similarly, the cost of forming the IDF Hessian scales with the number of optimization variables.

Motivated by applications where H and A are expensive to compute, the optimization community has been actively developing matrix-free algorithms; see, for example, [15–18]. These algorithms typically use an iterative Krylov solver to inexactly solve for the Newton step in (11); hence the name inexact-Newton-Krylov (INK). The terminology matrix-free comes from the fact that Krylov solvers for (11) require only Jacobian-vector, Jacobian-transposed-vector, and Hessian-vector products. That is, these methods do not require H and A explicitly.

The application of matrix-free optimization methods to the IDF formulation is an important and unexplored area of MDO research. However, there are two important aspects of matrix-free algorithms that must be addressed before they will be practical for high-fidelity MDO problems posed in the IDF formulation: the KKT-matrix-vector products and preconditioning. We discuss the products below and describe our preconditioning strategy in Section IV.

III.A. KKT-matrix-vector Products and Second-order Adjoints

As explained earlier, RSNK optimization algorithms require several types of matrix-vector products. The accuracy and computational cost of these products has a significant impact on the overall efficiency of the INK algorithm.

One can show that the necessary Jacobian-vector products can be formed by solving two forward linear systems of size p and q. Similarly, Jacobian-transposed-vector products, can be formed by solving two reverse (adjoint) systems of size p and q. For both of these products, the partial derivatives required are already computed as part of the KKT conditions (8). For example, if $\mathbf{w}_{\boldsymbol{\sigma}} \in \mathbb{R}^{r+s}$ is an arbitrary vector, then

$$A^T \mathbf{w}_{\boldsymbol{\sigma}} = \left[\frac{\partial \bar{\mathbf{u}}}{\partial \mathbf{y}} + R^{(\mathbf{u})} \left(\frac{\partial \boldsymbol{\mathcal{C}}^{(\mathbf{u})}}{\partial \mathbf{u}} \right)^{-1} \frac{\partial \boldsymbol{\mathcal{C}}^{(\mathbf{u})}}{\partial \mathbf{y}} \right]^T \mathbf{w}_{\boldsymbol{\lambda}} + \left[\frac{\partial \bar{\mathbf{v}}}{\partial \mathbf{y}} + R^{(\mathbf{v})} \left(\frac{\partial \boldsymbol{\mathcal{C}}^{(\mathbf{v})}}{\partial \mathbf{v}} \right)^{-1} \frac{\partial \boldsymbol{\mathcal{C}}^{(\mathbf{v})}}{\partial \mathbf{y}} \right]^T \mathbf{w}_{\boldsymbol{\mu}}$$

where $\mathbf{w}_{\boldsymbol{\lambda}} \in \mathbb{R}^r$ and $\mathbf{w}_{\boldsymbol{\mu}} \in \mathbb{R}^s$ are the components of $\mathbf{w}_{\boldsymbol{\sigma}}$ corresponding to $\boldsymbol{\lambda}$ and $\boldsymbol{\mu}$, respectively. The inverse matrices in the above product can be contracted with $(R^{(\mathbf{u})})^T \mathbf{w}_{\boldsymbol{\lambda}}$ and $(R^{(\mathbf{v})})^T \mathbf{w}_{\boldsymbol{\mu}}$, and these contractions define the adjoint variables that must be solved to find the product; notice that no matrix-matrix products are needed, which is key to the scalability of the matrix-free approach.

Thus, the Jacobian-vector and Jacobian-transposed-vector products do not pose a challenge for an INK solution of the IDF formulation. The Hessian-vector products are a different matter, because these products contain second-order derivatives. Consequently, the Hessian is typically approximated using a quasi-Newton method like the Broyden-Fletcher-Goldberg-Shanno (BFGS) update [9, 10].

We believe that using quasi-Newton approximations for the Hessian-vector products will not be efficient for the IDF formulation. The algorithmic scaling of quasi-Newton methods depends on the number of optimization variables and linear scaling is not unusual. Consequently, the potentially large size of the coupling variables will render matrix-free INK methods based on quasi-Newton methods impractical. This was recognized by the early pioneers of IDF, who suggested the need to find a lower-dimensional approximation to the coupling variables in order to keep the optimization problem small [3]; however, reducing the dimension of the coupling variables through approximation has its own drawbacks. For example, what is a suitable model for the approximation, and how does the approximation impact the optimization results?

Rather than using a quasi-Newton approximation, we can compute the Hessian-vector products directly. Our motivation is a true inexact-Newton-Krylov algorithm with scaling that is independent of the size of the coupling variables and design space. While including second-order derivatives adds some complexity, the value of higher-order derivatives has been known for some time, even in the context of IDF [19].

Rather than computing the Hessian-vector, Jacobian-vector, and Jacobian-transposed-vector products separately, we compute the matrix-vector product for the entire KKT system matrix, i.e. the matrix in (11), which is also known as the primal-dual matrix. We will show that these products can be computed with the same number of forward and reverse adjoint systems — 4 in the case of a two discipline problem — as needed by the Jacobian-vector and Jacobian-transposed-vector products alone. Moreover, any second-order partial derivatives that are needed can be found using inexpensive finite-difference approximations of directional derivatives.

To derive the desired matrix-vector product, we construct the functional

$$\mathcal{K} \equiv \left(\frac{\partial \mathcal{L}}{\partial \mathbf{y}}\right) \mathbf{w}_{\mathbf{y}} + \left(\frac{\partial \mathcal{L}}{\partial \boldsymbol{\sigma}}\right) \mathbf{w}_{\boldsymbol{\sigma}},$$

where, $\mathbf{w} \in \mathbb{R}^{n+2(r+s)}$ is an arbitrary vector and $\mathbf{w}_{\mathbf{y}} \in \mathbb{R}^{n+r+s}$ and $\mathbf{w}_{\boldsymbol{\sigma}} \in \mathbb{R}^{r+s}$ are its components corresponding to $\mathbf{y}$ and $\boldsymbol{\sigma}$, respectively. The functional $\mathcal{K}$ is simply the product of the KKT conditions (8) with the vector $\mathbf{w}$.

Next, we recognize that the matrix-vector product is the total derivative of $\mathcal{K}$ with respect to $\mathbf{y}$ and $\boldsymbol{\sigma}$. To find this total derivative, we construct a Lagrangian that includes the state equations *and* the adjoint equations, since $\mathcal{K}$ depends on $\mathbf{u}, \mathbf{v}$ and $\boldsymbol{\Psi}_1$ and $\boldsymbol{\Phi}_1$.

$$\hat{\mathcal{L}} \equiv \mathcal{K} + \boldsymbol{\Psi}_2^T \boldsymbol{\mathcal{C}}^{(\mathbf{u})} + \boldsymbol{\Phi}_2^T \boldsymbol{\mathcal{C}}^{(\mathbf{v})} + \mathbf{u}_2^T \boldsymbol{\mathcal{S}}^{(\mathbf{u})} + \mathbf{v}_2^T \boldsymbol{\mathcal{S}}^{(\mathbf{v})},$$

where $\boldsymbol{\Psi}_2, \mathbf{u}_2 \in \mathbb{R}^p$ and $\boldsymbol{\Phi}_2, \mathbf{v}_2 \in \mathbb{R}^q$ are second-order adjoint variables [20]. The desired KKT-matrix-vector product is found by differentiating $\hat{\mathcal{L}}$ with respect to $\mathbf{y}$ and $\boldsymbol{\sigma}$:

$$\begin{pmatrix} \frac{\partial \hat{\mathcal{L}}}{\partial \mathbf{y}} \\ \frac{\partial \hat{\mathcal{L}}}{\partial \boldsymbol{\sigma}} \end{pmatrix} \equiv \begin{bmatrix} H & A^T \\ A & 0 \end{bmatrix} \begin{pmatrix} \mathbf{w}_{\mathbf{y}} \\ \mathbf{w}_{\boldsymbol{\sigma}} \end{pmatrix}. \tag{12}$$

The above product depends on the second-order adjoint variables, which are found by differentiating $\hat{\mathcal{L}}$ with respect to $\mathbf{u}$, $\mathbf{v}$, $\boldsymbol{\Psi}_1$, and $\boldsymbol{\Phi}_1$, and solving the resulting linear systems. These linear systems are given below for completeness.

$$\frac{\partial \boldsymbol{\mathcal{C}}^{(\mathbf{u})}}{\partial \mathbf{u}} \mathbf{u}_2 = -\left(\frac{\partial \mathcal{K}}{\partial \boldsymbol{\Psi}_1}\right)^T \tag{13}$$

$$\left(\frac{\partial \boldsymbol{\mathcal{C}}^{(\mathbf{u})}}{\partial \mathbf{u}}\right)^T \boldsymbol{\Psi}_2 = -\left(\frac{\partial \mathcal{K}}{\partial \mathbf{u}}\right)^T - \left(\frac{\partial \boldsymbol{\mathcal{S}}^{(\mathbf{u})}}{\partial \mathbf{u}}\right)^T \mathbf{u}_2 \tag{14}$$

$$\frac{\partial \boldsymbol{\mathcal{C}}^{(\mathbf{v})}}{\partial \mathbf{v}} \mathbf{v}_2 = -\left(\frac{\partial \mathcal{K}}{\partial \boldsymbol{\Phi}_1}\right)^T \tag{15}$$

$$\left(\frac{\partial \boldsymbol{\mathcal{C}}^{(\mathbf{v})}}{\partial \mathbf{v}}\right)^T \boldsymbol{\Phi}_2 = -\left(\frac{\partial \mathcal{K}}{\partial \mathbf{v}}\right)^T - \left(\frac{\partial \boldsymbol{\mathcal{S}}^{(\mathbf{v})}}{\partial \mathbf{v}}\right)^T \mathbf{v}_2. \tag{16}$$

III.B. Implementing KKT-matrix-vector Products

Second-order adjoint problems are uncommon in MDO applications, so some remarks are in order regarding the practical implementation of the KKT-matrix-vector product (12) and the second-order adjoint equations (13)–(16).

Notice that the second-order adjoints for the individual disciplines are decoupled; (13) and (15) can be solved independently in parallel. Subsequently, (14) and (16) can be solved in parallel. The decoupled nature of the second-order adjoints is a property inherited from the structure of the IDF formulation.

An efficient solution algorithm for the second-order adjoint equations is obviously an important aspect of the implementation of the KKT-matrix-vector product. Fortunately, the linear system matrices appearing in the second-order adjoint equations are the Jacobians or transposed Jacobians of the state equations. The infrastructure to solve systems involving the transposed Jacobians is typically available in gradient-based optimization algorithms already. Moreover, adapting this infrastructure to systems involving the Jacobian (i.e. linearized PDE problems) is straightforward.

Finally, a possible disadvantage of the proposed algorithm is the existence of second-derivative matrices in both (12) and the right-hand-side of the adjoint equations for $\mathbf{\Psi}_2$ and $\mathbf{\Phi}_2$. However, the second-derivative matrices always appear in products with vectors. These products are directional derivatives that can be approximated using finite-difference methods or algorithmic differentiation; see [21] for details in the single discipline case. It is also important to recognize that these are partial derivatives that do not involve expensive state or adjoint systems.

IV. An MDF-based Preconditioner for IDF

To be effective for practical problems, Krylov-subspace methods require a preconditioner. This is especially true for ill-conditioned saddle-point problems like the KKT-system (11). Indeed, we have found that GMRES without preconditioning has significant difficulty solving (11) and often stalls after a couple of iterations, even for modest-sized problems.

For the KKT-systems that arise in IDF, we have developed a preconditioner that approximates MDF. The preconditioner is inspired by the $\tilde{P}_2$ preconditioner proposed by Biros and Ghattas in Ref. [4] for full-space PDE-constrained optimization. Their preconditioner approximates a linearized reduced-space approach and, in essence, solves an approximate Schur-complement on the design variables. Here, we approximate a linearized MDF approach.

Consider the ideal case, where we precondition a set of IDF variables by inverting the KKT-matrix exactly:

$$\begin{bmatrix} H_{\mathbf{xx}} & H_{\bar{\mathbf{u}}\mathbf{x}} & H_{\bar{\mathbf{v}}\mathbf{x}} & A_{\mathbf{x}}^T \\ H_{\mathbf{x}\bar{\mathbf{u}}} & H_{\bar{\mathbf{u}}\bar{\mathbf{u}}} & H_{\bar{\mathbf{v}}\bar{\mathbf{u}}} & A_{\bar{\mathbf{u}}}^T \\ H_{\mathbf{x}\bar{\mathbf{v}}} & H_{\bar{\mathbf{u}}\bar{\mathbf{v}}} & H_{\bar{\mathbf{v}}\bar{\mathbf{v}}} & A_{\bar{\mathbf{v}}}^T \\ A_{\mathbf{x}} & A_{\bar{\mathbf{u}}} & A_{\bar{\mathbf{v}}} & 0 \end{bmatrix} \begin{pmatrix} \mathbf{z}_{\mathbf{x}} \\ \mathbf{z}_{\bar{\mathbf{u}}} \\ \mathbf{z}_{\bar{\mathbf{v}}} \\ \mathbf{z}_{\boldsymbol{\sigma}} \end{pmatrix} = \begin{pmatrix} \mathbf{w}_{\mathbf{x}} \\ \mathbf{w}_{\bar{\mathbf{u}}} \\ \mathbf{w}_{\bar{\mathbf{v}}} \\ \mathbf{w}_{\boldsymbol{\sigma}} \end{pmatrix},$$

where $\mathbf{w}^T = (\mathbf{w}_{\mathbf{x}}^T, \mathbf{w}_{\bar{\mathbf{u}}}^T, \mathbf{w}_{\bar{\mathbf{v}}}^T, \mathbf{w}_{\boldsymbol{\sigma}}^T)$ is the vector being preconditioned and $\mathbf{z}^T = (\mathbf{z}_{\mathbf{x}}^T, \mathbf{z}_{\bar{\mathbf{u}}}^T, \mathbf{z}_{\bar{\mathbf{v}}}^T, \mathbf{z}_{\boldsymbol{\sigma}}^T)$ denotes the preconditioned vector. Above, we have partitioned the Hessian and Jacobian according to $\mathbf{x}$, $\bar{\mathbf{u}}$ and $\bar{\mathbf{v}}$. For example

$$H_{\bar{\mathbf{u}}\mathbf{x}} \equiv \frac{\partial^2 \mathcal{L}}{\partial \bar{\mathbf{u}} \partial \mathbf{x}}, \qquad \text{and} \qquad A_{\mathbf{x}} \equiv \frac{\partial^2 \mathcal{L}}{\partial \mathbf{x} \partial \boldsymbol{\sigma}}.$$

The first step in deriving the preconditioner is to drop all second-order matrices from the KKT-matrix with the exception of $H_{\mathbf{xx}}$, which is replaced with the $n \times n$ identity matrix:

$$\begin{bmatrix} I & 0 & 0 & A_{\mathbf{x}}^T \\ 0 & 0 & 0 & A_{\bar{\mathbf{u}}}^T \\ 0 & 0 & 0 & A_{\bar{\mathbf{v}}}^T \\ A_{\mathbf{x}} & A_{\bar{\mathbf{u}}} & A_{\bar{\mathbf{v}}} & 0 \end{bmatrix} \begin{pmatrix} \mathbf{z}_{\mathbf{x}} \\ \mathbf{z}_{\bar{\mathbf{u}}} \\ \mathbf{z}_{\bar{\mathbf{v}}} \\ \mathbf{z}_{\boldsymbol{\sigma}} \end{pmatrix} = \begin{pmatrix} \mathbf{w}_{\mathbf{x}} \\ \mathbf{w}_{\bar{\mathbf{u}}} \\ \mathbf{w}_{\bar{\mathbf{v}}} \\ \mathbf{w}_{\boldsymbol{\sigma}} \end{pmatrix}. \tag{17}$$

Next, recall that the coupling constraints are $\bar{\mathbf{u}} - R^{(\mathbf{u})}\mathbf{u} = \mathbf{0}$ and $\bar{\mathbf{v}} - R^{(\mathbf{v})}\mathbf{v} = \mathbf{0}$. Thus, the submatrix

$$\begin{bmatrix} A_{\bar{\mathbf{u}}} & A_{\bar{\mathbf{v}}} \end{bmatrix} = \begin{bmatrix} I & -R^{(\mathbf{u})} \frac{\partial \mathbf{u}}{\partial \bar{\mathbf{v}}} \\ -R^{(\mathbf{v})} \frac{\partial \mathbf{v}}{\partial \bar{\mathbf{u}}} & I \end{bmatrix},$$

and its transpose are square $(r+s) \times (r+s)$ matrices. Moreover, these matrices are invertible, except in certain pathological cases described below. Therefore, we can solve (17) using a block LU decomposition that amounts to the following sequence of steps:

1. Solve (or approximately solve)
$$\begin{bmatrix} A_{\bar{\mathbf{u}}}^T \\ A_{\bar{\mathbf{v}}}^T \end{bmatrix} \mathbf{z}_{\boldsymbol{\sigma}} = \begin{pmatrix} \mathbf{w}_{\bar{\mathbf{u}}} \\ \mathbf{w}_{\bar{\mathbf{v}}} \end{pmatrix}$$
for the preconditioned dual $\mathbf{z}_{\boldsymbol{\sigma}}$.

2. Compute the preconditioned design using
$$\mathbf{z}_{\mathbf{x}} = \mathbf{w}_{\mathbf{x}} - A_{\mathbf{x}}^T \mathbf{z}_{\boldsymbol{\sigma}}.$$

3. Solve (or approximately solve)

$$\begin{bmatrix} A_{\bar{\mathbf{u}}} & A_{\bar{\mathbf{v}}} \end{bmatrix} \begin{pmatrix} \mathbf{z}_{\bar{\mathbf{u}}} \\ \mathbf{z}_{\bar{\mathbf{v}}} \end{pmatrix} = \mathbf{w}_{\boldsymbol{\sigma}} - A_{\mathbf{x}} \mathbf{z}_{\mathbf{x}}$$

for the preconditioned coupling variables $\mathbf{z}_{\bar{\mathbf{u}}}$ and $\mathbf{z}_{\bar{\mathbf{v}}}$.

The final detail that we must address is the solution of the linear systems in steps 1 and 3. The Jacobian blocks $A_{\bar{\mathbf{u}}}$ and $A_{\bar{\mathbf{v}}}$, like the Jacobian itself, are too expensive to form explicitly, so a matrix-free solver is necessary. For this work we use FGMRES [22] to solve these systems to a relative tolerance of 0.1 or a maximum of 10 iterations, which ever comes first. For the numerical experiments described below, the target tolerance is typically achieved in the first iteration.

The products required by FGMRES in steps 1 and 3 can be computed using second-order adjoints, as described earlier. However, the preconditioner contains several approximations (i.e. simplifying the Hessian blocks), which suggests that accurate Jacobian-vector products may be unnecessarily expensive. An inexpensive alternative, which we have found to be effective, is to replace the second-order adjoints with approximate adjoints obtained by using the primal or adjoint PDE preconditioner in place of the actual PDE Jacobian.

Finally, we remarked above that the Jacobian block $[A_{\bar{\mathbf{u}}} A_{\bar{\mathbf{v}}}]$ may fail to have an inverse in certain situations. Specifically, the preconditioner will be ill-posed if either one of the products

$$R^{(\mathbf{v})} \frac{\partial \mathbf{v}}{\partial \bar{\mathbf{u}}} R^{(\mathbf{u})} \frac{\partial \mathbf{u}}{\partial \bar{\mathbf{v}}} \qquad \text{or} \qquad R^{(\mathbf{u})} \frac{\partial \mathbf{u}}{\partial \bar{\mathbf{v}}} R^{(\mathbf{v})} \frac{\partial \mathbf{v}}{\partial \bar{\mathbf{u}}}$$

have nontrivial null spaces. We have yet to observe such a pathology, but further test cases are needed to improve our confidence in the preconditioner.

V. Results

V.A. Quasi-1D Flow in an Elastic Nozzle

We use the inverse design of a quasi-one-dimensional nozzle to investigate the effectiveness of using RSNK to solve IDF-formulated MDO problems. The flow through the nozzle is modelled using the quasi-one-dimensional Euler equations, while the structural deformations due to pressure are determined using the Euler-Bernoulli beam theory combined with axial stiffnesses. We begin by describing the individual discipline equations and their discretizations.

V.A.1. Quasi-1D Flow Solver

Let $A(x)$ denote the area of the (deformed) nozzle over the spatial domain $[0, 1]$. The quasi-one-dimensional Euler questions, governing the flow through the nozzle, are given by

$$\frac{\partial \mathcal{F}}{\partial x} - \mathcal{G} = 0, \qquad \forall\, x \in [0, 1], \tag{18}$$

where the flux and source are

$$\mathcal{F} = \begin{pmatrix} \rho u A \\ (\rho u^2 + p) A \\ u(e + p) A \end{pmatrix}, \qquad \text{and} \quad \mathcal{G} = \begin{pmatrix} 0 \\ p \frac{dA}{dx} \\ 0 \end{pmatrix},$$

respectively. The pressure is determined using the ideal gas law, $p = (\gamma - 1)(e - \frac{1}{2}\rho u^2)$. Hence, the unknowns are the density, ρ, momentum per unit volume, ρu, and energy per unit volume, e. These conservative variables become the state variables associated with the flow solver in the MDA. The Euler equations are coupled to the structural solver via the nozzle area, $A(x)$.

Boundary conditions for (18) are provided by the exact solution, which is determined using the Area-Mach-number relations. The exact solution is based on a stagnation temperature and pressure of 300K and 100 kPa, respectively. The specific gas constant is taken to be 287 J/(kg K) and the critical nozzle area is

$A^* = 0.8$. The equations and variables are nondimensionalized using the density and sound speed at the inlet.

Derivatives in the Euler equations are discretized using a third-order accurate summation-by-parts operator [23, 24], with boundary conditions imposed weakly using penalty terms [25, 26]. Scalar third-order accurate artificial dissipation is introduced to stabilize the solution [27].

V.A.2. *Linear Elastic Structural Solver*

The elastic deformation of the nozzle due to fluid pressure is modeled using a structural solver that combines axial stiffness,

$$\frac{wtE}{l}\delta_x = F_x, \tag{19}$$

with the Euler-Bernoulli beam theory,

$$\frac{Ewt^3}{12}\frac{d^4\delta_y}{dx^4} = q(x), \tag{20}$$

where E is the Young's Modulus, l is the axial length of the beam, and w and t are the cross sectional width and thickness, respectively. In this system, F_x defines the axial force on the beam and $q(x)$ the distributed transverse forces along the axial direction, x. While a computationally cheaper structural solver can be developed using only (19), we incorporate Euler-Bernoulli beam theory to capture the full affect of the pressure on the structure.

Discretizing (19) and (20) using two-dimensional beam finite elements yields the following structural discipline residual

$$\mathcal{C} = K\delta - F = 0, \tag{21}$$

where the solution vector δ is

$$\delta = \begin{pmatrix} \delta_x(x) & \delta_y(x) & \delta_\theta(x) \end{pmatrix}^T,$$

K is the stiffness matrix, F is the forcing vector, and $\delta_x(x)$, $\delta_y(x)$ and $\delta_\theta(x)$ are the horizontal, vertical and angular deformations in global coordinates, respectively, for each node of the finite element mesh. In the context of the MDA, these elements of the solution vector, δ, are the state variables associated with the structural discipline, while the forcing vector, F, is a function of nodal pressures calculated by the flow solver.

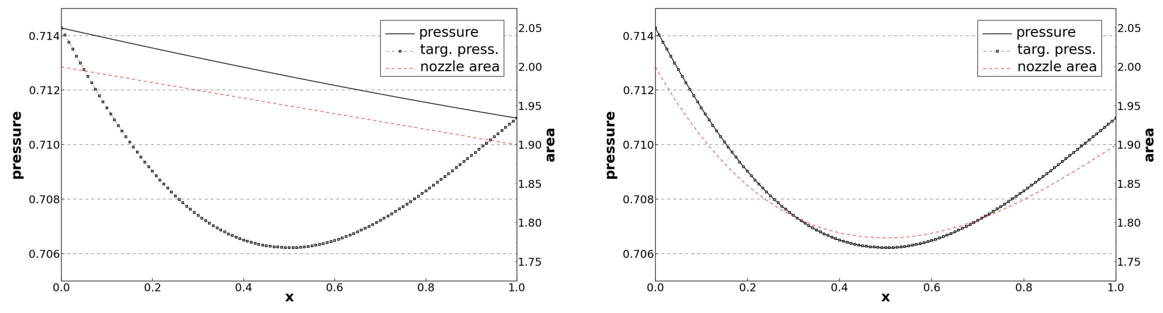
To simplify the MDA for this preliminary work, we fix the x-displacements; consequently, the objective function described below has no explicit[b] dependence on δ_x. In addition, the nozzle area is fixed at the inlet and outlet, so the left- and right-end nodes of the nozzle are fixed in both the x and y directions; however, all nodes are permitted to have angular deformations.

V.A.3. *MDA Primal and Adjoint Solvers*

We also adopt a Newton-Krylov method to solve the MDA, for example, at each major iteration of MDF. At each Newton step of the multidisciplinary analysis, the linearized MDA is solved using FGMRES preconditioned with a block Jacobi method. Specifically, the flow-solver block is preconditioned using an LU factorization of a first-order accurate discretization based on the nearest-neighbors, while the structural solver block is preconditioned using a nested conjugate gradient method, with an absolute tolerance of 10^{-5}. The MDA is solved to a relative tolerance of 10^{-7}.

Similarly, the adjoint problem for the MDF formulation is also solved to a relative tolerance of 10^{-7} using FGMRES. This solution is preconditioned using the transposed version of the block Jacobi preconditioner used for the primal system.

[b]There remains, of course, an implicit dependence via the influence of δ_x on the pressure.



(a) initial (b) optimized

Figure 1. Deformed nozzle areas and corresponding pressure distributions for the initial design (left) and optimized design (right).

V.A.4. Nozzle Definition

The undeformed nozzle area is parameterized using a cubic b-spline with an open uniform knot vector. The control points at the ends of the nozzle are fixed such that the area satisfies $A(0) = 2$ and $A(1) = 1.9$. The design variables are the remaining B-spline control points. The initial control-point values are set to recover a nozzle with a linearly varying area between the fixed inlet and outlet. Figure 1(a) shows the initial deformed nozzle area and the corresponding pressure distribution.

The structural solver discretizes the nozzle by querying the b-spline for areas at evenly spaced x coordinates in the domain. These areas are transformed into nodal y coordinates under the assumption of a rectangular nozzle cross-section with height $h = 200$ and width $w = 0.01$. Following the same assumption, the structural solver is responsible for calculating the displaced areas needed by the flow solver at each Newton iteration of the MDA evaluation.

The comparisons in this section have been established on a nozzle definition with a unit nozzle length of $l = 1$, nozzle wall thickness of $t = 0.01$, and nozzle material Young's Modulus of $E = 10^7$.

V.A.5. Objective Function

The objective function corresponds to an inverse problem for the pressure:

$$\mathcal{J} = \int_0^1 \frac{1}{2}(p - p_{\text{targ}})^2 \, dx. \tag{22}$$

The objective function is discretized using the SBP quadrature from the flow solver discretization [28]. The target (undeformed) nozzle area is a cubic function of x that passes through the inlet and outlet areas and has a local minimum at $x = 0.5$ given by $A(0.5) = 1.8$. The target pressure p_{targ} is found, during a preprocessing step, by solving the coupled MDA system with this target nozzle area. Figure 1 shows the target pressure.

V.B. Optimization Results

For these results, we apply the Reduced-Space inexact Newton-Krylov (RSNK) algorithm to both MDF and IDF formulations of the nozzle-optimization problem. Figure 1(b) shows the optimized pressure and nozzle area, which are successfully recovered by both formulations. Note that the recovered pressures are indistinguishable from the target pressures.

Figure 2 shows a comparison of the convergence histories from MDF and IDF. The results shown correspond to 20 design variables and a discretization with 121 nodes. The computational cost is measured in terms of equivalent MDA evaluations; specifically, the total number of PDE preconditioner calls used during the optimization is divided by the number of preconditioner calls made by a single evaluation of the MDA applied to the target area. For example, the MDA with 121 nodes requires 42 preconditioner calls. The convergence histories shown are typical of most sets of design variables considered here.

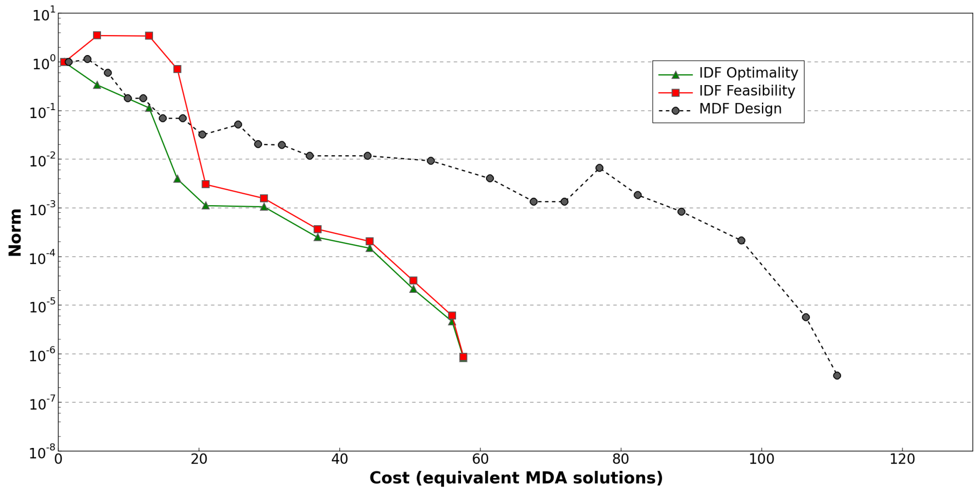


Figure 2. Convergence histories of MDF and IDF for the nozzle-optimization problem with 20 design variables.

IDF has two distinct norms that the optimization algorithm must drive to zero. The optimality norm, shown in green, is the gradient norm associated with the (reduced-space) Lagrangian. The IDF feasibility norm, shown in red, is associated with the coupling-variable constraint feasibilities within the IDF formulation, described in section II above. An IDF solution is considered converged if the optimality and feasibility norms are reduced 6 and 3 orders of magnitude, respectively, relative to their initial values; in practice, the feasibility norm is reduced significantly more, as illustrated in Figure 2. An MDF solution is considered converged when the gradient norm is reduced 6 orders of magnitude relative to its initial value.

Note that the norms drop rapidly during the last few iterations of both the IDF and MDF formulations. This superlinear convergence is indicative of Newton's method. One of IDF's potential advantages is its relative freedom to explore intermediate designs that may not be feasible with respect to the entire multidisciplinary system. This freedom may allow IDF to reach superlinear convergence more rapidly than MDF.

Finally, we are also interested in how these algorithms scale with number of design variables. Figure 3 plots the number of equivalent MDA evaluations required to converge the nozzle problem for a range of design variables between 10 and 30. For this problem, IDF is at least as good as MDF, and often 30% − 40% faster, especially at higher design variables. While these results are promising, additional tests cases are necessary to demonstrated that this behavior holds true for large-scale MDO problems involving hundreds of design and thousands of coupling variables.

VI. Conclusions

We investigated reduced-space Newton-Krylov (RSNK) algorithms in the solution of multidisciplinary design optimization (MDO) problems posed in the multidisciplinary feasible (MDF) and individual discipline feasible (IDF) formulations. To the best of our knowledge, this is the first example of using RSNK to solve an MDO problem, whether formulated with MDF or IDF.

Our results demonstrate the effectiveness of using RSNK to solve an IDF-formulated MDO problem. Not only is the IDF formulation at least as efficient as MDF for this nozzle optimization problem, but it is also scales well with the number of design variables.

The KKT-system for IDF is an ill-conditioned saddle-point problem that cannot be solved efficiently without preconditioning. Thus, we developed a novel preconditioner for the IDF formulation that is based on an approximate solution to the (linearized) MDF problem. This IDF-Schur preconditioner was critical to the effectiveness of RSNK applied to IDF.

We believe that IDF has languished as a formulation for high-fidelity MDO problems, not because of issues inherent to the architecture, but due to the limitations of conventional optimization algorithms applied to the formulation. In particular, conventional algorithms scale poorly with the number of design variables

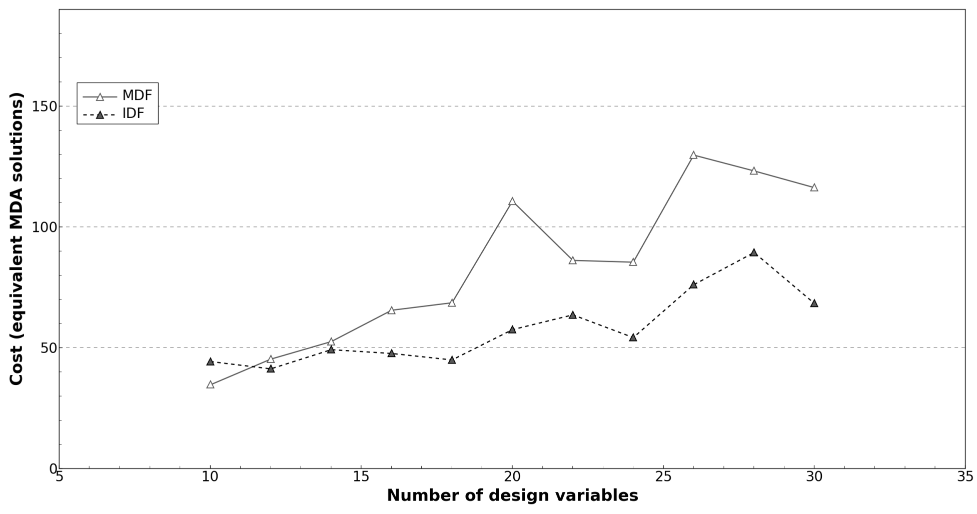


Figure 3. Computational cost scaling with respect to the number of design variables.

and require the coupling-constraint Jacobian explicitly. In contrast, the proposed matrix-free Newton-Krylov algorithm, applied in the reduced space, eliminates these issues.

Our results suggest that matrix-free Newton-Krylov algorithms have the potential to make IDF practical and efficient for large-scale physics-based MDO problems and, thereby, simplify their implementation. Therefore, our future work is focused on IDF optimization problems involving high-fidelity, three-dimensional, aero-structural analyses. This should help illuminate whether the results observed in this paper are restricted to simple, one-dimensional problems, or whether reduced-space Newton-Krylov algorithms can make IDF a powerful and practical alternative for larger-scale problems as well.

Acknowledgments

This material is based upon work supported by the National Science Foundation under Grant No. 1332819. The authors gratefully acknowledge this support.

References

[1]Kenway, G., Kennedy, G., and Martins, J., "A Scalable Parallel Approach for High-Fidelity Aerostructural Analysis and Optimization," *53rd AIAA/ASME/ASCE/AHS/ASC Structures, Structural Dynamics and Materials Conference*, No. AIAA-2012-1922, Honolulu, Hawaii, April 2012.

[2]Haftka, R. T., Sobieszczanski-Sobieski, J., and Padula, S. L., "On options for interdisciplinary analysis and design optimization," *Structural and Multidisciplinary Optimization*, Vol. 4, No. 2, 1992, pp. 65–74, 10.1007/BF01759919.

[3]Cramer, E. J., Dennis, J. E., Frank, P. D., Lewis, R. M., and Shubin, G. R., "Problem Formulation for Multidisciplinary Optimization," *SIAM Journal on Optimization*, Vol. 4, No. 4, Nov. 1994, pp. 754–776.

[4]Biros, G. and Ghattas, O., "Parallel Lagrange-Newton-Krylov-Schur methods for PDE-constrained optimization. Part I: the Krylov-Schur solver," *SIAM Journal on Scientific Computing*, Vol. 27, 2005, pp. 687–713.

[5]Biros, G. and Ghattas, O., "Parallel Lagrange-Newton-Krylov-Schur methods for PDE-constrained optimization. Part II: the Lagrange-Newton solver and its application to optimal control of steady viscous flows," *SIAM Journal on Scientific Computing*, Vol. 27, 2005, pp. 714–739.

[6]Zhu, C., Byrd, R. H., Lu, P., and Nocedal, J., "Algorithm 778: L-BFGS-B: Fortran subroutines for large-scale bound-constrained optimization," *ACM Transactions on Mathematical Software (TOMS)*, Vol. 23, No. 4, 1997, pp. 550–560.

[7]Wächter, A. and Biegler, L. T., "On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming," *Mathematical Programming*, Vol. 106, No. 1, May 2006, pp. 25–57.

[8]Waltz, R. A. and Nocedal, J., "KNITRO user's manual," *Northwestern University, Evanston, Illinois, Technical Report OTC-2003/5*, 2003.

[9]Liu, D. C. and Nocedal, J., "On the limited memory BFGS method for large scale optimization," *Mathematical Programming*, Vol. 45, 1989, pp. 503–528.

[10]Nocedal, J. and Wright, S. J., *Numerical Optimization*, Springer–Verlag, Berlin, Germany, 2nd ed., 2006.

[11]Martins, J. R. R. A. and Lambe, A. B., "Multidisciplinary design optimization: A Survey of architectures," *AIAA Journal*, 2013, (In press).

[12]Martins, J. R. R. A., Alonso, J. J., and Reuther, J. J., "High-fidelity aerostructural design optimization of a supersonic business jet," *Journal of Aircraft*, Vol. 41, No. 3, May 2004, pp. 863–873.

[13]Maute, K., Nikbay, M., and Farhat, C., "Coupled Analytical Sensitivity Analysis and Optimization of Three-Dimensional Nonlinear Aeroelastic Systems," *AIAA Journal*, Vol. 39, No. 11, Nov. 2001, pp. 2051–2061.

[14]Maute, K., Nikbay, M., and Farhat, C., "Sensitivity analysis and design optimization of three-dimensional non-linear aeroelastic systems by the adjoint method," *International Journal for Numerical Methods in Engineering*, Vol. 56, No. 6, Feb. 2003, pp. 911–933.

[15]Byrd, R. H., Curtis, F. E., and Nocedal, J., "An Inexact SQP Method for Equality Constrained Optimization," *SIAM Journal on Optimization*, Vol. 19, No. 1, 2008, pp. 351–369, 10.1137/060674004.

[16]Ridzal, D., *Trust–Region SQP Methods with Inexact Linear System Solves for Large–Scale Optimization*, Ph.D. thesis, Department of Computational and Applied Mathematics, Rice University, Houston, TX, April 2006.

[17]Curtis, F. E., Nocedal, J., and Wächter, A., "A Matrix-Free Algorithm for Equality Constrained Optimization Problems with Rank Deficient Jacobians," *SIAM Journal on Optimization*, Vol. 20, No. 3, 2009, pp. 1224–1249.

[18]Curtis, F. E., Schenk, O., and Wächter, A., "An Interior-Point Algorithm for Large-Scale Nonlinear Optimization with Inexact Step Computations," *SIAM Journal on Scientific Computing*, Vol. 32, No. 6, 2010, pp. 3447–3475.

[19]Ide, H. and Levine, M., "Use of second order CFD generated global sensitivity derivatives for coupled problems," *30th AIAA/ASME/ASCE/AHS Structures, Structural Dynamics and Materials Conference*, No. AIAA-89-1178, American Institute of Aeronautics and Astronautics, Mobile, Alabama, 1989.

[20]Wang, Z., Navon, I. M., Dimet, F. X., and Zou, X., "The second order adjoint analysis: Theory and applications," *Meteorology and Atmospheric Physics*, Vol. 50, 1992, pp. 3–20.

[21]Hicken, J. E. and Alonso, J. J., "Comparison of Reduced- and Full-space Algorithms for PDE-constrained Optimization," *51st AIAA Aerospace Sciences Meeting*, No. AIAA–2013–1043, Grapevine, Texas, United States, Jan. 2013.

[22]Saad, Y., "A flexible inner-outer preconditioned GMRES algorithm," *SIAM Journal on Scientific and Statistical Computing*, Vol. 14, No. 2, 1993, pp. 461–469.

[23]Kreiss, H. O. and Scherer, G., "Finite element and finite difference methods for hyperbolic partial differential equations," *Mathematical Aspects of Finite Elements in Partial Differential Equations*, edited by C. de Boor, Mathematics Research Center, the University of Wisconsin, Academic Press, 1974.

[24]Strand, B., "Summation by parts for finite difference approximations for d/dx," *Journal of Computational Physics*, Vol. 110, No. 1, 1994, pp. 47–67.

[25]Funaro, D. and Gottlieb, D., "A new method of imposing boundary conditions in pseudospectral approximations of hyperbolic equations," *Mathematics of Computation*, Vol. 51, No. 184, Oct. 1988, pp. 599–613.

[26]Carpenter, M. H., Gottlieb, D., and Abarbanel, S., "Time-stable boundary conditions for finite-difference schemes solving hyperbolic systems: methodology and application to high-order compact schemes," *Journal of Computational Physics*, Vol. 111, No. 2, 1994, pp. 220–236.

[27]Mattsson, K., Svärd, M., and Nordström, J., "Stable and accurate artificial dissipation," *Journal of Scientific Computing*, Vol. 21, No. 1, 2004, pp. 57–79.

[28]Hicken, J. E. and Zingg, D. W., "Summation-by-parts operators and high-order quadrature," *Journal of Computational and Applied Mathematics*, Vol. 237, No. 1, Jan. 2013, pp. 111–125.