Struct Multidisc Optim (2017) 56:1429–1446
DOI 10.1007/s00158-017-1734-0

RESEARCH PAPER

# Matrix-free algorithm for the optimization of multidisciplinary systems

**Alp Dener**[1] · **Jason E. Hicken**[1]

Received: 6 April 2017 / Revised: 18 May 2017 / Accepted: 1 June 2017 / Published online: 23 June 2017


**Abstract** Multidisciplinary engineering systems are usually modeled by coupling software components that were developed for each discipline independently. The use of disparate solvers complicates the optimization of multidisciplinary systems and has been a long-standing motivation for optimization architectures that support modularity. The individual discipline feasible (IDF) formulation is particularly attractive in this respect. IDF achieves modularity by introducing optimization variables and constraints that effectively decouple the disciplinary solvers during each optimization iteration. Unfortunately, the number of variables and constraints can be significant, and the IDF constraint Jacobian required by most conventional optimization algorithms is prohibitively expensive to compute. Furthermore, limited-memory quasi-Newton approximations, commonly used for large-scale problems, exhibit linear convergence rates that can struggle with the large number of design variables introduced by the IDF formulation. In this work, we show that these challenges can be overcome using a reduced-space inexact-Newton-Krylov algorithm. The proposed algorithm avoids the need for the explicit constraint Jacobian and Hessian by using a Krylov iterative method to solve the Newton steps. The Krylov method requires matrix-vector products, which can be evaluated in a matrix-free manner using second-order adjoints. The Krylov method also needs to be preconditioned, and a key contribution of this work is a novel and effective preconditioner that is based on approximating a monolithic solution of the (linearized) multidisciplinary system. We demonstrate the efficacy of the algorithm by comparing it with the popular multidisciplinary feasible formulation on two test problems.

**Keywords** Multidisciplinary design optimization · Individual discipline feasible · Inexact-Newton-Krylov · Matrix-free · Second-order adjoint · Preconditioning

## 1 Introduction

The field of multidisciplinary design optimization (MDO) aims to develop methods for the optimization of engineering systems involving more than one physical discipline or subsystem. The field of MDO seeks to exploit unintuitive trade-offs emerging from the interaction between the disciplines that might not be captured or exploited by a sequential (and usually suboptimal) optimization process.

We are particularly interested in MDO applications that require one or more of the disciplines to be modeled using high-fidelity analyses, such as partial-differential equations (PDEs), in order to achieve sufficient accuracy in engineering quantities of interest. Examples include flexible aerodynamic surfaces that experience significant deformations, and boundary-layer ingestion systems where changes to the airframe have a strong effect on engine airflow. In these situations, single-discipline analysis often fails to capture the phenomena of interest, and high-fidelity coupled simulations are necessary to accurately capture the governing physics.

On the other hand, domain experts have developed specialized strategies for discipline-specific PDE models to

✉ Alp Dener
denera@rpi.edu

Jason E. Hicken
hickej2@rpi.edu

[1] Mechanical, Aerospace and Nuclear Engineering, Rensselaer Polytechnic Institute, 110 8th Street, Troy, NY 12180, USA

deal with their unique characteristics. This has led to the development and use of stand-alone solvers for each discipline, employing different discretization, globalization, solution and preconditioning techniques that may be either unsuitable or unavailable for other disciplines.

Stationary iterative methods offer a straightforward way to couple these disparate PDE solvers in multidisciplinary analysis, but they also involve repeated sequential solutions of each discipline within a single optimization iteration. Monolithic methods that solve all disciplines simultaneously at once can avoid this expense, but they also require greater intrusion into the solvers. These observations present a distinct need for a general-purpose MDO algorithm to be both modular and efficient in its coupling of disparate PDE solvers.

The individual discipline feasible (IDF) formulation aims to address this need for efficient modularity (Haftka et al. 1992; Cramer et al. 1994). The IDF formulation separates the disciplines from each other by introducing coupling variables that capture the cross-discipline dependencies. These coupling variables are controlled by the optimization algorithm as part of the design space, and permit the disciplines to be evaluated independently from each other. In order to ensure multidisciplinary feasibility of the optimum solution, the coupling variables are then constrained to be equal to the actual cross-discipline variables they mimic. As a result, IDF avoids an expensive fully-coupled multidisciplinary analysis at every optimization iteration while leveraging existing discipline-specific solution infrastructure.

Unfortunately, the IDF formulation also brings a set of challenges that must be addressed. The coupling variables in a high-fidelity problem can introduce thousands or tens of thousands of new degrees of freedom and just as many nonlinear constraints. State-of-the-art gradient-based optimization algorithms frequently use limited-memory quasi-Newton approximations of the Hessian for such large-scale problems (Schittkowski 1986; Gill et al. 2005), and it is known that these approaches scale unfavorably with the size of the design space (Nash and Nocedal 1991). More significantly, most gradient-based optimization algorithms require the explicit constraint Jacobian to be provided, but computing this Jacobian for the IDF problem is prohibitively expensive as it requires one adjoint solution for each coupling constraint. The primary contribution of this work is to show that these disadvantages can be eliminated by using matrix-free Newton-Krylov optimization algorithms.

The remaining paper is organized as follows. We begin by reviewing background material and introducing notation in Section 2. We then describe a reduced-space inexact-Newton-Krylov (RSNK) algorithm in Section 3, including a description of the method we use to compute KKT-matrix-vector products within RSNK in Section 3.2. An effective preconditioner for the IDF primal-dual matrix is described in Section 3.3. Numerical results comparing MDF and IDF formulations are presented in Section 4. The paper concludes with a summary of findings and discussion of future work in Section 5.

## 2 MDO architecture review: SAND, MDF and IDF

We begin our discussion with a brief review of monolithic MDO architectures. We refer the reader to Martins and Lambe (2013) for a more comprehensive review of MDO architectures.

In order to effectively discuss MDO architectures, we first introduce a model multidisciplinary analysis (MDA) problem. For simplicity, we will restrict our discussion to two physical disciplines represented by the algebraic state equations

$$\begin{aligned} \boldsymbol{\mathcal{R}_u}(\boldsymbol{u}, \boldsymbol{\mathcal{E}_u}(\boldsymbol{v}), \boldsymbol{x}) &= \boldsymbol{0}, \\ \boldsymbol{\mathcal{R}_v}(\boldsymbol{v}, \boldsymbol{\mathcal{E}_v}(\boldsymbol{u}), \boldsymbol{x}) &= \boldsymbol{0}, \end{aligned} \tag{1}$$

where $\boldsymbol{x} \in \mathbb{R}^n$ are the design variables, and $\boldsymbol{u} \in \mathbb{R}^p$ and $\boldsymbol{v} \in \mathbb{R}^q$ are state variables associated with each discipline. These state equations depend on each other through $\boldsymbol{\mathcal{E}_u}(\boldsymbol{v}) \in \mathbb{R}^s$ and $\boldsymbol{\mathcal{E}_v}(\boldsymbol{u}) \in \mathbb{R}^r$, where $\boldsymbol{\mathcal{E}_u} : \mathbb{R}^q \to \mathbb{R}^s$ and $\boldsymbol{\mathcal{E}_v} : \mathbb{R}^p \to \mathbb{R}^r$ are functions that extract or compute the information that couples the two disciplines together.

*Remark 1* In the applications of interest, the MDA defined by (1) represents a set of discretized PDEs. For instance, in an aero-structural problem, $\mathcal{R}_u$ might be the Euler equations discretized using a finite-volume scheme, and $\mathcal{R}_v$ might be the equations of linear elasticity discretized using the finite-element method. In this example, the state $\boldsymbol{u}$ would denote the conservative flow variables averaged over each volume in the mesh, and $\boldsymbol{v}$ would be the nodal displacements of the finite-element model. Furthermore, $\boldsymbol{\mathcal{E}_v}$ uses the state $\boldsymbol{u}$ to compute the pressure on the wing surface, while $\boldsymbol{\mathcal{E}_u}$ uses $\boldsymbol{v}$ to determine the structural deformations of the outer mold line.

Next, we introduce the objective function, $\mathcal{J} : \mathbb{R}^n \times \mathbb{R}^p \times \mathbb{R}^q \to \mathbb{R}$, and the corresponding multidisciplinary design optimization problem

$$\begin{aligned} \underset{\boldsymbol{x},\boldsymbol{u},\boldsymbol{v}}{\text{minimize}} \quad & \mathcal{J}(\boldsymbol{x}, \boldsymbol{u}, \boldsymbol{v}), \\ \text{subject to} \quad & \boldsymbol{\mathcal{R}_u}(\boldsymbol{u}, \boldsymbol{\mathcal{E}_u}(\boldsymbol{v}), \boldsymbol{x}) = \boldsymbol{0}, \\ & \boldsymbol{\mathcal{R}_v}(\boldsymbol{v}, \boldsymbol{\mathcal{E}_v}(\boldsymbol{u}), \boldsymbol{x}) = \boldsymbol{0}, \end{aligned} \tag{2}$$

where the discipline residuals are expressed as constraints to be satisfied by the optimization. In the full-space formulation (2), the optimization variables include both the design variables, $\boldsymbol{x}$, and the state variables $\boldsymbol{u}$ and $\boldsymbol{v}$. In the context of MDO, (2) is also referred to as Simultaneous Analysis and Design (SAND) (Haftka 1985).

Efficient full-space PDE-constrained optimization algorithms have been developed and applied to large-scale single-discipline problems (Ta'asan et al. 1992; Herskovits et al. 2005; Biros and Ghattas 2005a, b; Özkaya and Gauger 2009) however, the full-space formulation can be difficult to implement and poses challenges to modularity. In particular, as discussed in the introduction, PDE models often take advantage of specialized solution techniques for efficiency and robustness. This is especially true for nonlinear PDEs, which may require tailored globalization strategies to ensure robust convergence. A general-purpose full-space algorithm cannot easily encode all such globalization strategies. The complexity of the task also increases with each added discipline and, to the best of our knowledge, no one has solved high-fidelity MDO problems using full-space methods.

These difficulties motivate the use of the reduced-space formulation, given by

$$\begin{aligned} \underset{\boldsymbol{x}}{\text{minimize}} \quad & \mathcal{J}(\boldsymbol{x}, \boldsymbol{u}(\boldsymbol{x}), \boldsymbol{v}(\boldsymbol{x})) \\ \text{governed by} \quad & \mathcal{R}_{\boldsymbol{u}}(\boldsymbol{u}, \mathcal{E}_{\boldsymbol{u}}(\boldsymbol{v}), \boldsymbol{x}) = \boldsymbol{0}, \\ & \mathcal{R}_{\boldsymbol{v}}(\boldsymbol{v}, \mathcal{E}_{\boldsymbol{v}}(\boldsymbol{u}), \boldsymbol{x}) = \boldsymbol{0}. \end{aligned} \tag{3}$$

In the reduced-space formulation, the state variables are implicit functions of the design $\boldsymbol{x}$ via the state equations. Thus, the optimization variables consist only of the design variables. In the context of MDO, the reduced-space formulation (3) is referred to as multidisciplinary feasible (MDF).

Under the MDF formulation, every evaluation of the objective function $\mathcal{J}$ requires a complete solution of the coupled state equations (1). Block-Jacobi and block-Gauss-Seidel are popular choices for solving the coupled state equations in MDF, but these iterative methods have been shown to be considerably less efficient than monolithic Newton-Krylov MDAs (Kennedy and Martins 2010). Conversely, implementing such a monolithic MDA with existing legacy solvers requires significant intrusion into the source code, and may be impractical with commercial software. Furthermore, the use of gradient-based algorithms necessitates expensive (and potentially inaccurate) finite-difference approximations or a coupled-adjoint implementation. Despite these challenges, there has been significant success applying the MDF formulation to, for example, aero-elastic optimization problems (Maute et al. 2001, 2003; Martins et al. 2004; Kenway et al. 2014).

In summary, there are clear trade-offs between the full- and reduced-space formulations of solving multidisciplinary optimization problems. The SAND formulation may achieve better optimization efficiency, but it comes at the cost of greater time investment into software development. The MDF formulation can facilitate the reuse of existing PDE solvers, but at the expense of fully coupled MDAs at each optimization iteration.

This trade-off motivates a compromise, which we believe is provided by the individual discipline feasible (IDF) formulation (Haftka et al. 1992; Cramer et al. 1994):

$$\begin{aligned} \underset{\boldsymbol{x}, \bar{\boldsymbol{u}}, \bar{\boldsymbol{v}}}{\text{minimize}} \quad & \mathcal{J}(\boldsymbol{x}, \boldsymbol{u}(\boldsymbol{x}, \bar{\boldsymbol{v}}), \boldsymbol{v}(\boldsymbol{x}, \bar{\boldsymbol{u}})), \\ \text{subject to} \quad & \mathcal{E}_{\boldsymbol{v}}(\boldsymbol{u}) - \bar{\boldsymbol{u}} = \boldsymbol{0}, \\ & \mathcal{E}_{\boldsymbol{u}}(\boldsymbol{v}) - \bar{\boldsymbol{v}} = \boldsymbol{0}, \\ \text{governed by} \quad & \mathcal{R}_{\boldsymbol{u}}(\boldsymbol{u}, \bar{\boldsymbol{v}}, \boldsymbol{x}) = \boldsymbol{0}, \\ & \mathcal{R}_{\boldsymbol{v}}(\boldsymbol{v}, \bar{\boldsymbol{u}}, \boldsymbol{x}) = \boldsymbol{0}, \end{aligned} \tag{4}$$

which modifies (3) by introducing coupling variables, $\bar{\boldsymbol{u}} \in \mathbb{R}^r$ and $\bar{\boldsymbol{v}} \in \mathbb{R}^s$, and their corresponding coupling constraints. Under the IDF formulation, the states are still expressed as implicit functions of design variables, $\boldsymbol{x}$, but they are no longer directly dependent on each other. This is because the state equations for each discipline can be solved independently using the coupling variables prescribed by the optimization algorithm. Consequently, IDF avoids a full MDA at every evaluation of the objective function. This advantage extends to sensitivity analysis as well, where the adjoint solutions are also decoupled.

Unfortunately, the decoupled state and adjoint equations come at a cost, as IDF presents two significant disadvantages. First, the introduction of the coupling variables increases the size of the optimization problem, which can negatively impact convergence rates for limited-memory quasi-Newton approximations of the Hessian. Secondly, and more significantly, the cost of evaluating the Jacobian of the IDF coupling constraints is prohibitively expensive for most multidisciplinary problems modeled with PDEs. To illustrate, the IDF formulation applied to a three-dimensional aero-structural problem could yield on the order of $10^3$–$10^4$ new degrees of freedom in the design space, and require just as many adjoint solutions at every optimization iteration to form the constraint Jacobian. In the next section, we describe an optimization framework that addresses both of these concerns.

## 3 Inexact-Newton-Krylov algorithm for IDF

In the previous section we motivated the IDF formulation and highlighted its challenges, namely the number of optimization variables and constraints that it introduces. In this section we present an algorithm that resolves these challenges. At the heart of this algorithm is an inexact Newton-Krylov approach, which we review in Section 3.1. The Newton-Krylov approach relies on matrix-vector products and an effective preconditioner, which are covered in Sections 3.2 and 3.3, respectively. The section concludes with an overview of the complete algorithm and implementation details.

### 3.1 Newton-Krylov optimization

In order to describe Newton-Krylov optimization in a more general form, we first combine the coupling constraints into a single constraint vector and the state equations into a single residual vector:

$$\mathcal{C}(\boldsymbol{y}, \boldsymbol{w}) = \begin{bmatrix} \mathcal{E}_v(\boldsymbol{u}) - \bar{\boldsymbol{u}} \\ \mathcal{E}_u(\boldsymbol{v}) - \bar{\boldsymbol{v}} \end{bmatrix},$$
$$\mathcal{R}(\boldsymbol{y}, \boldsymbol{w}) = \begin{bmatrix} \mathcal{R}_u(\boldsymbol{x}, \boldsymbol{u}, \bar{\boldsymbol{v}}) \\ \mathcal{R}_v(\boldsymbol{x}, \boldsymbol{v}, \bar{\boldsymbol{u}}) \end{bmatrix}, \tag{5}$$

where $\boldsymbol{w}^T = [\boldsymbol{u}^T \ \boldsymbol{v}^T]$ denotes the multidisciplinary state and $\boldsymbol{y}^T = [\boldsymbol{x}^T \ \bar{\boldsymbol{u}}^T \ \bar{\boldsymbol{v}}^T]$ denotes the optimization variables. With this notation, the IDF optimization statement (4) simplifies to

$$\begin{aligned} \underset{\boldsymbol{y}}{\text{minimize}} \quad & \mathcal{J}(\boldsymbol{y}, \boldsymbol{w}(\boldsymbol{y})), \\ \text{subject to} \quad & \mathcal{C}(\boldsymbol{y}, \boldsymbol{w}(\boldsymbol{y})) = \boldsymbol{0}, \\ \text{governed by} \quad & \mathcal{R}(\boldsymbol{y}, \boldsymbol{w}) = \boldsymbol{0}. \end{aligned}$$

This simplification will allow us to define a general-purpose optimization algorithm applicable to both single and multidisciplinary problems with an arbitrary number of disciplines.

Next, we introduce the Lagrangian

$$\mathcal{L}(\boldsymbol{y}, \boldsymbol{\lambda}) = \mathcal{J}(\boldsymbol{y}, \boldsymbol{w}(\boldsymbol{y})) + \boldsymbol{\lambda}^T \mathcal{C}(\boldsymbol{y}, \boldsymbol{w}(\boldsymbol{y})), \tag{6}$$

where $\boldsymbol{\lambda}$ are the Lagrange multipliers associated with the equality constraints. Differentiating (6) with respect to $\boldsymbol{y}$ and $\boldsymbol{\lambda}$ produces the Karush-Kuhn-Tucker (KKT) first-order optimality conditions that must be satisfied by the local optimum,

$$\frac{d\mathcal{L}}{d\boldsymbol{y}} = \frac{d\mathcal{J}}{d\boldsymbol{y}} + \boldsymbol{\lambda}^T \frac{d\mathcal{C}}{d\boldsymbol{y}} = \boldsymbol{0}, \tag{7a}$$
$$\frac{d\mathcal{L}}{d\boldsymbol{\lambda}} = \mathcal{C} = \boldsymbol{0}, \tag{7b}$$

where $d/d\boldsymbol{y}$ denotes a total derivative operator with respect to $\boldsymbol{y}$, and $d/d\boldsymbol{\lambda}$ denotes a total derivative with respect to $\boldsymbol{\lambda}$.

The total derivatives in (7) are evaluated using the adjoint method (see, e.g. Jameson 1989), which we will review by introducing the full-space Lagrangian,

$$\hat{\mathcal{L}}(\boldsymbol{y}, \boldsymbol{\lambda}, \boldsymbol{w}, \boldsymbol{\Psi}) = \mathcal{J}(\boldsymbol{y}, \boldsymbol{w}) + \boldsymbol{\lambda}^T \mathcal{C}(\boldsymbol{y}, \boldsymbol{w}) + \boldsymbol{\Psi}^T \mathcal{R}(\boldsymbol{y}, \boldsymbol{w}), \tag{8}$$

where the state equations are now imposed as constraints and associated with the Lagrange multipliers in $\boldsymbol{\Psi}$. Differentiating (8) with respect to its inputs produces the KKT conditions for the full-space problem,

$$\frac{\partial\hat{\mathcal{L}}}{\partial\boldsymbol{y}} = \frac{\partial\mathcal{J}}{\partial\boldsymbol{y}} + \boldsymbol{\lambda}^T \frac{\partial\mathcal{C}}{\partial\boldsymbol{y}} + \boldsymbol{\Psi}^T \frac{\partial\mathcal{R}}{\partial\boldsymbol{y}} = \boldsymbol{0}, \tag{9a}$$
$$\frac{\partial\hat{\mathcal{L}}}{\partial\boldsymbol{\lambda}} = \mathcal{C} = \boldsymbol{0}, \tag{9b}$$
$$\frac{\partial\hat{\mathcal{L}}}{\partial\boldsymbol{w}} = \frac{\partial\mathcal{J}}{\partial\boldsymbol{w}} + \boldsymbol{\lambda}^T \frac{\partial\mathcal{C}}{\partial\boldsymbol{w}} + \boldsymbol{\Psi}^T \frac{\partial\mathcal{R}}{\partial\boldsymbol{w}} = \boldsymbol{0}, \tag{10}$$
$$\frac{\partial\hat{\mathcal{L}}}{\partial\boldsymbol{\Psi}} = \mathcal{R} = \boldsymbol{0}, \tag{11}$$

where all the derivatives above denote partial derivatives rather than total derivatives.

Recall that in the reduced-space formulation, the state variables are implicit functions of the design via the state equations. This means that (11) is satisfied at every design point, and can be eliminated from the problem. The same applies to (10), which we rewrite below in residual form:

$$\mathcal{S}(\boldsymbol{y}, \boldsymbol{\lambda}, \boldsymbol{w}, \boldsymbol{\Psi}) = \frac{\partial\mathcal{J}}{\partial\boldsymbol{w}} + \boldsymbol{\lambda}^T \frac{\partial\mathcal{C}}{\partial\boldsymbol{w}} + \boldsymbol{\Psi}^T \frac{\partial\mathcal{R}}{\partial\boldsymbol{w}} = \boldsymbol{0}. \tag{12}$$

Consequently, the multipliers in $\boldsymbol{\Psi}$ are implicit functions of $\boldsymbol{y}$ and the multipliers in $\boldsymbol{\lambda}$.

Solving (10) and (11) at each optimization iteration, and, thus, eliminating them from the full-space KKT conditions, establishes an equivalence between (7) and (9),

$$\begin{aligned} \frac{d\mathcal{L}}{d\boldsymbol{y}} &= \frac{d\mathcal{J}}{d\boldsymbol{y}} + \boldsymbol{\lambda}^T \frac{d\mathcal{C}}{d\boldsymbol{y}} \\ &= \frac{\partial\mathcal{J}}{\partial\boldsymbol{y}} + \boldsymbol{\lambda}^T \frac{\partial\mathcal{C}}{\partial\boldsymbol{y}} + \boldsymbol{\Psi}^T \frac{\partial\mathcal{R}}{\partial\boldsymbol{y}} = \boldsymbol{0}, \\ \frac{d\mathcal{L}}{d\boldsymbol{\lambda}} &= \mathcal{C} = \boldsymbol{0}, \end{aligned}$$

which reveals the role of $\boldsymbol{\Psi}$ in assembling the total derivatives in the reduced-space. We will henceforth refer to $\boldsymbol{\Psi}$ as the first-order adjoint, and (12) as the first-order adjoint equation.

With the first-order sensitivity analysis completed, we apply Newton's method to the nonlinear system of equations in (7) to produce the KKT system,

$$\begin{bmatrix} \mathsf{H} & \mathsf{A}^T \\ \mathsf{A} & 0 \end{bmatrix} \begin{pmatrix} \Delta\boldsymbol{y} \\ \Delta\boldsymbol{\lambda} \end{pmatrix} = -\begin{pmatrix} \frac{d\mathcal{L}}{d\boldsymbol{y}} \\ \frac{d\mathcal{L}}{d\boldsymbol{\lambda}} \end{pmatrix}, \tag{13}$$

where $\mathsf{H} = d^2\mathcal{L}/d\boldsymbol{y}^2$ and $\mathsf{A} = d\mathcal{C}/d\boldsymbol{y}$ are the Hessian of the Lagrangian and the total constraint Jacobian blocks, respectively, that together form the KKT matrix in (13).

Note that each row in the constraint Jacobian $\mathsf{A}$ in the reduced-space is a total derivative. The conventional approach taken in gradient-based optimization algorithms requires this matrix to be provided explicitly, so it can be factored to determine a basis for its nullspace. However, computing $\mathsf{A}$ explicitly for the IDF formulation requires a separate adjoint solution for each nonlinear coupling constraint, which is intractable in general.

Few reduced-space algorithms address the challenge presented by the IDF Jacaobian, and state-based constraint Jacobians more generally. One notable exception is the matrix-free augmented Lagrangian algorithm developed by Arreckx et al. (2016); however, their approach relies on quasi-Newton approximations. In contrast, there have

been significant efforts in the full-space optimization literature to develop matrix-free Newton-Krylov algorithms that avoid explicit constraint Jacobians (Byrd et al. 2008, 2010; Heinkenschloss and Ridzal 2008; Heinkenschloss and Ridzal 2014).

While the desire for a matrix-free full-space algorithm is similar to that found in the reduced-space, the motivation is different. Usually, in the full-space formulation the constraint Jacobian is not a total derivative and can be computed explicitly (see below). However, the state equations are imposed as optimization constraints, which leads to a dimensionally large constraint Jacobian that cannot be factored directly to find its nullspace. Full-space matrix-free Newton-Krylov algorithms address this by solving the full-space KKT system (or the associated augmented systems) system inexactly at each Newton iteration using a Krylov solver.

Krylov iterative methods require a means of computing matrix-vector products and an effective preconditioner. In the full-space formulation, the constraint Jacobian is a PDE Jacobian, which is a partial derivative. Products with this Jacobian are readily available, for example, through algorithmic differentiation. The reduced-space constraint Jacobian, on the other hand, is a total derivative and requires additional considerations. Furthermore, the matrix-free nature of this strategy poses some challenges for preconditioning. Traditional preconditioning techniques based on explicit matrices, such as incomplete factorizations, are not applicable. Practitioners of full-space methods have developed effective preconditioners tailored to their applications. However, to the best of our knowledge, no such preconditioner exists in the literature for the primal-dual system in the IDF formulation.

In summary, in order to make the inexact-Newton-Krylov approach practical for the IDF formulation, we need an efficient means of computing the KKT-matrix-vector products, and an effective preconditioner for the IDF problem.

### 3.2 KKT-matrix-vector products using second-order adjoints

Given that the KKT matrix in (13) is the Jacobian of the KKT conditions, we can approximate products with this matrix and an arbitrary vector $z^T = [z_y^T, z_\lambda^T]$ via forward-differencing on (7), that is

$$\begin{aligned} \mathsf{K}z &= \begin{bmatrix} \mathsf{H} & \mathsf{A}^T \\ \mathsf{A} & 0 \end{bmatrix} \begin{pmatrix} z_y \\ z_\lambda \end{pmatrix} \\ &\approx \frac{1}{\epsilon} \begin{pmatrix} \mathcal{G}(y + \epsilon z_y, \lambda + \epsilon z_\lambda) - \mathcal{G}(y, \lambda) \\ \mathcal{C}(y + \epsilon z_y) - \mathcal{C}(y) \end{pmatrix}, \end{aligned} \tag{14}$$

where $\mathcal{G} = d\mathcal{L}/dy$ is the total derivative of the lagrangian with respect to the design variables $y$, and $\epsilon$ is the finite-difference step size.

In Jacobian-free Newton-Krylov methods that are used to solve nonlinear PDEs (Knoll and Keyes 2004), the PDE residual explicitly depends on the state variables that are being perturbed in the forward-difference approximation. This allows PDE-Jacobian-vector products to be approximated relatively cheaply. In contrast, in reduced-space PDE-governed optimization, the state variables are implicit functions of the design; therefore, in a naive implementation of the forward-difference approximation (14), we must solve the nonlinear state equations and the first-order adjoint at the perturbed point every time we evaluate a KKT-matrix-vector product. As we will demonstrate below, the linear adjoint solution is unavoidable, but the nonlinear state solution can be replaced with a linear problem.

Let $w + \epsilon\sigma$ be the state corresponding to the perturbed design point $y + \epsilon z_y$ such that,

$$\mathcal{R}(y + \epsilon z_y, w + \epsilon\sigma) = 0.$$

For an infinitesimal perturbation $\epsilon$, we can use a first-order Taylor expansion to arrive at,

$$\mathcal{R}(y+\epsilon z_y, w+\epsilon\sigma) = \mathcal{R}(y, w)+\epsilon \frac{\partial \mathcal{R}}{\partial y} z_y+\epsilon \frac{\partial \mathcal{R}}{\partial w} \sigma = 0. \tag{15}$$

Recognizing that $\mathcal{R}(y, w) = 0$, we can simplify (15) into the linear system,

$$\frac{\partial \mathcal{R}}{\partial w} \sigma = - \frac{\partial \mathcal{R}}{\partial y} z_y, \tag{16}$$

where $\sigma$ is a second-order adjoint (Wang et al. 1992), so called because it helps us assemble second-derivative information.

We repeat this process with the first-order adjoint residual in (12). Let $\Psi+\epsilon\Phi$ be the first-order adjoint corresponding to the perturbed primal-dual point $(y + \epsilon z_y, \lambda + \epsilon z_\lambda)$ such that,

$$\begin{aligned} &\mathcal{S}(y + \epsilon z_y, \lambda + \epsilon z_\lambda, w + \epsilon\sigma, \Psi + \epsilon\Phi) \\ &= \mathcal{S}(y, \lambda, w, \Psi) + \epsilon \frac{\partial \mathcal{S}}{\partial y} z_y + \epsilon \frac{\partial \mathcal{S}}{\partial \lambda} z_\lambda \\ &\quad +\epsilon \frac{\partial \mathcal{S}}{\partial w} \sigma + \epsilon \frac{\partial \mathcal{S}}{\partial \Psi} \Phi = 0. \end{aligned} \tag{17}$$

Similar to the primal residual, we recognize that the adjoint residual, $\mathcal{S}(y, \lambda, w, \Phi)$, is always zero and eliminate it from (17). Additionally, we note that the first-derivatives of $\mathcal{S}$ with respect to $\lambda$ and $\Psi$ are given by

$$\frac{\partial \mathcal{S}}{\partial \lambda} = \left( \frac{\partial \mathcal{C}}{\partial w} \right)^T \quad \text{and} \quad \frac{\partial \mathcal{S}}{\partial \Psi} = \left( \frac{\partial \mathcal{R}}{\partial w} \right)^T.$$

These modifications simplify (17) into the linear system,

$$\left(\frac{\partial \mathcal{R}}{\partial \boldsymbol{w}}\right)^T \boldsymbol{\Phi} = -\left(\frac{\partial \mathcal{C}}{\partial \boldsymbol{w}}\right)^T z_{\boldsymbol{\lambda}} - \begin{bmatrix} \frac{\partial \mathcal{S}}{\partial \boldsymbol{y}} & \frac{\partial \mathcal{S}}{\partial \boldsymbol{w}} \end{bmatrix} \begin{pmatrix} z_{\boldsymbol{y}} \\ \boldsymbol{\sigma} \end{pmatrix}, \tag{18}$$

where $\boldsymbol{\Phi}$ is another second-order adjoint used in assembling the KKT-matrix-vector product.

However, the first-derivatives with respect to $\boldsymbol{y}$ and $\boldsymbol{w}$ in (18) involve the second-derivatives of the objective function, the constraints, and the state equations. For a general-purpose algorithm, we cannot assume that the underlying PDE solvers possess the infrastructure needed to compute these second-derivatives directly. Instead, we define the second term on the right-side of (18) in terms of known first-derivatives. To do so, we make use of the fact that this term is in the form of a Jacobian-vector product, and evaluate it using forward-differencing on (12):

$$\begin{bmatrix} \frac{\partial \mathcal{S}}{\partial \boldsymbol{y}} & \frac{\partial \mathcal{S}}{\partial \boldsymbol{w}} \end{bmatrix} \begin{pmatrix} z_{\boldsymbol{y}} \\ \boldsymbol{\sigma} \end{pmatrix} \approx \frac{1}{\epsilon}\left[\, \boldsymbol{\mathcal{S}}(\boldsymbol{y} + \epsilon z_{\boldsymbol{y}}, \boldsymbol{\lambda}, \boldsymbol{w} + \epsilon \boldsymbol{\sigma}, \boldsymbol{\Psi}) - \boldsymbol{\mathcal{S}}(\boldsymbol{y}, \boldsymbol{\lambda}, \boldsymbol{w}, \boldsymbol{\Psi}) \,\right].$$

We have now described all the pieces necessary to compute (14) while avoiding a solution of the nonlinear state equations. To summarize, we begin the assembly with the primal component of the KKT conditions, namely

$$\begin{aligned} \mathsf{H} z_{\boldsymbol{y}} + \mathsf{A}^T z_{\boldsymbol{\lambda}} &\approx \frac{1}{\epsilon}\left[\, \boldsymbol{\mathcal{G}}(\boldsymbol{y} + \epsilon z_{\boldsymbol{y}}, \boldsymbol{\lambda} + \epsilon z_{\boldsymbol{\lambda}}) - \boldsymbol{\mathcal{G}}(\boldsymbol{y}, \boldsymbol{\lambda}) \,\right] \\ &\approx \frac{1}{\epsilon}\left[ \left(\frac{\partial \mathcal{J}}{\partial \boldsymbol{y}}\right)_{\epsilon} + (\boldsymbol{\lambda} + \epsilon z_{\boldsymbol{\lambda}})^T \left(\frac{\partial \mathcal{C}}{\partial \boldsymbol{y}}\right)_{\epsilon} \right. \\ &\qquad + (\boldsymbol{\Psi} + \epsilon \boldsymbol{\Phi})^T \left(\frac{\partial \mathcal{R}}{\partial \boldsymbol{y}}\right)_{\epsilon} \\ &\qquad \left. - \frac{\partial \mathcal{J}}{\partial \boldsymbol{y}} - \boldsymbol{\lambda}^T \frac{\partial \mathcal{C}}{\partial \boldsymbol{y}} - \boldsymbol{\Psi}^T \frac{\partial \mathcal{R}}{\partial \boldsymbol{y}} \right] \end{aligned}$$

where the $\epsilon$ subscript denotes first-derivatives that are evaluated at the perturbed design $\boldsymbol{y} + \epsilon z_{\boldsymbol{y}}$ and the corresponding state variables $\boldsymbol{w} + \epsilon \boldsymbol{\sigma}$. Finally, we perform a Taylor expansion on the dual component,

$$\begin{aligned} \mathsf{A} z_{\boldsymbol{y}} &\approx \frac{1}{\epsilon}\left[\, \boldsymbol{\mathcal{C}}(\boldsymbol{y} + \epsilon z_{\boldsymbol{y}}) - \boldsymbol{\mathcal{C}}(\boldsymbol{y}) \,\right] \\ &\approx \frac{1}{\epsilon}\left[ \boldsymbol{\mathcal{C}}(\boldsymbol{y}) + \epsilon \frac{\partial \mathcal{C}}{\partial \boldsymbol{y}} z_{\boldsymbol{y}} + \epsilon \frac{\partial \mathcal{C}}{\partial \boldsymbol{w}} \boldsymbol{\sigma} - \boldsymbol{\mathcal{C}}(\boldsymbol{y}) \right] \\ &\approx \frac{\partial \mathcal{C}}{\partial \boldsymbol{y}} z_{\boldsymbol{y}} + \frac{\partial \mathcal{C}}{\partial \boldsymbol{w}} \boldsymbol{\sigma}. \end{aligned}$$

With this second-order adjoint approach, we can compute KKT-matrix-vector products at a fixed cost of two linear system solutions based on the state Jacobian and transposed Jacobian, independent of the number of design variables and the number of constraints. This formulation extends to MDO problems without any changes, via the consolidated vectors in (5), where the two required linear system solutions are based on the Jacobian of the consolidated state equations. Another important observation is that the linear system solutions are performed at the $(\boldsymbol{y}, \boldsymbol{w})$ design-state point instead of the perturbed counterparts, so the PDE Jacobian(s) does not need to be reassembled nor preconditioner(s) refactored.

### 3.3 Preconditioning

It is well known that saddle-point systems like (13) are ill-conditioned; see, for example, the review by Benzi et al. (2005). Therefore, any Krylov iterative method used to solve (13) will require an effective preconditioner. In this section we show that a preconditioner for the IDF formulation can be constructed by approximately inverting the coupling-constraint Jacobian.

First, recall that a preconditioner is an operator that approximates the inverse of a target matrix and is relatively inexpensive to apply. In our case, for an arbitrary vector $\boldsymbol{b}$, we want an operator $\mathsf{M}^{-1}$ such that $\mathsf{M}^{-1}\boldsymbol{b} \approx \mathsf{K}^{-1}\boldsymbol{b}$.

To identify a preconditioner, it is instructive to begin with the ideal case, where $\mathsf{M} = \mathsf{K}$. In this case, given $\boldsymbol{b}^T = [\boldsymbol{b}_{\boldsymbol{x}}^T, \boldsymbol{b}_{\bar{\boldsymbol{w}}}^T, \boldsymbol{b}_{\boldsymbol{\lambda}}^T]$, we must solve the following linear system:

$$\begin{bmatrix} \mathsf{H}_{\boldsymbol{xx}} & \mathsf{H}_{\bar{\boldsymbol{w}}\boldsymbol{x}} & \mathsf{A}_{\boldsymbol{x}}^T \\ \mathsf{H}_{\boldsymbol{x}\bar{\boldsymbol{w}}} & \mathsf{H}_{\bar{\boldsymbol{w}}\bar{\boldsymbol{w}}} & \mathsf{A}_{\bar{\boldsymbol{w}}}^T \\ \mathsf{A}_{\boldsymbol{x}} & \mathsf{A}_{\bar{\boldsymbol{w}}} & 0 \end{bmatrix} \begin{pmatrix} \boldsymbol{p}_{\boldsymbol{x}} \\ \boldsymbol{p}_{\bar{\boldsymbol{w}}} \\ \boldsymbol{p}_{\boldsymbol{\lambda}} \end{pmatrix} = \begin{pmatrix} \boldsymbol{b}_{\boldsymbol{x}} \\ \boldsymbol{b}_{\bar{\boldsymbol{w}}} \\ \boldsymbol{b}_{\boldsymbol{\lambda}} \end{pmatrix}, \tag{19}$$

where the subscripts on the Hessian and constraint Jacobian blocks denote the variables by which the Lagrangian and the IDF constraints are (total) differentiated, respectively. In addition, we have introduced $\bar{\boldsymbol{w}} \in \mathbb{R}^l$ for the vector of IDF coupling variables; in the two-discipline case this vector would be $\bar{\boldsymbol{w}}^T = [\bar{\boldsymbol{u}}^T \bar{\boldsymbol{v}}^T]$.

For our first step toward the preconditioner, we drop all the Hessian terms from $\mathsf{K}$ except for $\mathsf{H}_{\boldsymbol{xx}} = \nabla^2_{\boldsymbol{xx}}\mathcal{L}$, which is replaced by a symmetric positive-definite approximation $\mathsf{B}$:

$$\begin{bmatrix} \mathsf{B} & 0 & \mathsf{A}_{\boldsymbol{x}}^T \\ 0 & 0 & \mathsf{A}_{\bar{\boldsymbol{w}}}^T \\ \mathsf{A}_{\boldsymbol{x}} & \mathsf{A}_{\bar{\boldsymbol{w}}} & 0 \end{bmatrix} \begin{pmatrix} \boldsymbol{p}_{\boldsymbol{x}} \\ \boldsymbol{p}_{\bar{\boldsymbol{w}}} \\ \boldsymbol{p}_{\boldsymbol{\lambda}} \end{pmatrix} = \begin{pmatrix} \boldsymbol{b}_{\boldsymbol{x}} \\ \boldsymbol{b}_{\bar{\boldsymbol{w}}} \\ \boldsymbol{b}_{\boldsymbol{\lambda}} \end{pmatrix}, \tag{20}$$

We will justify the above approximation in the sequel, when we connect the preconditioner with the MDF formulation.

Next, notice that the Jacobian $\mathsf{A}_{\bar{\boldsymbol{w}}} = d\boldsymbol{\mathcal{C}}/d\bar{\boldsymbol{w}}$, which we will call the IDF Jacobian, is a square matrix: indeed, as Theorem 1 below shows, the IDF Jacobian is not merely square but also nonsingular provided the MDF Jacobian and the discipline Jacobians are nonsingular.

**Theorem 1** *Let $\mathsf{A}_i = \partial \mathcal{R}_i / \partial \boldsymbol{w}_i$ be the Jacobian of the $i$th discipline's residual respect to its state variables. If $\mathsf{A}_i$ is nonsingular $\forall i$, then the IDF Jacobian $\mathsf{A}_{\bar{\boldsymbol{w}}} = d\boldsymbol{\mathcal{C}}/d\bar{\boldsymbol{w}}$ is*

*nonsingular if and only if the MDA Jacobian* $\mathsf{A}_{\boldsymbol{w}} = \partial \boldsymbol{\mathcal{R}}/\partial \boldsymbol{w}$ *is nonsingular.*

The proof of Theorem 1 is a straightforward application of the Schur complement; the details can be found in Appendix. The significance of the Theorem is that it guarantees that $\mathsf{A}_{\bar{\boldsymbol{w}}}$ is invertible whenever the linearized MDA itself is solvable.

The invertibility of $\mathsf{A}_{\bar{\boldsymbol{w}}}$ allows us to perform a block factorization of the matrix in (20). Applying this factorization, we arrive at the initial version of our preconditioner:

1. Solve $\mathsf{A}_{\bar{\boldsymbol{w}}}^T \boldsymbol{p}_{\boldsymbol{\lambda}} = \boldsymbol{b}_{\bar{\boldsymbol{w}}}$ for $\boldsymbol{p}_{\boldsymbol{\lambda}}$.
2. Solve $\mathsf{B}\boldsymbol{p}_{\boldsymbol{x}} = \boldsymbol{b}_{\boldsymbol{x}} - \mathsf{A}_{\boldsymbol{x}}^T \boldsymbol{p}_{\boldsymbol{\lambda}}$ for $\boldsymbol{p}_{\boldsymbol{x}}$.
3. Solve $\mathsf{A}_{\bar{\boldsymbol{w}}} \boldsymbol{p}_{\bar{\boldsymbol{w}}} = \boldsymbol{b}_{\boldsymbol{\lambda}} - \mathsf{A}_{\boldsymbol{x}} \boldsymbol{p}_{\boldsymbol{x}}$ for $\boldsymbol{p}_{\bar{\boldsymbol{w}}}$.

In Steps 1 and 3 above, we must solve systems with the IDF Jacobian and its transpose. Recall that the rows of $\mathsf{A}_{\bar{\boldsymbol{w}}}$ consist of total derivatives, so forming this matrix is not feasible in practice. Therefore, to solve the linear systems in Steps 1 and 3, we again turn to Krylov iterative methods.

For simplicity, consider the system in Step 3 above and a two-discipline problem. A Krylov method used to solve this system requires products between $\mathsf{A}_{\bar{\boldsymbol{w}}}$ and an arbitrary vector $\bar{\boldsymbol{z}}^T = [\bar{\boldsymbol{z}}_{\boldsymbol{u}}^T, \bar{\boldsymbol{z}}_{\boldsymbol{v}}^T]$. Such a product can be expressed as (recall $\mathsf{A}_{\bar{\boldsymbol{w}}} = d\boldsymbol{\mathcal{C}}/d\bar{\boldsymbol{w}}$ where $\boldsymbol{\mathcal{C}}$ is defined by (5))

$$\begin{aligned}\mathsf{A}_{\bar{\boldsymbol{w}}}\bar{\boldsymbol{z}} &= \begin{bmatrix} -\mathsf{I} & \frac{\partial \boldsymbol{\mathcal{E}}_{\boldsymbol{v}}}{\partial \boldsymbol{u}} \frac{d\boldsymbol{u}}{d\bar{\boldsymbol{v}}} \\ \frac{\partial \boldsymbol{\mathcal{E}}_{\boldsymbol{u}}}{\partial \boldsymbol{v}} \frac{d\boldsymbol{v}}{d\bar{\boldsymbol{u}}} & -\mathsf{I} \end{bmatrix} \begin{pmatrix} \bar{\boldsymbol{z}}_{\boldsymbol{u}} \\ \bar{\boldsymbol{z}}_{\boldsymbol{v}} \end{pmatrix} \\ &= \begin{bmatrix} -\bar{\boldsymbol{z}}_{\boldsymbol{u}} - \frac{\partial \boldsymbol{\mathcal{E}}_{\boldsymbol{v}}}{\partial \boldsymbol{u}} \left( \frac{\partial \boldsymbol{\mathcal{R}}_{\boldsymbol{u}}}{\partial \boldsymbol{u}} \right)^{-1} \frac{\partial \boldsymbol{\mathcal{R}}_{\boldsymbol{u}}}{\partial \bar{\boldsymbol{v}}} \bar{\boldsymbol{z}}_{\boldsymbol{v}} \\ -\bar{\boldsymbol{z}}_{\boldsymbol{v}} - \frac{\partial \boldsymbol{\mathcal{E}}_{\boldsymbol{u}}}{\partial \boldsymbol{v}} \left( \frac{\partial \boldsymbol{\mathcal{R}}_{\boldsymbol{v}}}{\partial \boldsymbol{v}} \right)^{-1} \frac{\partial \boldsymbol{\mathcal{R}}_{\boldsymbol{v}}}{\partial \bar{\boldsymbol{w}}} \bar{\boldsymbol{z}}_{\boldsymbol{w}} \end{bmatrix}, \end{aligned}$$

where we have replaced the total derivatives $d\boldsymbol{u}/d\bar{\boldsymbol{v}}$ and $d\boldsymbol{v}/d\bar{\boldsymbol{u}}$ with their definitions (see the proof in Appendix, which reviews these definitions). The above expression shows that each product with $\mathsf{A}_{\bar{\boldsymbol{w}}}$ requires the inversion of the discipline Jacobians. While these inversions are decoupled from one another, their cost is unnecessarily high given the other approximations inherent to the preconditioner (e.g. approximating and discarding the Hessian blocks).

Instead, we replace the inverse Jacobians with approximate inversions induced by the discipline preconditioners; that is, we assume that each discipline has a preconditioner it uses to solve its PDE system, and that we can apply these preconditioners to arbitrary vectors. Thus, the products with $\mathsf{A}_{\bar{\boldsymbol{w}}}$ are approximated as

$$\tilde{\mathsf{A}}_{\bar{\boldsymbol{w}}}\bar{\boldsymbol{z}} = \begin{bmatrix} -\bar{\boldsymbol{z}}_{\boldsymbol{u}} - \frac{\partial \boldsymbol{\mathcal{E}}_{\boldsymbol{v}}}{\partial \boldsymbol{u}} \left( \widetilde{\frac{\partial \boldsymbol{\mathcal{R}}_{\boldsymbol{u}}}{\partial \boldsymbol{u}}} \right)^{-1} \frac{\partial \boldsymbol{\mathcal{R}}_{\boldsymbol{u}}}{\partial \bar{\boldsymbol{v}}} \bar{\boldsymbol{z}}_{\boldsymbol{v}} \\ -\bar{\boldsymbol{z}}_{\boldsymbol{v}} - \frac{\partial \boldsymbol{\mathcal{E}}_{\boldsymbol{u}}}{\partial \boldsymbol{v}} \left( \widetilde{\frac{\partial \boldsymbol{\mathcal{R}}_{\boldsymbol{v}}}{\partial \boldsymbol{v}}} \right)^{-1} \frac{\partial \boldsymbol{\mathcal{R}}_{\boldsymbol{v}}}{\partial \bar{\boldsymbol{w}}} \bar{\boldsymbol{z}}_{\boldsymbol{w}} \end{bmatrix}, \tag{21}$$

where the $\tilde{\cdot}$ accents denote approximate matrices. We note that stationary PDE preconditioners are necessary in this context, otherwise the products with $\tilde{\mathsf{A}}_{\bar{\boldsymbol{w}}}$ will change from one iteration to the next and may cause the Krylov solver to fail.

A similar approach to the one described above is used to form the products with $\tilde{\mathsf{A}}_{\bar{\boldsymbol{w}}}^T$, which are needed by Krylov iterative methods to solve the system in Step 1 of the IDF preconditioner. In addition, this general approach is used to form products with $\tilde{\mathsf{A}}_{\boldsymbol{x}}^T$ and $\tilde{\mathsf{A}}_{\boldsymbol{x}}$, which are needed in steps 2 and 3, respectively. For the transposed matrices, the only significant difference is that these products rely on the transpose of the PDE preconditioners.

To solve the systems involving $\tilde{\mathsf{A}}_{\bar{\boldsymbol{w}}}^T$ and $\tilde{\mathsf{A}}_{\bar{\boldsymbol{w}}}$ in Steps 1 and 3, we use unpreconditioned GMRES; unlike the KKT matrix in (13), numerical experiments suggest the IDF Jacobian is well conditioned and preconditioning is not necessary. Indeed, in the numerical experiments in Section 4, we find that approximately 5 iterations are typically needed to reduce the relative residual by a factor of $10^{-2}$.

In summary, the preconditioner is given by

1. Inexactly solve $\tilde{\mathsf{A}}_{\bar{\boldsymbol{w}}}^T \boldsymbol{p}_{\boldsymbol{\lambda}} = \boldsymbol{b}_{\bar{\boldsymbol{w}}}$ for $\boldsymbol{p}_{\boldsymbol{\lambda}}$ using GMRES, where the necessary matrix-vector products with $\tilde{\mathsf{A}}_{\bar{\boldsymbol{w}}}^T$ are evaluated using the transposed variant of (21).
2. Solve $\mathsf{B}\boldsymbol{p}_{\boldsymbol{x}} = \boldsymbol{b}_{\boldsymbol{x}} - \tilde{\mathsf{A}}_{\boldsymbol{x}}^T \boldsymbol{p}_{\boldsymbol{\lambda}}$ for $\boldsymbol{p}_{\boldsymbol{x}}$, where the product $\tilde{\mathsf{A}}_{\boldsymbol{x}}^T \boldsymbol{p}_{\boldsymbol{\lambda}}$ is evaluated using the $\tilde{\mathsf{A}}_{\boldsymbol{x}}^T$ variant of (21).
3. Solve $\tilde{\mathsf{A}}_{\bar{\boldsymbol{w}}} \boldsymbol{p}_{\bar{\boldsymbol{w}}} = \boldsymbol{b}_{\boldsymbol{\lambda}} - \tilde{\mathsf{A}}_{\boldsymbol{x}} \boldsymbol{p}_{\boldsymbol{x}}$ for $\boldsymbol{p}_{\bar{\boldsymbol{w}}}$ using GMRES, where the matrix-vector products with $\tilde{\mathsf{A}}_{\bar{\boldsymbol{w}}}$ are evaluated using (21), and $\tilde{\mathsf{A}}_{\boldsymbol{x}} \boldsymbol{p}_{\boldsymbol{x}}$ is evaluated using the $\tilde{\mathsf{A}}_{\boldsymbol{x}}$ variant of (21).

The remaining unspecified detail of the IDF preconditioner is the approximate Hessian $\mathsf{B}$ appearing in Step 2. For this work we simply take $\mathsf{B} = \mathsf{I}$. More generally, a quasi-Newton approximation of the Hessian could be used for $\mathsf{B}$, but we did not find any significant improvement using, e.g., BFGS in our experiments.

We conclude this section with some remarks regarding the IDF preconditioner.

*Remark 2* The IDF preconditioner was inspired by the $\tilde{\mathsf{P}}_2$ preconditioner proposed by Biros and Ghattas for full-space PDE-constrained optimization (Biros and Ghattas 2005a). In the full-space context, the system matrices in Steps 3 and 1 are the PDE Jacobian and its transpose, respectively; thus, rather than using GMRES to solve these systems, they use the PDE preconditioner directly.

*Remark 3* Many modern open-source PDE solvers provide, or can be made to provide, the subroutines necessary for the IDF preconditioner, e.g. Trilinos (Heroux et al. 2005). For some legacy (linear) structural codes, it may be possible to use the exact stiffness matrix to compute products with $\tilde{\mathsf{A}}_{\bar{\boldsymbol{w}}}$ without significant cost. More generally, we acknowledge that the operations required by the IDF preconditioner may prevent its use with some legacy solvers.

*Remark 4* To see how the IDF preconditioner mimics MDF, first recognize that IDF acts like static condensation on the coupling variables for a linear MDA. Therefore, solutions with the IDF Jacobian, $\mathsf{A}_{\bar{\boldsymbol{w}}} = d\mathcal{C}/d\bar{\boldsymbol{w}}$, can be used to recover solutions for the full-space MDA. For this reason, Steps 1 and 3 in the IDF preconditioner are essentially equivalent to solving the MDF adjoint and (linearized) state, respectively. For Step 2, the vector $\boldsymbol{b_x} - \mathsf{A}_{\boldsymbol{x}}^T \boldsymbol{p_\lambda}$ is analogous to the total gradient, with $\boldsymbol{b_x}$ playing the role of $\partial\mathcal{J}/\partial x$ and the second term playing the role of $\boldsymbol{\Psi}^T \partial\boldsymbol{\mathcal{R}}/\partial x$. Thus, in the MDF context, Step 2 finds the step in design space, where $\mathsf{B}$ would typically be some quasi-Newton approximation to the reduced Hessian.

### 3.4 Algorithm summary

We present an overview of our reduced-space inexact-Newton-Krylov implementation in Algorithm 1. In line 9, the matrix-free KKT-matrix-vector product defined in Section 3.2 is used in conjunction with FLECS (Hicken and Dener 2015), a Krylov solver developed by the authors for nonconvex saddle-point problems. FLECS addresses nonconvexity by using a trust-radius constraint, and utilizes an augmented Lagrangian quadratic model with the penalty parameter $\sigma$. The update to this penalty term in line 7 differs slightly from the original: inclusion of $\|\boldsymbol{\mathcal{G}}_k\|$ in the denominator causes $\sigma$ to grow rapidly when optimality out paces feasibility, which has the effect of enforcing the constraints more strongly in the FLECS solution. Additionally, the Krylov tolerance $\eta$ in line 8 is dynamically adjusted to obtain superlinear convergence and avoid oversolving (Eisenstat and Walker 1996).

The Newton step $\boldsymbol{s}_k$ is globalized in a trust-region framework (see, e.g. Conn et al. 2000) using a basic filter (Fletcher and Leyffer 2002). The globalization features a second-order correction in line 16 to address the Maratos effect, and an efficient re-solution of (13) in line 21 that recycles the subspace built up during the original Newton step

**Algorithm 1** Reduced-space inexact-Newton-Krylov with filter globalization

**Data**: $\boldsymbol{y}_0, \boldsymbol{\lambda}_0, \sigma_0, \eta_0, \Delta_0, \Delta_{\max}, \tau_p, \tau_d$
**Result**: estimate for the optimal solution $\boldsymbol{y}^*$

1 set $\sigma = \sigma_0$, $\eta = \eta_0$, $\Delta = \Delta_0$
2 **for** $k = 0, 1, 2, \ldots,$ *max_iter* **do** `// start Newton loop`
3 compute KKT conditions $\nabla\mathcal{L}_k = (\boldsymbol{\mathcal{G}}_k^T, \boldsymbol{\mathcal{C}}_k^T)^T$
4 **if** $\|\boldsymbol{\mathcal{G}}_k\| \le \tau_p\|\boldsymbol{\mathcal{G}}_0\|$ *and* $\|\boldsymbol{\mathcal{C}}_k\| \le \tau_d\|\boldsymbol{\mathcal{C}}_0\|$ **then** `// check convergence`
5 set $\boldsymbol{y}^* = \boldsymbol{y}_k$ and return
6 **end**
7 set $\sigma \leftarrow \max\left[\sigma,\ \sigma_0\|\boldsymbol{\mathcal{C}}_0\|/\min\left(\|\boldsymbol{\mathcal{G}}_k\|,\ \|\boldsymbol{\mathcal{C}}_k\|\right)\right]$
8 set $\eta \leftarrow \max\left[\eta,\ \min\left(1.0,\ \sqrt{\nabla\mathcal{L}_k/\nabla\mathcal{L}_0}\right),\ \min\left(\tau_p/\|\boldsymbol{\mathcal{G}}_k\|,\ \tau_d/\|\boldsymbol{\mathcal{C}}_k\|\right)\right]$
9 solve (13) for $\boldsymbol{s}_k = (\Delta\boldsymbol{y}_k, \Delta\boldsymbol{\lambda}_k)^T$ using FLECS, with tolerance $\eta$, penalty $\sigma$, and trust-radius $\Delta$
10 **for** $i = 0, 1, 2, \ldots,$ *max_filter_iter* **do** `// start filter loop`
11 **if** $\left[\mathcal{J}(\boldsymbol{y}_k + \Delta\boldsymbol{y}_k),\ \|\boldsymbol{\mathcal{C}}(\boldsymbol{y}_k + \Delta\boldsymbol{y}_k)\|\right]$ *is not dominated by filter* **then** `// step accepted by filter`
12 **if** $i = 0$ *and* $\|\Delta\boldsymbol{y}_k\| = \Delta$ **then** set $\Delta \leftarrow \min(2\Delta, \Delta_{\max})$
13 set filter_success = true, and exit filter loop
14 **else** `// step rejected by filter`
15 **if** $i = 0$ **then**
16 compute second-order correction $\Delta\boldsymbol{y}_c$
17 **if** $\left[\mathcal{J}(\boldsymbol{y}_k + \Delta\boldsymbol{y}_k + \Delta\boldsymbol{y}_c),\ \|\boldsymbol{\mathcal{C}}(\boldsymbol{y}_k + \Delta\boldsymbol{y}_k + \Delta\boldsymbol{y}_c)\|\right]$ *is not dominated by filter* **then**
18 set filter_success = true, set $\Delta\boldsymbol{y}_k \leftarrow \Delta\boldsymbol{y}_k + \Delta\boldsymbol{y}_c$, and exit filter loop
19 **end**
20 **end**
21 set $\Delta \leftarrow \frac{1}{4}\Delta$, and re-solve (13) for $\boldsymbol{s}_k$ using FLECS, with penalty $\sigma$ and trust-radius $\Delta$
22 **end**
23 **end**
24 **if** *filter_success* **then** set $\boldsymbol{y}_{k+1} = \boldsymbol{y}_k + \Delta\boldsymbol{y}_k$ and $\boldsymbol{\lambda}_{k+1} = \boldsymbol{\lambda}_k + \Delta\boldsymbol{\lambda}_k$
25 **else** set $\boldsymbol{y}_{k+1} = \boldsymbol{y}_k$ and $\boldsymbol{\lambda}_{k+1} = \boldsymbol{\lambda}_k$
26 **end**

solution. We refer the reader to Hicken and Dener (2015) for a comprehensive description of these methods.

Note that Algorithm 1 is formulated for equality-constrained optimization. Its counterpart for unconstrained problems is the trust-region Newton-CG algorithm (Wright and Nocedal 2006), which is an inexact-Newton-Krylov method where the Krylov solver of choice is the Steihaug-Toint Conjugate Gradient method (Steihaug 1983). We apply this latter algorithm to the unconstrained MDF problems in Section 4, in conjunction with matrix-free Hessian-vector products formulated using a second-order adjoint approach analogous to the one described in Section 3.2 for the IDF products.

## 4 Numerical experiments

### 4.1 Domain decomposition as a model MDO problem

Our first numerical example is an inverse problem based on the two-dimensional Laplace equation. The optimization problem is to determine the boundary value on the left edge that produces a target solution along the right edge. To mimic a multidisciplinary system, we use a domain decomposition approach in which the Laplace equation is solved independently on each subdomain. These independent subdomains are intended to mimic different "disciplines" in an MDO problem and are coupled together at the boundaries using consistency constraints imposed as Dirichlet conditions. Using this model MDO problem, we can investigate the IDF formulation with varying numbers of "disciplines".

The 2-dimensional Laplace equation on a unit domain is given by

$$\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} = 0, \quad \forall\, (x, y) \in \Omega = [0, 1]^2\,. \tag{22}$$

The boundary conditions for this test case are given by

$$\begin{aligned} u(x, y) &= 0, \quad \forall\, (x, y) \in \left(\partial\Omega|_{top} \cup \partial\Omega|_{bottom}\right), \\ u(0, y) &= d(y), \quad \forall\, y \in \partial\Omega|_{left}, \\ \frac{\partial u}{\partial x}\Big|_{x=1} &= 0, \quad \forall\, y \in \partial\Omega|_{right}, \end{aligned} \tag{23}$$

where the Dirichlet boundary on the left is defined by the control $d(y)$. The Laplace equation is discretized using a second-order accurate finite-difference scheme over a uniform grid, and, based on this discretization, we use $\boldsymbol{d}_i = d(y_i)$ to denote the $i$th design variable.

The unit domain with the boundary conditions specified in (23) is divided into equally sized subdomains, for example, as shown by the four-subdomain example in Fig. 1. As mentioned above, each subdomain is intended to mimic an individual "discipline" in a multidisciplinary problem. Decomposing the global domain creates new inter-domain

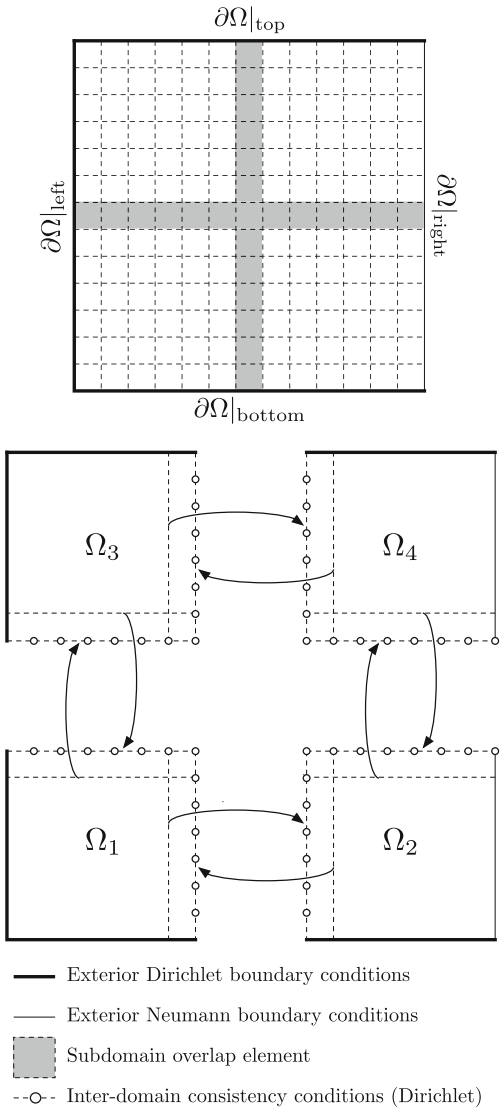



**Fig. 1** Four-subdomain example of the domain decomposition used for the Laplace problem

boundaries on which we must define boundary conditions, and it is through these conditions that we couple the "disciplines" together. Consequently, the definition of the boundary conditions depends on the MDAO formulation.

For the MDF architecture, the inter-domain boundary conditions are derived using the overlapping nodes shown in Fig. 1. The solution values of each domain that are one interval away from the edge are mapped onto the neighboring subdomain as a Dirichlet condition. An example expression of the inter-domain bounds is given as

$$\begin{aligned} u_1(x_1, y_{1,max}) &= u_3(x_1, y_{3,min} + \Delta y), \\ u_1(x_{1,max}, y_1) &= u_2(x_{2,min} + \Delta x, y_1), \end{aligned}$$

where $\Delta x$ and $\Delta y$ are the node spacing of the uniform grid, and the subscripts denote the domain numbers in the four-subdomain example from Fig. 1. In this instance, the row

of nodes second from the bottom in $\Omega_3$ has been mapped onto the top edge of $\Omega_1$ as a Dirichlet condition. Similarly, the column of nodes second from the left in $\Omega_2$ has been mapped onto the right edge of $\Omega_1$. Together with the exterior boundary conditions defined in (23), $\Omega_1$ now has all the information it needs for a well-posed solution of the Laplace equation. Again, these internal boundary conditions are intended to mimic the transfer of information between the coupled disciplines of a multidisciplinary system.

The multidisciplinary analysis (MDA) for the MDF architecture is then solved using a block-Jacobi approach. At each iteration of the MDA, the individual subdomains are solved independently, and the inter-domain Dirichlet conditions are updated before advancing to the next iteration. The multidisciplinary solution is considered converged when the maximum difference between overlapping nodes of neighboring subdomains falls below a specified tolerance.

*Remark 5* Stationary iterative methods, such as block-Jacobi, remain a popular solution method for multidisciplinary analysis (Bloebaum 1995; Kodiyalam and Sobieszczanski-Sobieski 2001; Martins and Lambe 2013) due to their ease of implementation when using disparate solvers. This choice is particularly appropriate for the present work, because MDF problems with block-Jacobi MDAs can be reformulated into IDF problems without necessitating any new solver infrastructure, and the two MDO architectures can be compared on equal footing in terms of implementation difficulty. We recognize that a monolithic Newton-Krylov approach can yield a more efficient solution of the multidisciplinary system (Kennedy and Martins 2010), and we will explore this alternative further with our second test problem in Section 4.2.

In the IDF architecture, the inter-domain boundary conditions have the same form as MDF, but the values are prescribed by coupling variables; for example,

$$u_1(x_1, y_{1,max}) = \bar{u}_{1,top}(x_1),$$
$$u_1(x_{1,max}, y_1) = \bar{u}_{1,right}(y_1),$$

where the IDF coupling variables, denoted by $\bar{u}$, are part of the optimization variables in the optimization problem. No multidisciplinary analysis is performed in the IDF formulation, and, instead, the multidisciplinary feasibility of the optimal solution is enforced through optimization constraints of the form

$$u_3(x_1, y_{3,min} + \Delta y) - \bar{u}_{1,top}(x_1) = 0,$$
$$u_2(x_{2,min} + \Delta x, y_1) - \bar{u}_{1,right}(y_1) = 0.$$

Consequently, the subdomains in the IDF formulation do not communicate any boundary information to each other during the solution of the state equations, nor do they require communication during the adjoint solution.

#### *4.1.1 Objective function and problem setup*

The objective function is the squared $L^2$ error between a target solution and the solution of the Laplace equation on the right edge of the full domain:

$$\mathcal{J} = \frac{1}{2}\int_0^1 (u(1, y) - u_{\text{targ}}(y))^2 dy$$

where $u_{\text{targ}}$ is the target solution along $\partial\Omega|_{\text{right}}$. Thus, the optimization seeks to recover the target solution through the control of the variables that define the Dirichlet boundary condition along the left edge. The target solution is the parabola $u_{\text{targ}}(y) = -4y(y-1)$, which has a maxima of 1 at $y = 0.5$ and complies with zero Dirichlet boundary conditions at $y = 0$ and $y = 1$, to ensure the solution is continuous.

Table 1 lists all the domain-decomposition cases evaluated in the computational cost comparison. The variables $n_x$ and $n_y$ denote the number of subdomains along the $x$ and $y$ axes, respectively. Since the design variables are defined directly on the left-edge nodes, the number of design variables in the optimization problem changes with $n_y$. In the IDF formulation, the full size of the design space is expanded further by the coupling variables. The cases are designed such that the subdomain grid sizes are fixed at $(15 \times 15)$, including the overlapping nodes, for a total of 225 state variables for each "discipline". This ensures that the computational cost of an individual subdomain solution is a fixed constant for both architectures in all domain partitioning configurations, and the number of subdomain solutions can be used as a cost metric for optimization runs.

Table 2 lists the algorithm parameters used for the solutions generated in this section. The initial and maximum trust radii vary for each test case to account for the differences in the size of the design space. The maximum radius is set as $\Delta_{\max} = \sqrt{0.25\, n_d}$ and the initial radius as $\Delta_0 = \Delta_{\max}/8$ for both formulations, where $n_d$ is the

**Table 1** Subdomain configurations tested with the Laplace problem

| Design vars | Domains | Partition ($n_x \times n_y$) | State vars | Coupling vars |
|---|---|---|---|---|
| 26 | 2 | $1 \times 2$ | 450 | 28 |
| 26 | 4 | $2 \times 2$ | 900 | 110 |
| 26 | 6 | $3 \times 2$ | 1350 | 192 |
| 39 | 6 | $2 \times 3$ | 1350 | 194 |
| 39 | 9 | $3 \times 3$ | 2025 | 332 |
| 39 | 12 | $4 \times 3$ | 2700 | 470 |

**Table 2** Optimization parameters for the Laplace problem

| Parameter | Value |
|---|---|
| Relative optimality tolerance, $\tau_p$ | $10^{-5}$ |
| Relative feasibility tolerance, $\tau_d$ | $10^{-5}$ |
| Initial penalty parameter, $\sigma_0$ | $10^5$ |
| Initial Krylov tolerance, $\eta_0$ | 0.5 |
| Krylov subspace size | 20 |

number of degrees of freedom for the optimization problem, including the coupling variables in the case of IDF.

#### *4.1.2 IDF preconditioner results*

In Section 3.3, we mentioned that the IDF preconditioner was critical in the practical application of the reduced-space Newton-Krylov algorithm to the IDF formulation. Without this preconditioner, the solution of the KKT system emerging from the IDF formulation stalls and fails to produce a useful search direction. To demonstrate this claim, Fig. 2 shows the convergence histories of the preconditioned and non-preconditioned FLECS solution of (13) in the four-subdomain ($2 \times 2$) test case. The residual norm being measured is that of the linearized KKT conditions, and the Krylov subspace is capped at 20 vectors. The preconditioned solver is able to meet the tolerance target using only 10 iterations, while the non-preconditioned solver terminates at the iteration limit without any significant progress.

For the sample shown in Fig. 2, the preconditioned Krylov solution required 246 subdomain solution versus 160 for its non-preconditioned counterpart. However, the majority of the subdomain solutions in the preconditioned case are approximate. Each Krylov iteration for both cases requires one KKT-matrix-vector product, which in turn requires 8 subdomain solutions in this four-subdomain example (i.e. : 4 solutions for each second-order adjoint). This accounts for only 80 subdomain solutions in the preconditioned case. The remaining 166 subdomain solutions are, in fact, cheap approximations used in Steps 1 and 3 of the IDF preconditioner. We believe that this is a small price well worth paying, as this KKT system cannot be adequately solved otherwise.

#### *4.1.3 Optimization results*

Figure 3 shows the optimum solution for the four-subdomain case, with the overlap between subdomains shaded in gray. To plotting accuracy, both the MDF and IDF formulations recover the same optimum, matching the targeted parabola on the right edge of the domain at $x = 1$.

In order to demonstrate the superlienar convergence expected of the Newton-Krylov algorithms, we plot the convergence history of the four-subdomain case in Fig. 4. The convergence metrics are the $l^2$-norms of the KKT conditions in (7), $\|d\mathcal{L}/d\boldsymbol{y}\|_2$ for optimality and $\|\boldsymbol{\mathcal{C}}\|_2$ for feasibility, normalized by their initial values. The trends we present here are representative of all the subdomain configurations tested for this problem. The IDF formulation exhibits clear superlinear convergence on this problem, while the MDF formulation is close to superlinear except for the numerical artifacts at the end. This behavior is due to the dynamic Krylov tolerance that attempts to prevent oversolving near the optimum.

Additionally, we observe that the optimization for both architectures also takes approximately the same number of nonlinear iterations, but the overall computational cost of the MDF formulation for this problem is an order of magnitude more expensive than IDF. This is due to the forward and adjoint solutions in the MDF formulation requiring expensive block-Jacobi iterations to achieve full multidisciplinary convergence.

The computational cost scaling against the number of subdomains is shown in Fig. 5. The trend-lines are separated

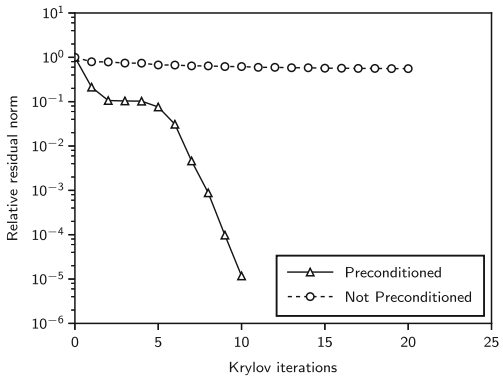



**Fig. 2** IDF KKT system Krylov solution convergence with four-subdomain ($2 \times 2$) decomposition

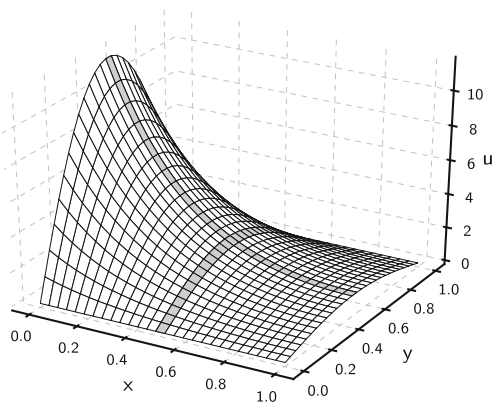



**Fig. 3** Optimal solution with a four-subdomain ($2 \times 2$) decomposition

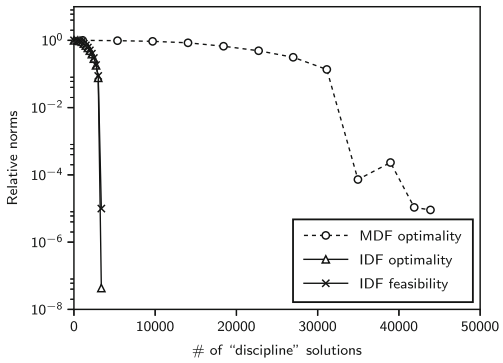



**Fig. 4** Convergence history for optimization with four-subdomain ($2 \times 2$) decomposition

per coupling architecture and number of design variables. It is clear that the MDF formulation is not only significantly more expensive on this problem, but it also scales poorly with increasing number of "disciplines".

### 4.2 Aero-structural optimization of a 2-D elastic nozzle

Our second test case is another inverse design problem, this time based on the quasi-one-dimensional flow in an elastic nozzle. The optimization problem is to find the (deformed) nozzle shape that recovers a prescribed pressure distribution. The flow is modeled using the quasi-one-dimensional Euler equations, and the structural deformations in the nozzle wall are modeled using combined beam-bending and axial stiffness.

#### *4.2.1 Quasi-1D flow solver*

The quasi-one-dimensional Euler equations are given by

$$\frac{\partial \mathcal{F}}{\partial x} - \mathcal{G} = 0, \quad \forall\, x \in [0, 1], \tag{24}$$

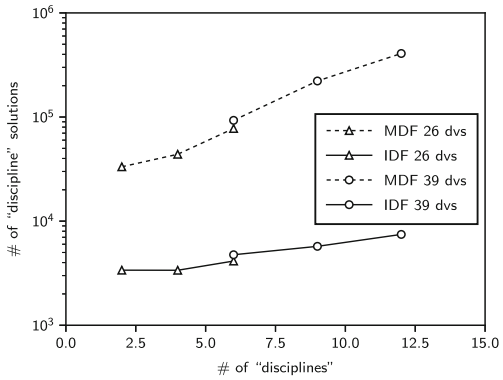



**Fig. 5** Cost scaling with number of subdomain "disciplines" under IDF and MDF architectures

where flux and source terms are defined by

$$\boldsymbol{F}^T = \left( \rho v A, (\rho v^2 + p)A, v(e + p)A \right)$$

and

$$\boldsymbol{\mathcal{G}}^T = \left( 0, p\tfrac{dA}{dx}, 0 \right),$$

respectively. The state variables in the Euler equations are the density, $\rho$, momentum per unit volume, $\rho v$, and energy per unit volume, $e$. The pressure is defined using the ideal gas law, $p = (\gamma - 1)(e - \frac{1}{2}\rho v^2)$. The flow solver is coupled to the structural solver through the nozzle area $A(x)$.

The boundary conditions are prescribed at the inlet, $x = 0$, and outlet, $x = 1$, using the Area-Mach-number relations, with a stagnation temperature of 300 K and a pressure of 100 kPa. The specific gas constant and the critical nozzle area are set as 287 J/(kg K) and 0.8, respectively.

The spatial derivatives in (24) are discretized with a third-order accurate summation-by-parts operator (Kreiss and Scherer 1974; Strand 1994), with boundary conditions imposed weakly using penalty terms (Funaro and Gottlieb 1988; Carpenter et al. 1994). The solution is stabilized with scalar third-order accurate artificial dissipation (Mattsson et al. 2004). The resulting set of nonlinear algebraic equations are denoted by

$$\boldsymbol{\mathcal{R}}_F(\boldsymbol{w}, \boldsymbol{u}) = \boldsymbol{0},$$

where $\boldsymbol{w}$ are the flow states and the $\boldsymbol{u}$ are the structure deformations (defined below) that determine the nozzle area $A(x)$.

#### *4.2.2 Linear elastic structural solver*

The structural deformations in the elastic nozzle wall are modeled using the Direct Stiffness Method (Turner 1959; Felippa 2001). The local stiffness system for frame elements is given by

$$\mathsf{K}_{e'} \boldsymbol{u}_{e'} = \boldsymbol{f}_{e'}, \tag{25}$$

where the local stiffness matrix and degrees of freedom are defined as,

$$\underbrace{\frac{wtE}{l}\begin{bmatrix} 1 & 0 & 0 & -1 & 0 & 0 \\ 0 & \frac{12t^2}{l^2} & \frac{6t^2}{l} & 0 & -\frac{12t^2}{l^2} & \frac{6t^2}{l} \\ 0 & \frac{6t^2}{l} & 4t^2 & 0 & -\frac{6t^2}{l} & 2t^2 \\ -1 & 0 & 0 & 1 & 0 & 0 \\ 0 & -\frac{12t^2}{l^2} & -\frac{6t^2}{l} & 0 & \frac{12t^2}{l^2} & -\frac{6t^2}{l} \\ 0 & \frac{6t^2}{l} & 2t^2 & 0 & -\frac{6t^2}{l} & 4t^2 \end{bmatrix}}_{\mathsf{K}_{e'}} \underbrace{\begin{pmatrix} u_{x',1} \\ u_{y',1} \\ u_{\theta,1} \\ u_{x',2} \\ u_{y',2} \\ u_{\theta,2} \end{pmatrix}}_{\boldsymbol{u}_{e'}}, \tag{26}$$

In (26), $w$ is the cross-section width, $t$ is the cross-section thickness, $l$ is the axial length, and $E$ is the Young's modulus

of the element. The subscripts 1 and 2 denote the information that belongs to the left and right nodes of the element, respectively. The degrees of freedom $\boldsymbol{u}_{e'} = (u_{x'}, u_{y'}, u_\theta)^T$ are the local axial, local transverse and rotational deformations, respectively.

Equation (25) is coupled to the flow solver through the local forcing vector,

$$\boldsymbol{f}_{e'} = \begin{pmatrix} f_{x',1} & f_{y',1} & m_1 & f_{x',2} & f_{y',2} & m_1 \end{pmatrix}^T.$$

In the elastic nozzle problem, the only forces exerted on the structure stem from the pressure of the flow through the nozzle. Since pressure forces are always normal to the surface, the local axial forces $f_{x'}$ and applied moments $m$ are always zero. The normal forces $f_{y'}$ are defined as

$$\begin{pmatrix} f_{y',1} \\ f_{y',2} \end{pmatrix} = -\frac{wl}{6} \begin{bmatrix} 2 & 1 \\ 1 & 2 \end{bmatrix} \begin{pmatrix} p_1 \\ p_2 \end{pmatrix},$$

where $p_1$ and $p_2$ are the flow pressures on the left and right nodes of the element.

The local stiffness system is transformed into the global Cartesian coordinate system to obtain

$$\mathsf{T}^T \mathsf{K}_{e'} \mathsf{T} \boldsymbol{u}_{e'} = \mathsf{T} \boldsymbol{f}_{e'}, \tag{27}$$

using the transformation matrix

$$\mathsf{T} = \begin{bmatrix} \cos\phi & \sin\phi & 0 & 0 & 0 & 0 \\ -\sin\phi & \cos\phi & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & \cos\phi & \sin\phi & 0 \\ 0 & 0 & 0 & -\sin\phi & \cos\phi & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix},$$

where $\phi$ is the angle between the frame element's axial centerline $x'$ and the global $x$ axis. The element systems in (27) are then assembled into a global system,

$$\boldsymbol{\mathcal{R}}_S(\boldsymbol{u}, \boldsymbol{w}) = \mathsf{K}\boldsymbol{u} - \boldsymbol{f}(\boldsymbol{w}) = \boldsymbol{0}, \tag{28}$$

where $\boldsymbol{u}$ contains the deformations of the nozzle at each node, defined in the global $(x, y, \theta)$ directions, and $\boldsymbol{w}$ are the flow states defined earlier.

One of the assumptions inherent to the flow solver is that the areas are always defined at fixed locations along the nozzle. Therefore, in order to simplify the transfer of information between the two disciplines, we fix the global $x$-displacements for all nodes. Since the inlet and outlet areas are also fixed, we impose simple-support boundary conditions at the left and right ends of the nozzle, that is

$$u_x(0) = u_x(L) = u_y(0) = u_y(L) = 0.$$

#### *4.2.3 Monolithic MDA and adjoint solutions*

It is clear that the quasi-one-dimensional nozzle flow and structural displacement are coupled: the pressure determines the displacement, which changes the nozzle shape, which determines the pressure. To solve this coupled MDA, which is necessary in MDF, we use a monolithic Newton-Krylov method. This iterative root-finding method solves the nonlinear discrete residual, denoted by the coupled discrete residual

$$\boldsymbol{\mathcal{R}}(\boldsymbol{w}, \boldsymbol{u}) = \begin{pmatrix} \boldsymbol{\mathcal{R}}_F(\boldsymbol{w}, \boldsymbol{u}) \\ \boldsymbol{\mathcal{R}}_S(\boldsymbol{u}, \boldsymbol{w}) \end{pmatrix} = \boldsymbol{0}. \tag{29}$$

Applying Newton's method to (29) produces

$$\begin{bmatrix} \frac{\partial \boldsymbol{\mathcal{R}}_F}{\partial \boldsymbol{w}} & \frac{\partial \boldsymbol{\mathcal{R}}_F}{\partial \boldsymbol{u}} \\ \frac{\partial \boldsymbol{\mathcal{R}}_S}{\partial \boldsymbol{w}} & \frac{\partial \boldsymbol{\mathcal{R}}_S}{\partial \boldsymbol{u}} \end{bmatrix} \begin{pmatrix} \Delta \boldsymbol{w} \\ \Delta \boldsymbol{u} \end{pmatrix} = \begin{pmatrix} -\boldsymbol{\mathcal{R}}_F \\ -\boldsymbol{\mathcal{R}}_S \end{pmatrix}, \tag{30}$$

which is solved at each Newton iteration using FGMRES. This linear solution is preconditioned with a block-Jacobi approach, where the (1, 1) flow block is solved using an LU factorization of a first-order accurate discretization based on nearest-neighbors, while the (2, 2) structural block is preconditioned using nested FGMRES with an absolute tolerance of $10^{-5}$. The nested FGMRES itself is preconditioned using the Gauss-Seidel method with 10 iterations. The MDA is converged to an absolute tolerance of $10^{-8}$.

The adjoint system uses the transpose of the matrix in (30) and is solved using FMGRES with a relative tolerance of $10^{-8}$. This solution is preconditioned with the transposed version of the block-Jacobi approach described above.

The IDF version of the problem solves the two disciplines independently. The discipline residuals are modified such that

$$\begin{pmatrix} \boldsymbol{\mathcal{R}}_F(\boldsymbol{w}, \bar{\boldsymbol{u}}) \\ \boldsymbol{\mathcal{R}}_S(\boldsymbol{u}, \bar{\boldsymbol{w}}) \end{pmatrix} = \boldsymbol{0},$$

where $\bar{\boldsymbol{u}}$ and $\bar{\boldsymbol{w}}$ are the target state variables prescribed by the optimization algorithm. In this particular problem, $\bar{\boldsymbol{u}}$ only consists of the transverse displacements at each node, $\boldsymbol{u}_y$, which are used to compute the nozzle areas for the flow solver. Similarly, $\bar{\boldsymbol{w}}$ only consists of the pressure at each nozzle node, $\boldsymbol{p}$, which are used to compute the structural forces. Under IDF, the disciplines are decoupled from each other and can be evaluated independently. Forward and adjoint linear systems for IDF use only the diagonal blocks of the matrix in (30), and their associated preconditioners. Multidisciplinary feasibility is enforced in IDF using constraints,

$$\boldsymbol{\mathcal{E}}_{\boldsymbol{w}}(\boldsymbol{u}) - \bar{\boldsymbol{u}} = 0 \quad \text{and} \quad \boldsymbol{\mathcal{E}}_{\boldsymbol{w}}(\boldsymbol{u}) - \bar{\boldsymbol{w}} = 0,$$

where $\boldsymbol{\mathcal{E}}_{\boldsymbol{w}}(\boldsymbol{u}) = \boldsymbol{u}_y$ and $\boldsymbol{\mathcal{E}}_{\boldsymbol{u}}(\boldsymbol{w}) = \boldsymbol{p}$.

#### *4.2.4 Nozzle geometry*

The nozzle area, $A(x)$, is parameterized using a cubic b-spline with an open uniform knot vector. The interior b-spline control points are used as design variables in the optimization. The nozzle areas at the inlet and the outlet are

fixed to be $A(0) = 2$ and $A(1) = 1.75$ respectively. The initial design variables are chosen such that they produce a linear nozzle area between the inlet and the outlet.

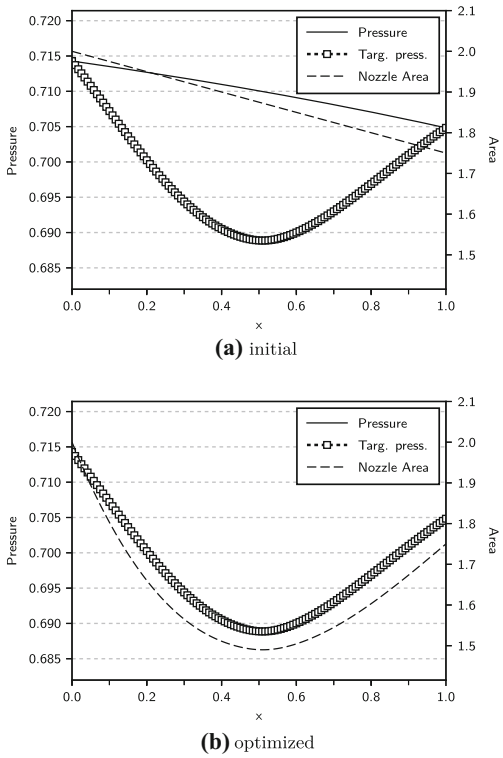
The displaced areas at each $x$-coordinate are calculated under the assumption of a rectangular nozzle cross-section with fixed width $w = 1\ m$ and varying height $h = 2[1-(y_0 + \boldsymbol{u}_y)]$, where $y_0$ is the initial $y$-coordinate of each structural node and $\boldsymbol{u}_y$ are the vertical deformations computed by the structural solver. Finally, the nozzle lenght is fixed at $l = 1$ m and the Young's Modulus at $E = 10^9$ Pa.

#### *4.2.5 Objective function and problem setup*

The objective function for the nozzle inverse design problem is

$$\mathcal{J} = \frac{1}{2}\int_0^1 (p - p_{targ})^2 dx, \tag{31}$$

where $p_{targ}$ is the target pressure distribution. The discrete integral is calculated with the same fourth-order accurate SBP quadrature used in the flow solver (Hicken and Zingg 2013). The target pressures are calculated from the MDA solution where the initial (i.e., unloaded) nozzle has a cubic dependence on $x$, such that the minimum area, $A(0.5) = 1.5$ occurs at the midpoint of the nozzle. Figure 6 shows the initial and optimal nozzle shapes, together with the corresponding pressure distributions. The target pressure is also provided and is seen to match the computed pressure on the optimal nozzle.




**(a)** initial

**(b)** optimized

**Fig. 6** Deformed nozzle areas and corresponding pressure distributions

**Table 3** Optimization parameters for the elastic nozzle problem

| Parameter | Value |
|---|---|
| Relative optimality tolerance, $\tau_p$ | $10^{-5}$ |
| Relative feasibility tolerance, $\tau_d$ | $10^{-5}$ |
| Initial trust radius, $\Delta_0$ | 2.0 |
| Maximum trust radius, $\Delta_{\max}$ | 4.0 |
| Initial penalty parameter, $\sigma_0$ | 0.1 |
| Initial Krylov tolerance, $\eta_0$ | 0.5 |
| Krylov subspace size | 15 |

The nozzle problem was solved using the MDF and IDF formulations for varying numbers of design variables ranging from 10 to 40 in increments of 2. Note that the $x$-axis in Fig. 8 only shows the number of b-spline control points: the IDF formulation for this problem has 242 additional coupling variables, and just as many Lagrange multipliers, that are also variables in the optimization algorithm.

A complete list of optimization parameters used in this problem is provided in Table 3.

#### *4.2.6 IDF preconditioner results*

We begin the numerical experiments by demonstrating the efficacy of the IDF preconditioner, as we did for the domain-decomposition problem. Figure 7 shows the convergence histories of the FLECS Krylov solver with 20 design variables, with and without the preconditioner. The sample presented here is representative of convergences typically observed throughout the optimization. Once again, the non-preconditioned solution stalls while the preconditioned solver is able to converge the relative KKT residual down three orders of magnitude for this particular nonlinear iteration. This mirrors our observations from the

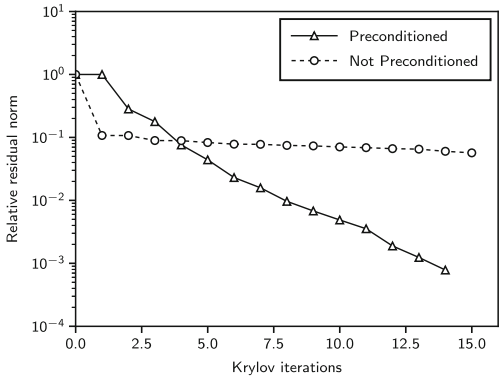



**Fig. 7** A sample of FLECS convergence histories for the IDF problem with 20 design variables

previous Laplace domain-decomposition problem. Without the preconditioner, the Krylov solver cannot produce useful Newton steps, and the optimization algorithm cannot make progress toward an optimum for the IDF problem.

### *4.2.7 Optimization results*

Figure 8 shows the cost scaling of the two MDO architectures. The computational cost is measured by recording the number of flow solver preconditioner applications and normalizing this by the number of flow solver preconditioner applications required for the MDA on the target geometry.

The cost scaling analysis for the elastic nozzle problem reveals a number of important findings. First, the Newton-Krylov MDA makes the MDF formulation considerably more efficient than what we observed with the block-Jacobi MDA of the domain-decomposition problem. Consequently, the MDF formulation here costs 8.3 fewer equivalent MDA evaluations, on average, relative to IDF—a 15.5% decrease. However, the inexact-Newton-Krylov strategy appears to exhibit better cost scaling with the IDF formulation than it does with MDF. The trust-region Newton-CG algorithm used for the MDF problem experiences an increase in cost at more than 30 design variables. Overall, the IDF formulation remains competitive in cost while offering significant advantages in terms of implementation.

We evaluate optimization convergence properties with sample convergence histories taken from the 20 and 40 design-variable cases, shown in Fig. 9. The inexact-Newton-Krylov method for IDF exhibits the expected superlinear convergence for both cases, and this is typical of the entire range of design variables we have tested. Convergence for the MDF formulation, however, loses superlinearity at higher numbers of design variables, indicating that the quality of the Newton step produced by the Krylov iterative method is degrading with the increasing size of the MDF optimization problem.

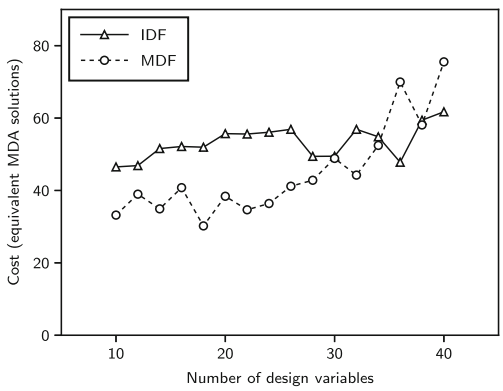
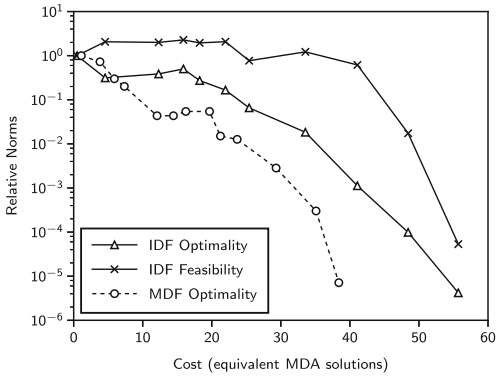
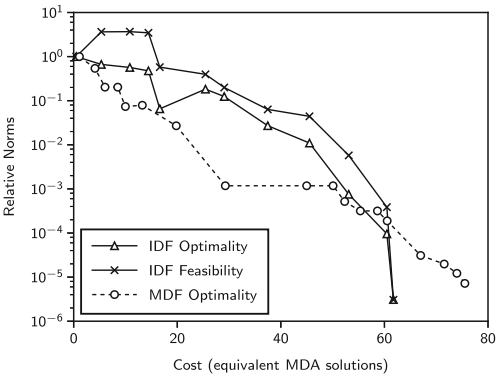



**Fig. 8** Computational cost of the optimization with varying numbers of design variables




**(a)** 20 design variables




**(b)** 40 design variables

**Fig. 9** Optimization convergence histories for the elastic nozzle problem

## 5 Conclusions

We have investigated the efficacy of an inexact-Newton-Krylov algorithm for the solution of multidisciplinary design optimization (MDO) problems formulated in both multidisciplinary feasible (MDF) and individual discipline feasible (IDF) architectures.

The inexact-Newton-Krylov approach applies Newton's method to the Karush-Kuhn-Tucker (KKT) first-order optimality conditions and solves the resulting system with a Krylov iterative method. To make this algorithm practical for the IDF formulation, we presented a matrix-free formulation for the KKT-matrix-vector products and developed a novel preconditioner for the linear primal-dual systems that arise in IDF.

On the Laplace problem with domain decomposition, the IDF formulation took an order of magnitude fewer domain solutions than the MDF formulation using a block-Jacobi MDA. The IDF formulation also exhibited considerably better cost scaling with increasing number of "disciplines". This is a significant finding, in that MDF using a block-Jacobi solution strategy for the MDA can be reformulated into a more efficient IDF problem without requiring

any new software infrastructure from the underlying PDE solvers. The IDF preconditioner was crucial in achieving this result: without it, the Krylov method could not converge and produce a useful Newton step.

The elastic nozzle problem illustrated a case with a more efficient Newton-Krylov MDA for the MDF formulation. Under these circumstances, optimization with MDF was found to be comparable to the IDF formulation. However, the inexact-Newton-Krylov strategy applied to IDF exhibited better cost scaling with respect to the number of design variables. Therefore, the IDF formulation remained competitive in cost while offering advantages in implementation difficulty over the MDF formulation with Newton-Krylov MDA.

Our numerical experiments suggest that the reduced-space inexact-Newton-Krylov algorithm, in conjunction with second-order adjoint-based KKT-matrix-vector products and an effective IDF preconditioner, can successfully mitigate the challenges associated with the IDF coupling variables and constraints. In future research, we plan to apply this strategy to a high-fidelity aerodynamic shape optimization problem under structural constraints. Our aim is to determine whether the trends we have observed in the present work translate to large-scale applications.

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

## Appendix

## Proof of Theorem 1

Consider an MDA with $d$ disciplines. Let the equation for the $i$th discipline be given by

$$\boldsymbol{\mathcal{R}}_i\left(\boldsymbol{w}_i, \{\boldsymbol{\mathcal{E}}_{ij}(\boldsymbol{w}_j) | j = 1, \ldots, d, j \neq i\}\right), \qquad \forall i = 1, \ldots, d.$$

where the set notation in the second argument indicates that the residual depends on the states of the other $d-1$ disciplines via the nonlinear mappings $\boldsymbol{\mathcal{E}}_{ij}$. Next, introduce the coupling variables, $\bar{\boldsymbol{w}}_{ij}$, which are defined by

$$\begin{aligned}\boldsymbol{\mathcal{C}}_{ij}(\boldsymbol{w}_j, \bar{\boldsymbol{w}}_{ij}) &\equiv \boldsymbol{\mathcal{E}}_{ij}(\boldsymbol{w}_j) - \bar{\boldsymbol{w}}_{ij} = 0,\\ &\forall i, j = 1, \ldots, d,\ j \neq i.\end{aligned}$$

Using the above conditions, the residual of the $i$th discipline can be expressed as a function of the coupling variables:

$$\boldsymbol{\mathcal{R}}_i\left(\boldsymbol{w}_i, \{\bar{\boldsymbol{w}}_{ij} | j = 1, \ldots, d, j \neq i\}\right), \qquad \forall i = 1, \ldots, d.$$

Let $\boldsymbol{w}^T = [\boldsymbol{w}_1^T, \ldots, \boldsymbol{w}_d^T]$ and $\bar{\boldsymbol{w}}^T = [\bar{\boldsymbol{w}}_{12}^T, \ldots, \bar{\boldsymbol{w}}_{d,d-1}^T]$ denote the compound vectors containing all the states and coupling variables, respectively, and consider the compound residual given by

$$\begin{pmatrix} \boldsymbol{\mathcal{R}}(\boldsymbol{w}, \bar{\boldsymbol{w}}) \\ \boldsymbol{\mathcal{C}}(\boldsymbol{w}, \bar{\boldsymbol{w}}) \end{pmatrix} = \begin{pmatrix} \boldsymbol{\mathcal{R}}_1(\boldsymbol{w}_1, \bar{\boldsymbol{w}}_{12}, \ldots, \bar{\boldsymbol{w}}_{1d}) \\ \vdots \\ \boldsymbol{\mathcal{R}}_d(\boldsymbol{w}_d, \bar{\boldsymbol{w}}_{d1}, \ldots, \bar{\boldsymbol{w}}_{d,d-1}) \\ \boldsymbol{\mathcal{C}}_{12}(\boldsymbol{w}_2, \bar{\boldsymbol{w}}_{12}) \\ \vdots \\ \boldsymbol{\mathcal{C}}_{d,d-1}(\boldsymbol{w}_{d-1}, \bar{\boldsymbol{w}}_{d,d-1}) \end{pmatrix}.$$

The Jacobian of this compound residual is the matrix

$$\mathsf{D} \equiv \begin{bmatrix} \frac{\partial \boldsymbol{\mathcal{R}}}{\partial \boldsymbol{w}} & \frac{\partial \boldsymbol{\mathcal{R}}}{\partial \bar{\boldsymbol{w}}} \\ \frac{\partial \boldsymbol{\mathcal{C}}}{\partial \boldsymbol{w}} & -\mathsf{I} \end{bmatrix}.$$

The (2,2) block of $\mathsf{D}$ is the negative identity, which is clearly invertible, and the (1,1) block of $\mathsf{D}$ is the block diagonal matrix $\mathsf{diag}(\mathsf{A}_1, \ldots, \mathsf{A}_d)$, which is also invertible by assumption that the $\mathsf{A}_i$ are invertible. Therefore, we can form the Schur complement of $\mathsf{D}$ with respect to either block.

The Schur complement with respect to the (2,2) block of $\mathsf{D}$ is the matrix

$$\mathsf{S}_{(2,2)} \equiv \frac{\partial \boldsymbol{\mathcal{R}}}{\partial \boldsymbol{w}} + \frac{\partial \boldsymbol{\mathcal{R}}}{\partial \bar{\boldsymbol{w}}} \frac{\partial \boldsymbol{\mathcal{C}}}{\partial \boldsymbol{w}}.$$

Now, the $(i, j)$ block of second term above is, for $i \neq j$,

$$\left(\frac{\partial \boldsymbol{\mathcal{R}}}{\partial \bar{\boldsymbol{w}}} \frac{\partial \boldsymbol{\mathcal{C}}}{\partial \boldsymbol{w}}\right)_{i,j} = \frac{\partial \boldsymbol{\mathcal{R}}_i}{\partial \bar{\boldsymbol{w}}_{ij}} \frac{\partial \boldsymbol{\mathcal{E}}_{ij}}{\partial \boldsymbol{w}_j} = \frac{\partial \boldsymbol{\mathcal{R}}_i}{\partial \boldsymbol{w}_j},$$

where the last equality follows from the chain rule. There is no contribution to the $(i, i)$ block from the second term, since there is no coupling variable between a discipline and itself. Conversely, recall that the first term in the Schur complement is $\mathsf{A}_i = \partial \boldsymbol{\mathcal{R}}_i / \partial \boldsymbol{w}_i$, which has no off-diagonal blocks. Summarizing, the Schur complement with respect to the (2,2) block of $\mathsf{D}$ is the monolithic MDA Jacobian

$$\mathsf{S}_{(2,2)} = \left(\frac{\partial \boldsymbol{\mathcal{R}}}{\partial \boldsymbol{w}} + \frac{\partial \boldsymbol{\mathcal{R}}}{\partial \bar{\boldsymbol{w}}} \frac{\partial \boldsymbol{\mathcal{C}}}{\partial \boldsymbol{w}}\right)_{i,j} = \frac{\partial \boldsymbol{\mathcal{R}}_i}{\partial \boldsymbol{w}_j}.$$

Next, we consider the Schur complement of $\mathsf{D}$ with respect to the (1,1) block:

$$\mathsf{S}_{(1,1)} \equiv -\mathsf{I} - \frac{\partial \boldsymbol{\mathcal{C}}}{\partial \boldsymbol{w}} \left[\frac{\partial \boldsymbol{\mathcal{R}}}{\partial \boldsymbol{w}}\right]^{-1} \frac{\partial \boldsymbol{\mathcal{R}}}{\partial \bar{\boldsymbol{w}}}.$$

Now, the direct sensitivities of $\boldsymbol{w}_i$ with respect to $\bar{\boldsymbol{w}}_{ij}$ can be found by taking the total derivative of $\boldsymbol{\mathcal{R}}_i = \boldsymbol{0}$:

$$\begin{aligned}\frac{d\boldsymbol{\mathcal{R}}_i}{d\bar{\boldsymbol{w}}_{ij}} &= \frac{\partial \boldsymbol{\mathcal{R}}_i}{\partial \bar{\boldsymbol{w}}_{ij}} + \frac{\partial \boldsymbol{\mathcal{R}}_i}{\partial \boldsymbol{w}_i} \frac{d\boldsymbol{w}_i}{d\bar{\boldsymbol{w}}_{ij}} = 0\\ \Rightarrow \qquad \frac{d\boldsymbol{w}_i}{d\bar{\boldsymbol{w}}_{ij}} &= -\left[\frac{\partial \boldsymbol{\mathcal{R}}_i}{\partial \boldsymbol{w}_i}\right]^{-1} \frac{\partial \boldsymbol{\mathcal{R}}_i}{\partial \bar{\boldsymbol{w}}_{ij}}.\end{aligned}$$

Consequently, substituting this expression, the Schur complement with respect to the (1,1) block of $\mathsf{D}$ simplifies to

$$\mathsf{S}_{(1,1)} = -\mathsf{I} + \frac{\partial \mathcal{C}}{\partial \boldsymbol{w}} \frac{d\boldsymbol{w}}{d\bar{\boldsymbol{w}}} = \frac{d\mathcal{C}}{d\bar{\boldsymbol{w}}},$$

which is the total derivative of the coupling constraints with respect to the coupling variables, i.e. the IDF Jacobian.

The desired result now follows. If the MDA Jacobian is invertible, i.e. the Schur complement $\mathsf{S}_{(2,2)}$ is invertible, then $\mathsf{D}$ is invertible. If $\mathsf{D}$ is invertible, then the IDF Jacobian must be invertible, since it is the Schur complement $\mathsf{S}_{(1,1)}$. The reverse implication is analogous. This completes the proof.

## References


Arreckx S, Lambe AB, Martins JRRA, Orban D (2016) A matrix-free augmented lagrangian algorithm with application to large-scale structural design optimization. Optim Eng 17(2):359–384

Benzi M, Golub GH, Liesen J (2005) Numerical solution of saddle point problems. Acta Numer 14:1–137. doi:10.1017/s0962492904000212

Biros G, Ghattas O (2005a) Parallel lagrange–newton–krylov–schur methods for pde-constrained optimization. part i: the krylov–schur solver. SIAM J Sci Comput 27(2):687–713

Biros G, Ghattas O (2005b) 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 J Sci Comput 27(2):714–739

Bloebaum C (1995) Coupling strength-based system reduction for complex engineering design. Struct Optim 10(2):113–121

Byrd RH, Curtis FE, Nocedal J (2008) An inexact sqp method for equality constrained optimization. SIAM J Optim 19(1):351–369

Byrd RH, Curtis FE, Nocedal J (2010) An inexact newton method for nonconvex equality constrained optimization. Math Program 122(2):273–299

Carpenter MH, Gottlieb D, Abarbanel S (1994) Time-stable boundary conditions for finite-difference schemes solving hyperbolic systems: methodology and application to high-order compact schemes. J Comput Phys 111(2):220–236. doi:10.1006/jcph.1994.1057

Conn AR, Gould NI, Toint PL (2000) Trust region methods. SIAM

Cramer EJ, Dennis JE Jr, Frank PD, Lewis RM, Shubin GR (1994) Problem formulation for multidisciplinary optimization. SIAM J Optim 4(4):754–776

Eisenstat SC, Walker HF (1996) Choosing the forcing terms in an inexact newton method. SIAM J Sci Comput 17(1):16–32

Felippa CA (2001) A historical outline of matrix structural analysis: a play in three acts. Comput Struct 79(14):1313–1324

Fletcher R, Leyffer S (2002) Nonlinear programming without a penalty function. Math Program 91(2):239–269

Funaro D, Gottlieb D (1988) A new method of imposing boundary conditions in pseudospectral approximations of hyperbolic equations. Math Comput 51(184):599–613

Gill PE, Murray W, Saunders MA (2005) Snopt: an sqp algorithm for large-scale constrained optimization. SIAM Rev 47(1):99–131

Haftka RT (1985) Simultaneous analysis and design. AIAA J 23(7):1099–1103

Haftka RT, Sobieszczanski-Sobieski J, Padula SL (1992) On options for interdisciplinary analysis and design optimization. Struct Optim 4(2):65–74

Heinkenschloss M, Ridzal D (2008) An inexact trust-region sqp method with applications to pde-constrained optimization. In: Numerical mathematics and advanced applications. Springer, pp 613–620

Heinkenschloss M, Ridzal D (2014) A matrix-free trust-region sqp method for equality constrained optimization. SIAM J Optim 24(3):1507–1541

Heroux MA, Bartlett RA, Howle VE, Hoekstra RJ, Hu JJ, Kolda TG, Lehoucq RB, Long KR, Pawlowski RP, Phipps ET, Salinger AG, Thornquist HK, Tuminaro RS, Willenbring JM, Williams A, Stanley KS (2005) An overview of the trilinos project. ACM Trans Math Softw 31(3):397–423. doi:10.1145/1089014.1089021

Herskovits J, Mappa P, Goulart E, Soares CM (2005) Mathematical programming models and algorithms for engineering design optimization. Comput Methods Appl Mech Eng 194(30):3244–3268

Hicken JE, Dener A (2015) A flexible iterative solver for nonconvex, equality-constrained quadratic subproblems. SIAM J Sci Comput 37(4):A1801–A1824

Hicken JE, Zingg DW (2013) Summation-by-parts operators and high-order quadrature. J Comput Appl Math 237(1):111–125. doi:10.1016/j.cam.2012.07.015

Jameson A (1989) Aerodynamic design via control theory. In: Recent advances in computational fluid dynamics. Springer, pp 377–401

Kennedy GJ, Martins JRRA (2010) Parallel solution methods for aerostructural analysis and design optimization. In: 13th AIAA/ISSMO multidisciplinary analysis optimization conference, p 9308

Kenway GKW, Kennedy GJ, Martins JRRA (2014) Scalable parallel approach for high-fidelity steady-state aeroelastic analysis and adjoint derivative computations. AIAA J 52(5):935–951

Knoll DA, Keyes DE (2004) Jacobian-free newton–krylov methods: a survey of approaches and applications. J Comput Phys 193(2):357–397

Kodiyalam S, Sobieszczanski-Sobieski J (2001) Multidisciplinary design optimisation-some formal methods, framework requirements, and application to vehicle design. Int J Veh Des 25(1–2):3–22

Kreiss HO, Scherer G (1974) Finite element and finite difference methods for hyperbolic partial differential equations. In: de Boor C (ed) Mathematical aspects of finite elements in partial differential equations. Mathematics Research Center, the University of Wisconsin. Academic Press, New York

Martins JRRA, Lambe AB (2013) Multidisciplinary design optimization: a survey of architectures. AIAA J 51(9):2049–2075

Martins JRRA, Alonso JJ, Reuther JJ (2004) High-fidelity aerostructural design optimization of a supersonic business jet. J Aircr 41(3):523–530

Mattsson K, Svärd M, Nordström J (2004) Stable and accurate artificial dissipation. J Sci Comput 21(1):57–79

Maute K, Nikbay M, Farhat C (2001) Coupled analytical sensitivity analysis and optimization of three-dimensional nonlinear aeroelastic systems. AIAA J 39(11):2051–2061

Maute K, Nikbay M, Farhat C (2003) Sensitivity analysis and design optimization of three-dimensional non-linear aeroelastic systems by the adjoint method. Int J Numer Methods Eng 56(6):911–933

Nash SG, Nocedal J (1991) A numerical study of the limited memory bfgs method and the truncated-newton method for large scale optimization. SIAM J Optim 1(3):358–372

Özkaya E, Gauger NR (2009) Single-step one-shot aerodynamic shape optimization. In: Optimal control of coupled systems of partial differential equations. Springer, pp 191–204

Schittkowski K (1986) Nlpql: a fortran subroutine solving constrained nonlinear programming problems. Ann Oper Res 5(2):485–500

Steihaug T (1983) The conjugate gradient method and trust regions in large scale optimization. SIAM J Numer Anal 20(3):626–637

Strand B (1994) Summation by parts for finite difference approximations for d/dx. J Comput Phys 110(1):47–67. doi:10.1006/jcph.1994.1005

Ta'asan S, Kuruvila G, Salas M (1992) Aerodynamic design and optimization in one shot. In: 30th aerospace sciences meeting and exhibit, p 25

Turner M (1959) The direct stiffness method of structural analysis. Boeing Airplane Company

Wang Z, Navon IM, Le Dimet F, Zou X (1992) The second order adjoint analysis: theory and applications. Meteorog Atmos Phys 50(1–3):3–20

Wright S, Nocedal J (2006) Numerical optimization, 2nd edn. Springer Science