A Parallel Finite-Element Framework for Large-Scale Gradient-Based Design Optimization of High-Performance Structures

Graeme J. Kennedy[a,1,*], Joaquim R.R.A. Martins[b,2]

[a]*Georgia Institute of Technology, School of Aerospace Engineering, Atlanta, Georgia, 30332-0150*
[b]*University of Michigan, Department of Aerospace Engineering, Ann Arbor, Michigan, 48109*

Abstract

Structural optimization using gradient-based methods is an extremely powerful design technique that is well suited for the design of high-performance structures. However, as the complexity of finite-element models and design formulations grow, the computational cost of computing the gradient may become a computational bottleneck in the overall time required to solve a structural optimization problem. Furthermore, in light of current high-performance computing trends, any methods intended to address this bottleneck must efficiently utilize parallel computing resources. Therefore, there is a need for solution and gradient evaluation methods that scale well with numbers of design variables, numbers of functions, and numbers of processors. In this paper, we present an integrated parallel finite-element analysis tool for gradient-based design optimization designed to address these issues. This framework is designed to solve large-scale high-fidelity structural optimization problems with thousands of design variables, millions of state variables and hundreds of load cases using specialized parallel solution methods. We describe the most relevant details of the parallel algorithms used within the tool. We present a consistent constraint formulation and aggregation technique for both material failure and buckling constraints. To demonstrate both the solution and functional accuracy, we compare our results to an exact solution of a pressure-loaded cylinder made with either isotropic or orthotropic material. To demonstrate the parallel solution and gradient evaluation performance, we use a large transport aircraft wing with over 5.44 million unknowns. The results show excellent scalability of the structural solution and gradient computation with number of functions, number of design variables, and number of processors which makes this framework well suited for large-scale high-fidelity structural design optimization.

1. Introduction

Over the past few decades increasingly powerful high-performance computational resources and the development of sophisticated numerical algorithms, have enabled the solution of large-scale, high-fidelity structural design optimization problems (Venkataraman and Haftka, 2004). Here we use the term *large-scale* to refer to design problems

*Corresponding author
Email addresses: graeme.kennedy@aerospace.gatech.edu (Graeme J. Kennedy), jrram@umich.edu (Joaquim R.R.A. Martins)
[1]Assistant Professor
[2]Associate Professor

with either a large number of design variables, a large number of structural state variables, a large number of load cases or a combination thereof such that significant high-performance parallel computing resources are required to solve the problem within a reasonable time. This definition will change depending on advancements in high-performance computing hardware. At present, this definition of large-scale translates roughly to design problems with either more than $O(10^5)$ design variables, $O(10^6)$ state variables, or $O(10^2)$ load cases. Several authors have presented solution methods for large-scale problems, including structural shape and sizing problems (Padula and Stone, 1998; Papadrakakis et al., 2003, 2001), large 3D topology problems with $O(10^6)$ design variables (Borrvall and Petersson, 2001; Wang et al., 2007), and high-fidelity multidisciplinary design optimization problems involving structural analysis as a discipline with $O(10^6)$ state variables (Kenway et al., 2013).

In this paper, we present an integrated approach to parallel analysis and gradient evaluation for large-scale structural design optimization problems. We have developed this framework for the analysis and design of thin shell structures that are used in many high-performance aerospace applications where strength, weight and stiffness are critical design considerations. These aerospace structures are often manufactured using high-performance materials, such as laminated composites or advanced metallic alloys, that can be utilized to achieve high stiffness-to-weight and strength-to-weight ratios. As a result, the design problem involves the simultaneous consideration of the geometry of the structure, the sizing of the members and, in the case of composites, manufacturing details such as the lamination stacking sequence (Abrate, 1994). Therefore, the structural design problem may include stacking sequence design optimization schemes that can significantly increase the size of the design space (Stegmann and Lund, 2005; Hvejsel and Lund, 2011; Hvejsel et al., 2011; Kennedy and Martins, 2013b). In addition, slender shell structures subjected to in-plane loading are susceptible to buckling and, as a result, structural requirements frequently include both strength and buckling constraints. Within this framework, we impose the buckling constraints in a conventional manner using a global-local analysis approach in which a global model is used to determine the edge-loads for a local stiffened panel buckling problem. Other authors have reformulated these design problems using a bi-level approach where the thicknesses are determined from the global level design problem and the detailed lamination sequence is determined from a local-level design problem (Liu and Haftka, 2004; Liu et al., 2000a; Herencia et al., 2008; Liu et al., 2008).

Although gradient-free optimization methods have been successfully applied to many important structural design problems, including lamination stacking sequence design (Le Riche and Haftka, 1993; Liu et al., 2000b; Adams et al., 2004) and sizing and shape optimization problems (Jansen and Perez, 2011), these applications typically involve at most $O(10^2)$ design variables. While gradient-free methods only require function values and are therefore easier to use, these methods scale very poorly with increasing dimensionality of the design space. Since our focus is on large-scale, high-fidelity applications, we use gradient-based methods and accept the additional requirement of evaluating the gradient of the objective and constraint functions in the design optimization problem. There are two questions that must be raised when evaluating the gradients required for optimization: first, how much computational time will be required to evaluate the derivatives and second, how accurately must the gradient be in order for the optimization to converge. For many large-scale linear structural design problems the evaluation of the objective and constraint

gradients requires more time than the analysis and function evaluations and is therefore the critical computational bottleneck (Venkataraman and Haftka, 2004). In order to minimize the computational time, we use the adjoint method which requires more time for each additional gradient, but scales very slowly with the number of design variables. We detail the precise costs of our adjoint implementation in Section 5.3. The other issue when evaluating the gradient is the accuracy. While some work has been done on gradient-based optimization algorithms which are designed to converge with inexact gradient information (Carter, 1991), in this work we focus on minimizing the amount of time required to evaluate the gradient to the maximum precision possible. In order to achieve this goal, we do not use any finite-difference methods to evaluate derivatives. Instead, we exclusively use hand-coded derivative routines that achieve good computational performance while avoiding subtractive cancellation issues suffered by finite-difference methods. We note that other authors have used automatic differentiation methods, rather than hand-coded routines, to obtain accurate derivatives (Mader et al., 2008). In order to verify the accuracy of our derivative implementation, we use the complex-step derivative evaluation technique. The complex-step method uses a complex perturbation of the variables to determine the derivative and, unlike finite-difference methods, does not suffer from subtractive cancelation. As a result, a very small step size may be used, yielding derivatives with the same number of significant digits as the functional estimate (Squire and Trapp, 1998; Martins et al., 2003).

In this paper we present a fully verified, integrated framework for parallel analysis and gradient-evaluation of shell structures. We verify that our methods achieve the optimal solution and functional accuracy. In our opinion, within a design optimization framework, functional accuracy and solution accuracy are of equal importance, yet functional accuracy is often overlooked in structural optimization applications. In addition, we verify our gradient evaluation methods using a complex-step derivative approximation technique that enables accurate verification of derivatives without loss of accuracy due to subtractive cancellation. We have integrated the developments presented in this paper into a sophisticated parallel finite-element code that we call the Toolkit for the Analysis of Composite Structures (TACS). We have used TACS for large-scale structural analysis of composite beams (Kennedy and Martins, 2012b), as well as structural topology optimization (Lee and Martins, 2012; Lee et al., 2012), lamination sequence design (Kennedy and Martins, 2013b), and both static and dynamic aeroelastic design optimization (Kennedy and Martins, 2010; Kenway et al., 2013; Kennedy and Martins, 2012a; Liem et al., 2012; Kennedy and Martins, 2013a).

1.1. The model optimization problem

Since structural weight reduction is critical in many aerospace applications, the most common structural design problem is to minimize the structural mass subject to stress and possibly buckling constraints. These structural constraints are imposed at a series of design load cases to ensure the safety of the aerospace vehicle within a prescribed operational envelope. With this standard structural design optimization problem in mind, we pose the following

generic structural design optimization problem:

$$\begin{aligned} \text{minimize} \quad & f(\mathbf{x}, \mathbf{u}_1, \ldots, \mathbf{u}_{n_\ell}) \\ \text{with respect to} \quad & \mathbf{x},\ \mathbf{u}_1, \ldots,\ \mathbf{u}_{n_\ell} \\ \text{governed by} \quad & \mathbf{R}_i(\mathbf{X}^N, \mathbf{x}_M, \mathbf{u}_i) = 0 \qquad \text{for } 1 \le i \le n_\ell \\ \text{such that} \quad & \mathbf{f}_i(\mathbf{x}, \mathbf{u}_i) \le 1 \\ & \mathbf{x}_l \le \mathbf{x} \le \mathbf{x}_u \end{aligned} \tag{1}$$

where $f(\mathbf{x}, \mathbf{u}_1, \ldots, \mathbf{u}_{n_\ell})$ is the objective function and $\mathbf{f}_i(\mathbf{x}, \mathbf{u}_i) \in \mathbb{R}^{n_f}$ represents a vector of constraints for the i^{th} load case. Note that there are a total of n_ℓ load cases. The design variables $\mathbf{x} = (\mathbf{x}_G, \mathbf{x}_M) \in \mathbb{R}^{n_x}$ are partitioned into either geometric or material design variables which we denote $\mathbf{x}_G \in \mathbb{R}^{n_{xg}}$ and $\mathbf{x}_M \in \mathbb{R}^{n_{xm}}$, respectively. The distinction between geometric and material design variables arises at the element level: geometric design variables modify the element nodes and material design variables modify the element constitutive behavior. The finite element residuals $\mathbf{R}_i \in \mathbb{R}^{6n}$ depend on the finite-element nodal locations $\mathbf{X}^N = \mathbf{X}^N(\mathbf{x}_G) \in \mathbb{R}^{3n}$, the design variables $\mathbf{x}$, and the state variables $\mathbf{u}_i \in \mathbb{R}^{6n}$, for the i^{th} load case.

While there are numerous techniques that have been developed to solve the optimization problem (1), we employ reduced spaced approach where the governing equations for each load case, $\mathbf{R}_i(\mathbf{X}^N, \mathbf{x}_M, \mathbf{u}_i) = 0$, are solved at each optimization iteration and the optimization problem is recast solely in terms of design variables. In this approach, the state variables are implicit functions of the design variables and the adjoint or direct method must be used to determine the objective and constraint gradients (Martins and Hwang, 2013). Note that we do not solve the optimization problem directly using our framework, but instead provide the objective and constraint values and gradients to a gradient-based optimizer. Typically, we solve the optimization problem with the Python-based optimization package pyOpt (Perez et al., 2012), where we often use the optimizer SNOPT (Gill et al., 2005).

The remainder of this paper is structured as follows: In Section 2, we briefly describe the linear shell element used for structural analysis. In Section 3, we describe the parallel finite-element methods we use to solve the linear systems arising from the finite-element discretization of shell structures and in the adjoint method. In Section 4, we describe the functions of interest that are used within the optimization problem (1). Finally, in Section 5, we describe the gradient evaluation method that we have implemented and we evaluate their computational performance.

2. Shell formulation

In this section, we briefly describe the general purpose higher-order shell element formulation used for linear analysis within our integrated framework. The shell element employs a mixed-interpolation of tensorial components (MITC) formulation (Dvorkin and Bathe, 1984; Bucalem and Bathe, 1993; Bathe et al., 2000) to avoid shear and membrane locking. One of the key advantages of the MITC-based approach is that it easily extends to high-order shell element formulations. The formulation enables strain-free, linearized rigid body rotations. The drilling degrees

of freedom are added based on a penalization approach to facilitate shell-shell intersections and shell-beam connections (Hughes and Brezzi, 1989; Fox and Simo, 1992). In addition, the drilling degree of freedom formulation enables the use block-based matrix algorithms since the number of degrees of freedom per node is fixed across the entire structural mesh (Simo, 1993). Following Milford and Schnobrich (1986), the formulation uses an explicit integration of the strain energy through the thickness, enabling the direct use of the classical first-order deformation theory (FSDT) constitutive relationships. This explicit integration approach introduces a modeling error on the order of the ratio of the thickness to the minimum radius of curvature (Buechter and Ramm, 1992). As a result, the shell element is not appropriate for very thick shells or shells with very high-curvature.

2.1. The element implementation

In the finite-element implementation, we use bi-Lagrange shape functions of order p to interpolate the mid-surface displacements, $\mathbf{U}_0$, and the small rotation angles, $\boldsymbol{\theta}$, based on the element nodal variables $\mathbf{u}^e \in \mathbb{R}^{6p^2}$, as follows:

$$\begin{bmatrix} \mathbf{U}_0(\boldsymbol{\xi}) \\ \boldsymbol{\theta}(\boldsymbol{\xi}) \end{bmatrix} = \mathbf{N}(\boldsymbol{\xi})\,\mathbf{u}^e \tag{2}$$

where $\mathbf{N}(\boldsymbol{\xi}) \in \mathbb{R}^{6\times n_e}$ are the shape functions and $\boldsymbol{\xi}$ are the isoparametric coordinates. The element residual is derived based on the method of virtual work and takes the form

$$\delta\mathbf{u}^{eT}\,\mathbf{R}^e(\mathbf{X}^e, \mathbf{x}_M, \mathbf{u}^e) = \delta W^e,$$

where $\mathbf{R}^e(\mathbf{X}^e, \mathbf{x}_M, \mathbf{u}^e)$ are the element residuals written as a function of the element nodal displacements and linearized Euler angles, $\mathbf{u}^e$, the element nodal locations, $\mathbf{X}^e$, and the material design variables, $\mathbf{x}_M$.

The element nodal locations and element displacements and rotations can be obtained based on the element injection operators $\mathbf{P}_j \in \mathbb{R}^{n\times p^2}$, for $j = 1, \ldots, N_e$, which distribute the element residual components to the global residual as follows:

$$\mathbf{R}_i(\mathbf{X}^N, \mathbf{x}, \mathbf{u}) = \sum_{j=1}^{N_e} \left(\mathbf{P}_j \otimes \mathbf{I}_6\right) \mathbf{R}_j^e(\mathbf{X}^e, \mathbf{x}_M, \mathbf{u}^e),$$

where $\otimes$ denotes the Kronecker product and $\mathbf{I}_6$ is the 6×6 identity matrix. Note that the element nodal displacements and rotations, $\mathbf{u}^e$, as well as the element nodal locations, $\mathbf{X}^e$ can be also be obtained from the global solution vector and global nodal location vector using the injection operators, $\mathbf{P}_j$, as follows:

$$\mathbf{u}^e = \left(\mathbf{P}_j^T \otimes \mathbf{I}_6\right)\mathbf{u},$$
$$\mathbf{X}^e = \left(\mathbf{P}_j^T \otimes \mathbf{I}_3\right)\mathbf{X}^N,$$

where $\mathbf{I}_3$ is the 3×3 identity matrix.

3. Parallel finite-element analysis

Parallel structural finite-element solvers used for gradient-based optimization must perform three central tasks efficiently in parallel: the assembly of the residual and stiffness matrix; the solution of linear systems arising from the finite-element discretization; and the parallel evaluation of functions and gradients required for design optimization. Of the these three required tasks, the most challenging to implement efficiently in parallel is the solution of linear systems. In this framework, due to the poor conditioning of the stiffness matrices associated with the analysis of thin shells, we use a parallel direct matrix factorization. However, as noted previously, often the critical bottleneck in gradient-based design optimization is the computation of the required gradients. Therefore, we seek an approach in which we balance the performance of assembly tasks, required for the evaluation of constraint and objective gradients, and the cost of the direct matrix factorization. This requirement lead us to use a domain-decomposition based approach to direct matrix factorization.

Typically, the matrices arising from the finite-element discretization of structures are symmetric. Here, however, we are interested in aeroelastic applications where nonlinear following forces may be present. Following forces are non-conservative and introduce non-symmetric terms in the Jacobian of the structural residuals (Elishakoff, 2005). The simplest approach would be to neglect the non-symmetric part of the Jacobian. However, in order to obtain accurate gradients it is critical to use the exact Jacobian of the structural residuals. Therefore, we concentrate on developing general non-symmetric direct solvers that can be used even when follower forces are present.

A number of authors have developed algorithms to solve non-symmetric sparse linear systems using parallel direct solution methods. Amestoy et al. (2000), developed the code MUMPS (MUltifrontal Massively Parallel Solver), a multi-frontal matrix factorization framework for distributed and multi-core architectures that uses partial pivoting to achieve stability. To achieve parallelism, each front in the factorization is assigned to a different processors. As the factorization proceeds, the size of the frontal matrices grow. Once the frontal matrices reach a certain size, they are factored across groups of processors using a row-oriented storage format. Finally, the root frontal matrix is factored in parallel using a dense 2D block-cyclic factorization algorithm in ScaLAPACK (Blackford et al., 1996). The direct solver SPOOLES (SParse Object Oriented Linear Equations Solver) developed by Ashcraft and Grimes (1999) also uses a multi-frontal factorization approach with either partial or full pivoting strategy. In SPOOLES parallelism in either distributed or multi-core environments is also achieved by assigning the fronts to different processes. Li and Demmel (2003) developed SuperLU_DIST a parallel distributed direct solver for both symmetric and non-symmetric linear systems. SuperLU_DIST uses a sparse block-cyclic data storage format where the size of the blocks are determined based on an analysis of the matrix. SuperLU_DIST uses a static pivoting approach to factorization with pre-processing heuristics designed to maximize the entries of the matrix on the diagonal. Excellent parallel performance is achieved by utilizing a process queue to enable execution of independent factorization tasks, and parallelizing the trailing-matrix update, which constitutes the single largest computational bottleneck in the factorization.

In the following section, we present our parallel direct matrix factorization approach. In a departure from the

direct solvers discussed above, we utilize a domain-decomposition, or sub-structuring, approach that employs static pivoting. This approach achieves excellent parallel performance, and conforms well with the repeated factorizations that are required at each optimization iteration.

3.1. Parallel direct matrix factorization

The matrix factorization proceeds in three main stages: the computation of the local Schur complement contribution; the global Schur complement assembly across all processors; and the global Schur complement factorization in parallel across all processors. Prior to factorization, the matrix is computed and stored in memory in a distributed fashion across all processors. Each processor has a local contribution to the global matrix that is partitioned as follows:

$$\mathbf{A}_i = \begin{bmatrix} \mathbf{B}_i & \mathbf{E}_i \\ \mathbf{F}_i & \mathbf{C}_i \end{bmatrix}, \tag{3}$$

with $i = 1, \ldots, N_p$, where N_p are the number of processors participating in the matrix factorization. The local matrix is split into blocks corresponding to unknowns in the interior of domain i, and interface unknowns that lie on the border between two or more domains. The matrix $\mathbf{B}_i$ represents the contributions from internal unknowns, the matrix $\mathbf{C}_i$ represents the contributions from the interface unknowns, and the matrices $\mathbf{E}_i$, and $\mathbf{F}_i$ couple the internal and interface unknowns for each domain. Note that the ordering of matrix $\mathbf{A}_i$ is constrained since the interface unknowns must be ordered last.

The first step in the factorization process is the calculation of the local contribution to the global Schur complement. We obtain the local Schur complement contribution by first computing the LU-decomposition of the diagonal block matrix, $\mathbf{B}_i$, such that $\mathbf{B}_i = \mathbf{L}_{B_i}\mathbf{U}_{B_i}$. Next, we compute the local Schur complement, $\mathbf{S}_i$, of the local matrix (3),

$$\mathbf{S}_i = \mathbf{C}_i - \mathbf{F}_i\mathbf{U}_{B_i}^{-1}\mathbf{L}_{B_i}^{-1}\mathbf{E}_i. \tag{4}$$

Note that both the LU-factorization of the block matrix $\mathbf{B}_i$, and the computation of the local Schur complement (4) are independent of the calculations on any other processors. For good parallel performance it is essential to achieve a good balance of the workload so that each processor completes the local calculation in roughly the same amount of time. Unfortunately, this is often difficult to achieve since equally sized domains may not have the same matrix structure and often require different computational times.

Once each processor has completed the local Schur complement computation, the global Schur complement can be assembled on all processors as follows:

$$\mathbf{S} = \sum_{i}^{N_p} \mathbf{T}_i^T\mathbf{S}_i\mathbf{T}_i. \tag{5}$$

The permutation matrices, $\mathbf{T}_i$, are selected to ensure a low fill-in for the LU factorization of the global Schur complement. Once the global Schur complement is formed, it is factored such that

$$\mathbf{S} = \mathbf{L}_S\mathbf{U}_S. \tag{6}$$

Note that both the assembly and factorization of the global Schur complement matrix require communication amongst all processors.

This outline summarizes the steps involved in the direct matrix factorization, however, the performance of the implementation depends in large part on the data structures and algorithms used for each step of the factorization. We now describe in greater detail the data structures and operations that are used to compute the local Schur complement matrix computation (4) and factorization of the global Schur complement (6).

3.1.1. Local Schur complement factorization

On each processor, we store the local matrix (3), in a block-compressed sparse row format (BCSR) (Saad, 2003). In this format, the matrix elements corresponding to a single finite-element node are grouped together into a block stored contiguously in memory. For instance, the block matrix size is 6 for finite-element discretization of shells with 3 displacements and 3 rotations per node. Our BCSR implementation uses optimized routines for specific block sizes that are designed to increase the number of arithmetic operations for each memory access, while also minimizing the amount of logical control operations.

In order to reduce the number of fill-ins experienced in the factorization, we experimented with different orderings of the local matrix (3). A good ordering reduces the time required to factor the matrix $\mathbf{B}_i$, and compute the local Schur complement $\mathbf{S}_i$ using Equation (4). We use the approximate minimum degree (AMD) algorithm of Amestoy et al. (1996), the nested-disection (ND) algorithm from the METIS package (Karypis and Kumar, 1998) to order $\mathbf{B}_i$ and $\mathbf{C}_i$ independently, and our own implementation of AMD that enforces the requirement that the interface unknowns be ordered last and takes into consideration off-diagonal fill-ins. We call this modified version of AMD, AMD Off-Diagonal (AMD-OD).

3.1.2. Sparse block-cyclic factorization

In order to achieve good parallelism during the global Schur complement factorization and to distribute the memory required for storage, we employ a sparse 2D block-cyclic matrix storage format. In this format, the global Schur complement matrix (5), is split into a series of rectangular block matrices denoted $\mathbf{S}_{ij}$. The blocks are assigned to each processor based on a logical 2D process grid that is overlaid on the matrix. Only non-zero blocks are stored. Figure 1 illustrates the storage format with a sparse matrix and a 2×3 process grid for a 6 processor case. In our implementation, we store each block as a dense matrix in column-major ordering and use BLAS level 3 operations for all block-level operations required for the matrix factorization. Memory savings could be achieved by exploiting the sparsity within each block matrix. We have found, however, that the global Schur complement has many more entries than the local matrices (3), due to the large number of fill-ins produced during the local Schur complement computations. As a result, the global Schur complement is much more dense than the original matrix. The size of the block matrices are determined based on the matrix structure. However, there is a performance tradeoff between the communication latency and memory cache size when computing the matrix factorization (Li and Demmel, 2003).

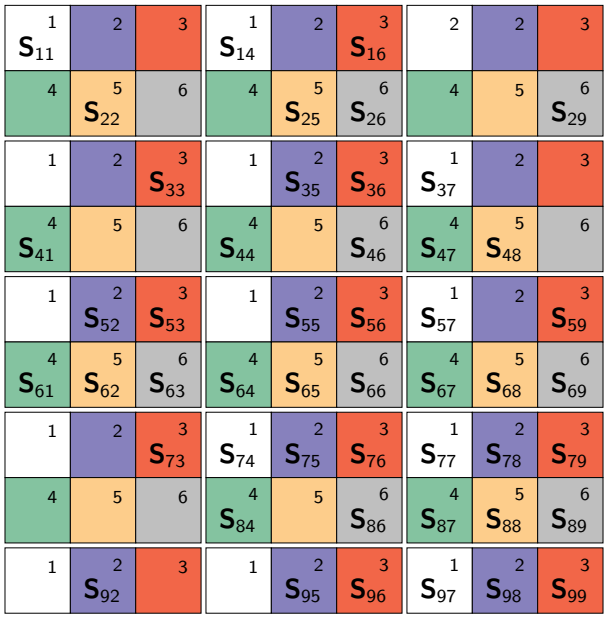


Figure 1: Illustration of the block-cyclic matrix format for a 2 $\times$ 3 grid.

An outline of the algorithm used to compute the LU-factorization for the block-cyclic matrix is show in Algorithm 1. Here, the function *is_block_owner*(i, j) returns true if the block matrix $\mathbf{S}_{ij}$ is non-zero and is assigned to the current processor, otherwise it returns false. The main loop of the algorithm consists of three main computational steps. First, the block diagonal is factored in place, $\mathbf{L}_{ii}\mathbf{U}_{ii} = \mathbf{S}_{ii}$. Second, row i and column i are updated by applying $\mathbf{L}_{ii}$ and $\mathbf{U}_{ii}$ to the blocks in row i and column i, respectively. Third, the trailing-matrix update must be applied to the remaining un-factored portion of the matrix. Note that the trailing-matrix update requires factored portions of the matrix that are not stored locally. Here we use a buffered approach where all required matrix components are buffered locally before being sent to the required processors.

The parallel performance of Algorithm 1 depends on the degree to which the column and row updates, as well as the trailing matrix update, can be parallelized. The row and column updates involve only a subset of the processors in column i and row i. As a result, these operations can only be distributed across these subsets of processors, resulting in sub-optimal parallel performance. However, these steps constitute a small portion of the computational time in the factorization. The trailing-matrix update, on the other hand, constitutes the main cost of the algorithm and utilizes all processors. This step is implemented efficiently in parallel.

3.2. Parallel direct factorization performance

We use a large finite-element model of a transport aircraft wing to demonstrate the performance of the domain-decomposition based parallel matrix factorization. This type of structure is currently the principal application area of TACS, so we focus on the performance for this case. The transonic aircraft wing is based on the geometry of a Boeing

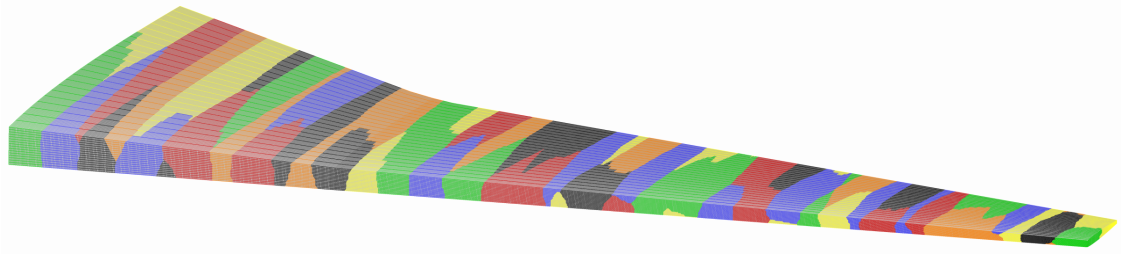
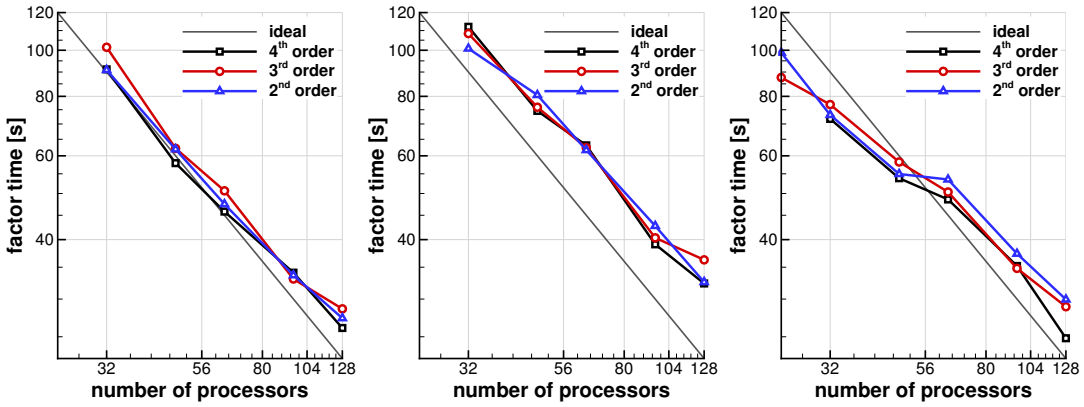
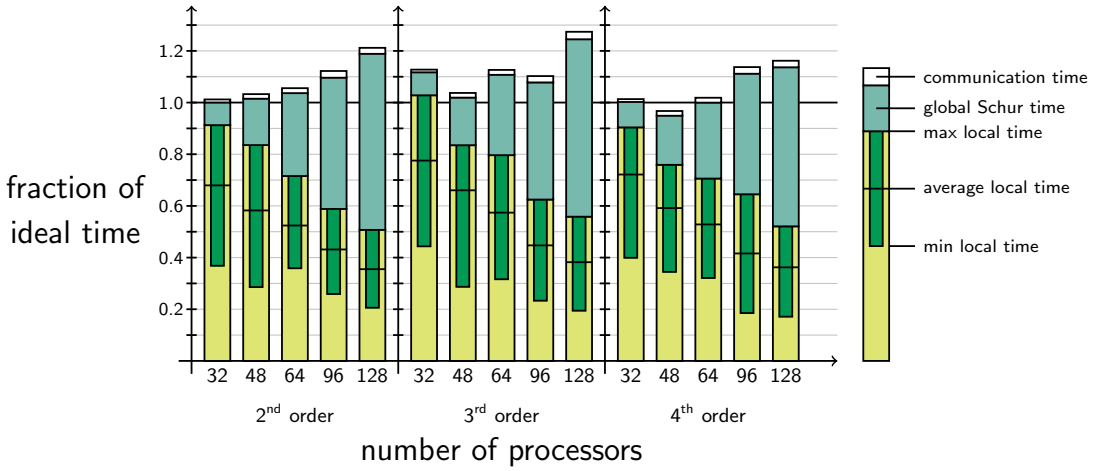


Figure 2: Domain decomposition for the transonic transonic wing on 64 processors.

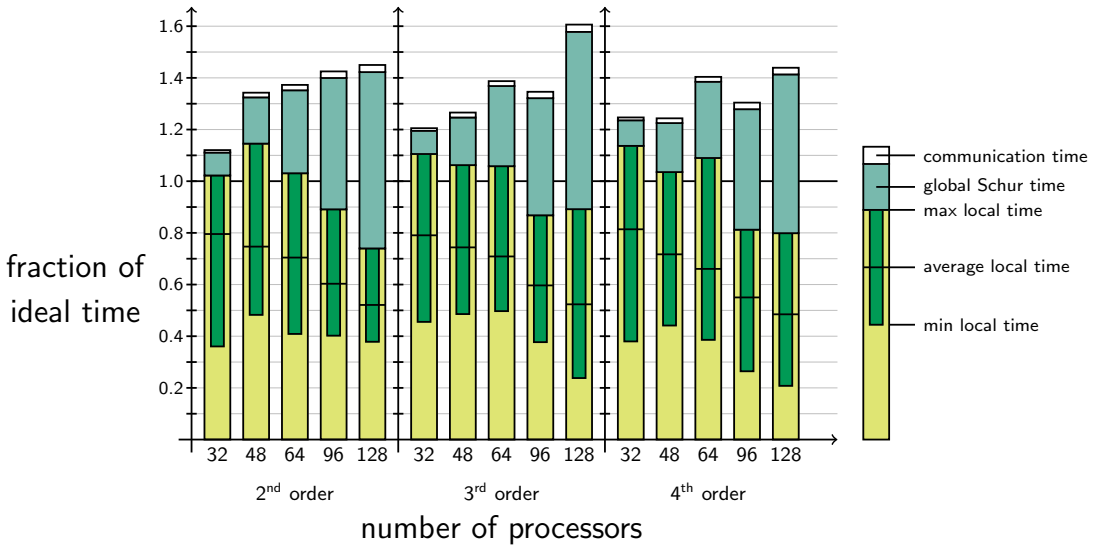


(a) ND (b) AMD (c) AMD-OD

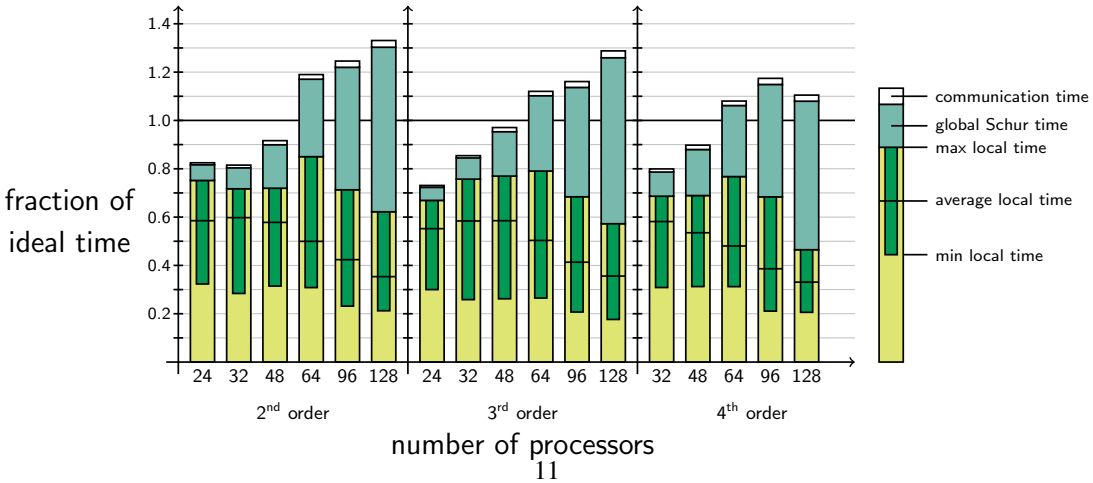
Figure 3: Total factorization time spent on 24, 32, 48, 64, 96 and 128 processors.



(a) ND



(b) AMD



(c) AMD-OD

Figure 4: Fraction of time spent in each step of the matrix factorization.

777-200ER wing. The wing has a semi-span of 30.45 m, a root chord of 13.6 m and a taper ratio of 0.2. The crank in the wing occurs at a station at 30% of the semi-span. The wing structure consists of 44 chord-wise ribs spaced evenly out the wing and two span-wise spars. The wing is modeled using either an isotropic or composite material shown in Table 1. For simplicity, the skin thickness of the wing is set to 5 mm uniformly over the entire structure.

The wing is discretized using either second, third, or fourth order MITC shell elements with 907 388 nodes resulting in just over 5.44 million degrees of freedom. The second order discretization contains 912 384 elements, the third order discretization contains 228 096 elements, and the fourth order discretization contains 101 376 elements. The wing is loaded with a set of aerodynamic forces computed at a 2.5 g maneuver flight condition. Figure 2 shows the domain decomposition corresponding to a 64 processor case where each contiguously colored segment corresponds to a domain belonging to a different processor.

The results in this section are based on calculations performed on the General Purpose Cluster (GPC) at SciNet (Loken et al., 2010). Each node of the GPC consists of dual Intel Xeon E5540 processors with a clock speeds of 2.53 GHz, with 16 GB of dedicated RAM and 8 processor cores. In these comparisons, we only use nodes connected with non-blocking 4x-DDR InfiniBand.

Figure 3 shows the factorization times for the finite-element wing model using AMD, ND and AMD-OD orderings for 24, 32, 48, 64, 96 and 128 processors for the second, third, and fourth order problems. Due to memory constraints, the problem must be run on at least 32 processors for most orderings. Note that the ordering has the largest impact on the factorization times and that there is significantly less variation between the factor times for the second, third, and fourth order problems. For the AMD-OD ordering, the computational time actually decreases slightly, in some cases, with increasing order. Both AMD and AMD-OD ordering methods are very effective at taking advantage of the matrix structure that exists in the higher-order problems through super-node identification (Amestoy et al., 1996). These super nodes help produce re-ordered matrices that are faster to factorize.

From Figure 3 it is clear that the AMD-OD ordering scheme results in the fastest factorization times for the 24, 32, and 48 processor cases. However, the AMD-OD factorization times do not scale as well as ND or AMD. The AMD-OD ordering is effective at reducing the off-diagonal fill-ins, but these fill-ins have the greatest impact when there are fewer processors and the off-diagonal matrices are large. As a result, AMD-OD tends to perform better for fewer numbers of processors, and for larger numbers of processors tends to perform as well or slightly better than AMD. In all cases, the ND ordering seems to scale the most consistently.

The fraction of time spent in each stage of the direct factorization change significantly as the number of processors is increased. Figure 4 presents a detailed study of the fraction of time spent in different factorization operations normalized to the ideal line presented in Figure 3. Figure 4 shows the fractions of time for the processors with the least idle time, the average idle time and the most idle time. These correspond to the processors that require the maximum local time, average local time and minimum local time, respectively. The local factorization time corresponds to the time to compute the block factorization of $\mathbf{B}_i$, and the local contribution to the Schur complement $\mathbf{S}_i$ from Equation (4). The communication time is the time required to communicate the local Schur complement

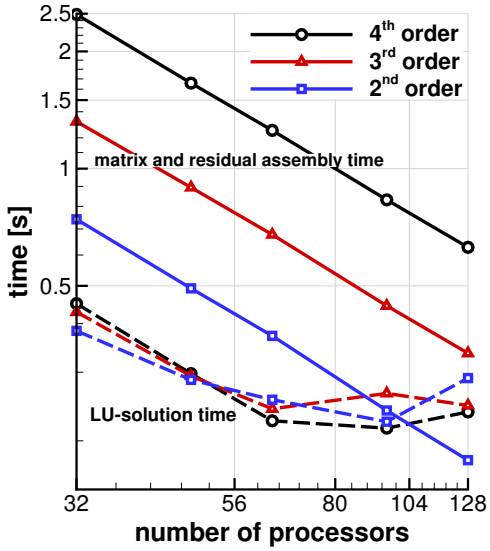


Figure 5: Assembly and LU-solution times for the second, third, and fourth order elements. Assembly times scale ideally while LU-solution times do not scale well beyond 64 processors.

contributions to the required processors for the global Schur complement. Finally, the global Schur complement time is the time required to factor the global Schur complement in the block-cyclic data format. Note that the proportion of time spent in each stage of the factorization changes as the number of processors increases. In particular, the fraction of time to factor the global Schur complement increases. This behavior makes it difficult to obtain an ideal speed up consistently.

The efficiency of the ordering techniques can be judged from Figure 4 based on the discrepancy between the processors with the maximum local time and average local time. Large gaps arise when one processor takes significantly longer than any of the others. Note that this gap is smallest for the AMD-OD ordering for the 24, 32 and 48 processor cases, but increases between 48 and 64 processors. For the ND ordering, this gap does not grow as rapidly as either AMD or AMD-OD with increasing numbers of processors, while the discrepancy between the maximum and minimum local times for the ND ordering becomes smaller. The gap between maximum and average local times is largest for the AMD ordering, but does not increase significantly.

3.3. Assembly and LU-solution performance

The matrix and residual assembly and LU-solution operations require an order of magnitude less computational time than the matrix factorization and are typically less performance-critical tasks. Nevertheless, it is important that these operations be as scalable as possible to enable the solution of large finite-element problems. Furthermore, we demonstrate in Section 5 that the LU solution times are critical for certain gradient-evaluation operations. Figure 5 shows the matrix and residual assembly times as well as the LU solution time for the 32, 48, 64, 96, and 128 processor

cases with ND ordering. Note that we restrict the results to a single ordering scheme, since the matrix and residual assembly are insensitive to the ordering of the unknowns, while the LU solution times for all other ordering schemes exhibit similar trends. The matrix and residual assembly of all orders scale ideally to plotting precision, with the third and fourth order elements taking roughly 1.8 and 3.4 times as long as the second order elements, respectively. The LU solution performance does not scale ideally and the relative performance varies with element order. While the solution scales moderately well between 32 and 64 processors, the efficiency decreases significantly for the 96 and 128 processor cases. In addition, the performance between element orders is not consistent. This behavior is due to the difficulty of implementing upper and lower triangular solution methods efficiently in parallel where the ratio of communication to computation operations is much higher than for the matrix factorization. While the code does not exhibit the desired ideal speed up behavior, a single LU solution for a system with 5.44 million degrees of freedom requires roughly 0.2 to 0.3 seconds on anywhere between 64 and 128 processors.

4. Functions of interest

Once the solution of the structural problem is obtained, it is necessary to evaluate the functions of interest required for the optimization problem (1). In this section, we present the formulation of the functions of interest including the structural mass, material failure constraints, and buckling constraints. We then present a verification of both the solution and the p-norm functional for a free cylinder test case problem. These verification cases ensure that both the finite-element implementation and the implementation of the functional is correct.

4.1. Structural mass

The structural mass is simply evaluated as a summation of the density per unit area integrated with Gauss quadrature over each element. The total mass is obtained by summing the mass contributions from each element in the entire finite-element mesh.

4.2. Failure constraints

Stress or failure constraints are imposed to limit the stress or strain within the structure to a permissible envelope defined by material allowables. We prefer the term failure constraint since either the stress or the strain may be restricted depending on the criterion selected by the designer. The permissible envelope will typically vary over the domain and depend on the structural design variables such as the thickness or lamination stacking sequence.

In this work, we present an approach to limit the point-wise failure function written as follows:

$$F(\mathbf{x}_M, \boldsymbol{\epsilon}) \leq 1, \tag{7}$$

where $F(\mathbf{x}_M, \boldsymbol{\epsilon}) \in \mathbb{R}$ is a scalar valued function that depends on some material characterization data, the material design variables, $\mathbf{x}_M$, and the local strain values, $\boldsymbol{\epsilon} = \boldsymbol{\epsilon}(\mathbf{X}^e, \mathbf{u}^e)$. Note that the failure function depends indirectly on the geometric design variables through the strain that is computed as a function of the element displacements and element

nodal locations. Many classical failure criteria can be written in the form of Equation (7). For instance, the von Mises failure criterion can be written as follows:

$$F_{vM}(\mathbf{x}_M, \boldsymbol{\epsilon}) = \frac{\sigma_{vM}}{\sigma_{ys}} = \frac{1}{\sigma_{ys}} \sqrt{\sigma_x^2 + \sigma_y^2 - \sigma_x\sigma_y + 3\sigma_{xy}^2}, \tag{8}$$

where σ_{ys} is some specified yield stress. The Tsai–Wu failure criterion for composite materials can be written as follows:

$$F_{TW}(\mathbf{x}_M, \boldsymbol{\epsilon}) = F_1\sigma_1 + F_2\sigma_2 + F_{11}\sigma_1^2 + 2F_{12}\sigma_1\sigma_2 + F_{22}\sigma_2^2 + F_{66}\sigma_{12}^2, \tag{9}$$

where the coefficients F_1, F_2, F_{11}, F_{12}, F_{22} and F_{66} are a function of the failure stress in the orthotropic material axis (Jones, 1996). Other failure criteria, such as the maximum stress or maximum strain criterion, can be written as a non-smooth function of the stress or strain.

4.3. Buckling constraints

We evaluate a point-wise local buckling constraint in a similar manner to the failure constraint presented above. In this work, we assume that buckling can be predicted at any point within the structure based on a single buckling envelope written here as:

$$B(\mathbf{x}_M, \boldsymbol{\epsilon}) \leq 1, \tag{10}$$

where $\mathbf{x}_M$ are the material design variables and $\boldsymbol{\epsilon}$ are the point-wise strains.

In practice, we construct the buckling envelope (10) for each component susceptible to buckling within the structure. For wings, we compute the buckling envelope in the panels formed by the intersections of the ribs and spars, as well as the spars themselves. Finally, we assume that the interaction between the longitudinal and shear buckling modes collapses into the following buckling envelope:

$$B(\mathbf{x}_M, \boldsymbol{\epsilon}) = B_1\sigma_1 + B_2\sigma_{12} + B_{22}\sigma_{12}^2 \leq 1, \tag{11}$$

where σ_1, σ_{12} are the axial and shear stress in a locally-aligned reference frame with the panel. The parameters B_1, B_2 and B_{22} in Equation (11), are given by:

$$\begin{aligned} B_1 &= -\frac{1}{\sigma_{1,cr}}, \\ B_2 &= \frac{\sigma_{12,cr}^+ - \sigma_{12,cr}^-}{\sigma_{12,cr}^+\sigma_{12,cr}^-}, \\ B_{22} &= \frac{1}{\sigma_{12,cr}^+\sigma_{12,cr}^-}, \end{aligned}$$

where, $\sigma_{1,cr}$ is the compressive buckling load of the panel and $\sigma_{12,cr}^+$ and $\sigma_{12,cr}^-$ are the positive and negative shear buckling loads.

4.4. Failure and buckling constraint aggregation

In many structural optimization formulations, the failure and buckling constraints are imposed only at a set of discrete points within the domain, such as the quadrature points, or other parametric trial points within each element (Poon and Martins, 2007). When the trial points are selected for each element, the number of constraints increases in direct proportion to the number of elements in the finite-element model. To reduce the number of constraints for gradient-based design optimization, a number of authors have developed aggregation techniques that combine sets of the failure constraints into a single equivalent constraint in a conservative manner (Akgun et al., 2001). In this section, we examine two failure constraint aggregation techniques: the Kreisselmeier–Steinhauser (KS) aggregation function and the p-norm aggregation technique.

The KS function was first employed in an optimization setting by Wrenn (1989), who used it to convert a constrained optimization problem into an unconstrained problem. Other authors have used the KS function to impose structural constraints in various applications including structural sizing problems (Akgun et al., 2001; Poon and Martins, 2007), topology optimization (Pereira et al., 2004; Pars et al., 2009; Le et al., 2010), and aerostructural design optimization (Martins et al., 2004; Kenway et al., 2013). The discrete KS function can be written as follows:

$$\mathrm{KS}_d(\mathbf{x}, \mathbf{u}) = \frac{1}{\rho} \ln \left[\sum_{i=1}^{M} e^{\rho F(\mathbf{x}_M, \boldsymbol{\epsilon}_i)} \right] \tag{12}$$

where ρ is a fixed parameter value and $\boldsymbol{\epsilon}_i$ is the strain at the i^{th} trial point. As $\rho \to \infty$, the KS approximation becomes more accurate such that in the limit, $\mathrm{KS}_d \to \max_i \{F(\mathbf{x}_M, \boldsymbol{\epsilon}_i)\}$. However large values of ρ lead to poorly conditioned optimization problems (Poon and Martins, 2007) so there is a tradeoff between accuracy and poor conditioning. As a result, the optimal value of the aggregation parameter is difficult to determine *a priori* so Poon and Martins (2007) developed an approach to adaptively select the parameter to achieve a tighter-bound to the original constraints.

Another constraint aggregation technique, related to the KS function, is the discrete p-norm. This approach is mostly employed in topology optimization problems (Le et al., 2010). The discrete p-norm function can be written as follows:

$$\mathrm{PN}_d(\mathbf{x}, \mathbf{u}) = \left[\sum_{i=1}^{M} |F(\mathbf{x}_M, \boldsymbol{\epsilon}_i)|^p \right]^{\frac{1}{p}} \tag{13}$$

where $p > 1$, is a parameter such that as $p \to \infty$, the p-norm approximation becomes more accurate such that in the limit, $\mathrm{PN}_d \to \max_i \{F(\mathbf{x}_M, \boldsymbol{\epsilon}_i)\}$. Like the KS function, large values of p lead to poorly conditioned optimization problems.

One issue with the discrete KS function and the discrete p-norm is that they do not converge asymptotically as the mesh is refined for fixed values of ρ and p. This is due to the discrete nature of the functions which use quadrature points or other discrete sets of trial points whose number and location vary depending on the discretization. Therefore, if the discrete aggregation approaches are used as constraints within an optimization problem, the optimal solution will also exhibit mesh dependence. As a result, we prefer either the continuous p-norm or KS functional. The KS functional was first introduced by Akgun et al. (2001). The KS and p-norm functionals are closely related to their

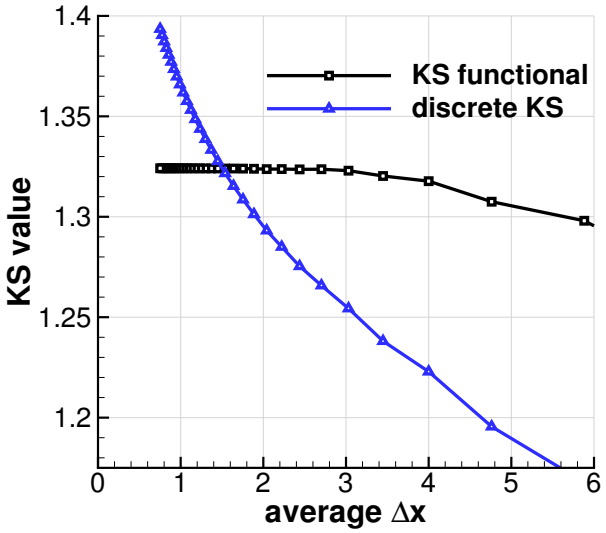


Figure 6: Comparison of the convergence behavior of the discrete KS function and the KS functional for a cylinder test problem. The discrete KS function exhibits mesh dependence, while the KS functional does not.

discrete counterparts but exhibit asymptotic convergence behavior since they employ integrals over the structural domain rather than discrete sums. For comparison, the convergence behavior of the discrete KS function and the KS functional are shown in Figure 6 with fixed ρ for a series of meshes for the isotropic cylinder problem described in Section 4.5. The discrete KS function does not display asymptotic convergence behavior.

The p-norm aggregation of the failure criterion can be written as follows:

$$\begin{aligned} \mathrm{PN}(\mathbf{x},\, \mathbf{u};\, p) = \|F(\mathbf{x}_M, \boldsymbol{\epsilon})\|_p &= \left(\int_\Omega |F(\mathbf{x}_M, \boldsymbol{\epsilon})|^p \, d\Omega \right)^{\frac{1}{p}}, \\ &= |F(\mathbf{x}_M, \boldsymbol{\epsilon}_m)| \left(\int_\Omega \left| \frac{F(\mathbf{x}_M, \boldsymbol{\epsilon})}{F(\mathbf{x}_M, \boldsymbol{\epsilon}_m)} \right|^p \, d\Omega \right)^{\frac{1}{p}}, \end{aligned} \tag{14}$$

where $p > 1$. When evaluating the p-norm, we use latter expression which is mathematically-equivalent to the former, but is less susceptible to numerical overflow. Here $\boldsymbol{\epsilon}_m$ is the strain that produces the maximum value of the failure criterion amongst all quadrature points within the domain of integration. This ensures that the integrand is only evaluated at points where it is bounded by unity. We evaluate the integral in Equation (14) approximately using the same order quadrature scheme used by the finite-element implementation. The p-norm functional has the property that $\mathrm{PN}(\mathbf{x}, \mathbf{u}; p) \to \|F\|_\infty$ as $p \to \infty$.

The KS functional is closely related to the p-norm and can be written as follows:

$$\begin{aligned} \mathrm{KS}(\mathbf{x},\, \mathbf{u};\, \rho) &= \frac{1}{\rho} \ln \left(\int_\Omega e^{\rho F(\boldsymbol{\sigma})} \, d\Omega \right) \\ &= F(\boldsymbol{\sigma}_m) + \frac{1}{\rho} \ln \left(\int_\Omega e^{\rho (F(\boldsymbol{\sigma}) - F(\boldsymbol{\sigma}_m))} \, d\Omega \right) \end{aligned} \tag{15}$$

where again $\boldsymbol{\epsilon}_m$ is the strain the produces that maximum failure criterion amongst all quadrature points within the domain. Again, in practice we approximate the integral in Equation (15) using the same quadrature scheme employed by each element in the domain. In a similar manner to the p-norm, the KS functional approaches an upper bound on the failure function: $KS(\mathbf{x}, \mathbf{u}; \rho) \to \|F\|_\infty$ as $\rho \to \infty$.

In theory, a single aggregation functional could be used to enforce the failure or buckling constraint over the entire structure, however, this often leads to highly-nonlinear constraints which leads to poor optimization performance (Poon and Martins, 2007). Instead, we use a series of n_d independent aggregation domains, Ω_i, with $i = 1, \ldots, n_d$, and aggregate the failure or buckling criterion separately over each domain. These sub-domains satisfy the following properties:

$$\begin{aligned} \bigcup_{i=1}^{n_d} \Omega_i &= \Omega, \\ \Omega_i \cap \Omega_j &= \emptyset \qquad \forall\, i \neq j, \end{aligned} \tag{16}$$

Instead of using a random domain assignment or domains based on the parallel domain decomposition, we use aggregation domains that share a common structural purpose or component. For instance, in the case of a wing-box optimization, we often form aggregation domains from all or part of the top skin, bottom skin, ribs, spars and stiffeners, independently. This ensures that the design variables that directly modify the structure are all scaled in a similar manner and that the constraints have a physical significance that is simple understand (e.g., the failure constraint for the top skin).

4.5. Solution and functional accuracy

In this section, we present a verification of both the solution and functional accuracy for our implementation. These tests demonstrate the accuracy of the second, third, and fourth-order shell elements for the analysis of both composite and isotropic materials. In addition, we demonstrate the accuracy of the functional estimates of the p-norm by comparison with the p-norm of the exact solution for both the von Mises and Tsai–Wu failure criteria. In addition to these tests, we have also verified that the elements pass the classical MacNeal–Harder tests (MacNeal and Harder, 1985) for shell elements, but do not reproduce those results here.

In this study, we use an exact solution of a specially orthotropic circular cylindrical shell that is subject to a distributed pressure load. An outline of the derivation of the exact solution as well as the exact value of the p-norm functional for both an isotropic and a specially orthotropic cylinder are presented in Appendix A. The dimensions of the cylinder are set such that the length of the cylinder is $L = 100$, the radius is $a = 100/\pi$, and the thickness is $t = 1$. The cylinder is subject to a distributed pressure load as follows:

$$p(\theta, z) = p_0 \sin\left(\frac{4\theta}{R}\right) \sin\left(\frac{3\pi z}{L}\right), \tag{17}$$

where z is the axial direction and the magnitude of the load, p_0, will defined below.

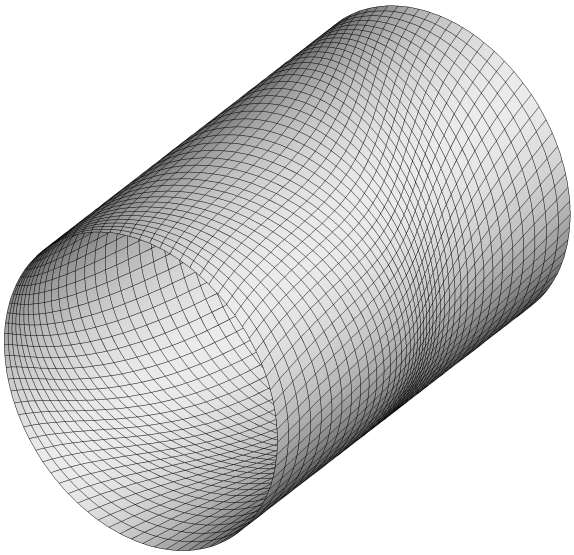


Figure 7: Distorted finite-element mesh used in the analysis of the cylinders with $m = 40$ axial elements.

Instead of using a finite-element mesh aligned with the directions of principle curvature, we analyze the full cylinder and use a distorted mesh parametrized as follows:

$$\begin{aligned} z &= L\xi_1 + \frac{L}{10}\sin(2\pi\xi_1)\cos(2\pi\xi_2), \\ \theta &= 2\pi\xi_2 + \frac{\pi}{4}\xi_1 + \frac{1}{10}\cos(2\pi\xi_1)\sin(2\pi\xi_2), \end{aligned} \tag{18}$$

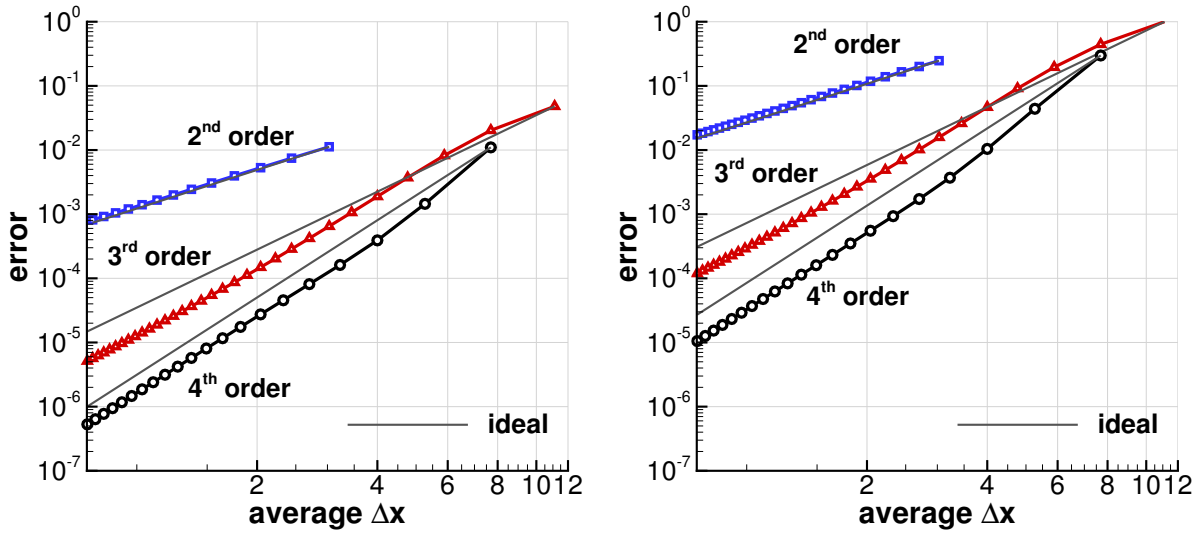
where $(\xi_1, \xi_2) \in [0, 1]^2$. The mesh for the cylinder under consideration is shown in Figure 7, and consists of m elements in the axial direction and $2m$ elements in the circumferential direction. These dimensions are selected to ensure that the aspect ratio of the elements is roughly unity.

In the context of a structural optimization problem, the failure function should be bounded by 1 everywhere such that the structure can sustain the applied load without a material failure. To mimic these conditions, we select p_0 such that the maximum value of the failure criterion is unity, i.e. $\|F\|_\infty = 1$. In order to obtain the value of p_0, we first determine the solution for a unit load and then calculate the value of p_0 required to obtain $\|F\|_\infty = 1$. In the cases presented here, the maximum von Mises failure stress is $\|\sigma_{vM}\|_\infty = \sqrt{a_{mn}^2 + b_{mn}^2 - a_{mn}b_{mn}}$, while the maximum Tsai–Wu criterion is $\|F_{TW}\|_\infty = f_{mn} + g_{mn}$, where the coefficients a_{mn}, b_{mn}, f_{mn}, and g_{mn} are defined in the Appendix. As a result, for the case of the isotropic cylinder, the required value of p_0 is:

$$p_0 = \frac{1}{\sqrt{a_{mn}^2 + b_{mn}^2 - a_{mn}b_{mn}}},$$

while for the case of the specially orthotropic cylinder, p_0 is obtained by obtaining the solution to the following quadratic equation:

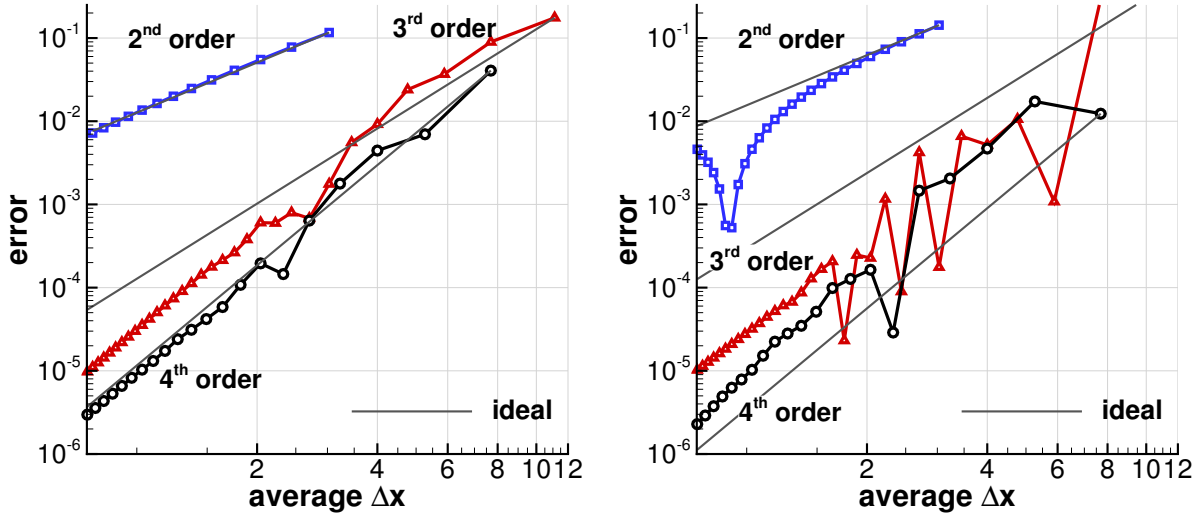
$$p_0^2 g_{mn} + p_0 f_{mn} - 1 = 0.$$



(a) Isotropic

(b) Specially orthotropic

Figure 8: Solution accuracy of the isotropic and specially orthotropic cylinders. All elements exhibit the expected solution accuracy.



(a) Isotropic

(b) Specially orthotropic

Figure 9: Functional p-norm accuracy for the isotropic and specially orthotropic cylinders. All elements exhibit the expected accuracy asymptotically.

Figure 8 shows the solution accuracy for the isotropic and specially orthotropic cylinders for the second, third, and fourth order elements. All elements achieve their expected accuracy. Figure 9 shows absolute error between the p-norm functional and the functional estimate obtained from the finite-element discretization for both the isotropic cylinder with the von Mises stress, and the specially orthotropic cylinder for the Tsai–Wu failure criterion, respectively. For the second order meshes we use $m = 32$ to $m = 128$ elements in increments of 4, for the third order meshes we use $m = 4$ to $m = 66$ in increments of 2 and for the fourth order meshes we use $m = 4$ to $m = 44$ in increments of 2. The finite-element mesh is distorted so we plot the solution error and functional estimate error against the average mesh spacing $\Delta x = L/(m + 1)$, $\Delta x = L/(2m + 1)$ and $\Delta x = L/(3m + 1)$. The solution error is estimated using the $\mathcal{L}_2$ norm of the displacement normal to the surface as follows:

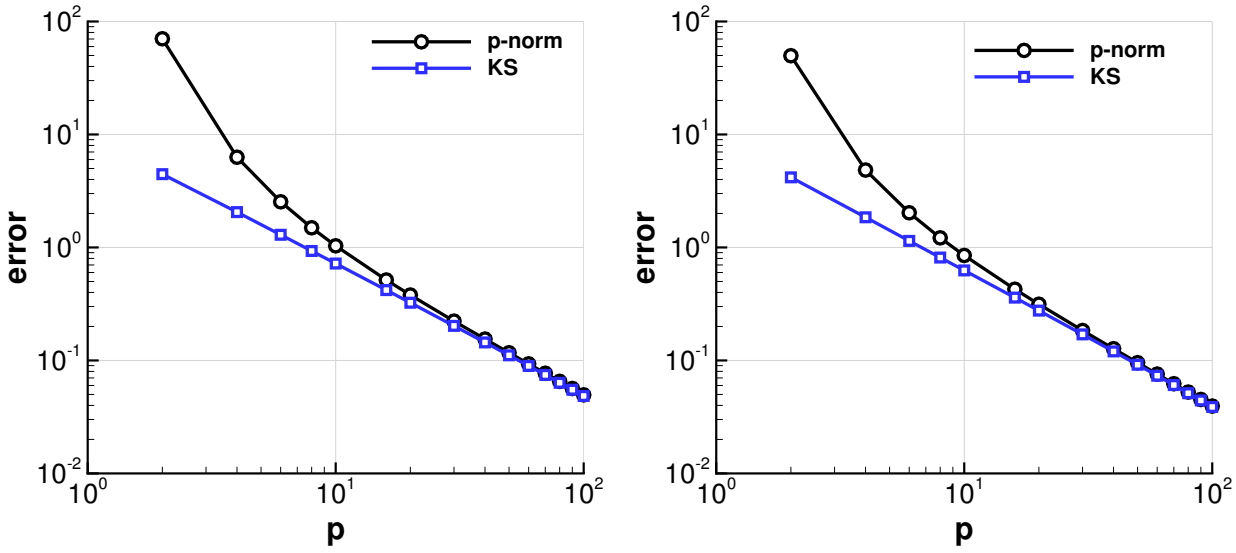
$$\|w - w_h\|_2 = \left(\int_\Omega (w - w_h)^2 \, d\Omega \right)^{\frac{1}{2}},$$

where w is the normal displacement. This error estimate is obtained by integrating the square of the error over the finite-element mesh utilizing a Gauss quadrature scheme one order higher than the order of the finite-element. From Figure 8, it is clear that all elements achieve the expected accuracy for both the isotropic and specially orthotropic cylinders. The relative offset between the errors is due to the normalization of the applied load, resulting in a higher load and larger displacement for the specially orthotropic cylinder. The convergence behavior of the p-norm functional estimates, shown in Figure 9, are not as smooth as the solution error. For the isotropic cylinder, the p-norm estimate error does not behave smoothly until the average discretization length, $\Delta x/L$, decreases below 2% for the third and fourth order meshes. For the specially orthotropic cylinder, the second order results do not enter a region of asymptotic convergence, within the range of $\Delta x/L$ shown here, while the third and fourth order solutions behave smoothly below an average discretization length of $\Delta x/L < 1.75\%$.

These results show that the p-norm estimate obtained from the finite-element solution converges to the p-norm of the exact solution. However, it is also important to assess how well the p-norm and KS functional estimates approximate the upper bound of the failure criterion. To address this issue, Figure 10 shows the difference between the p-norm and KS functional estimates for the third-order meshes with $m = 66$ for increasing values of the p and ρ parameters for both the isotropic and specially orthotropic cylinders. Both cases exhibit almost identical behavior: the KS functional estimate provides a tighter bound for the maximum value of the failure criterion, especially for low values of the parameters p and ρ. As discussed previously, large penalty parameters are often undesirable in optimization applications as these tend to increase the required optimization iterations (Poon and Martins, 2007). As a result, we prefer the KS functional to the p-norm in structural optimization applications.

5. Gradient evaluation

Gradient-based design optimization methods can only solve large-scale design problems in a reasonable computational time if both the analysis and gradient evaluation are performed efficiently. Furthermore, for a fixed computational budget, improving the performance of the gradient evaluation gives the user additional flexibility to modify



(a) Isotropic

(b) Specially orthotropic

Figure 10: Error between the upper bound and the p-norm and KS functional estimate of the upper bound for the isotropic and specially orthotropic cylinders. The KS functional provides a tighter bound.

the design problem by adding additional constraints. Therefore, we seek an approach to evaluate the gradient whose computational time scales as weakly as possible with both the number of functions of interest and the number of design variables, and parallelizes well with increasing number of processors. As we will show, we must compromise by accepting a higher scaling factor for either increasing numbers of functions, or increasing numbers of design variables.

Within the reduced space paradigm, we must evaluate the gradient of any function of interest while taking into account the effect of satisfying the governing equations. This means that we require the total derivative derivative of a vector of functions $\nabla_{\mathbf{x}}\mathbf{f}_i \in \mathbb{R}^{n_f \times n_x}$ which can be evaluated as follows:

$$\nabla_{\mathbf{x}}\mathbf{f}_i = \frac{\partial \mathbf{f}_i}{\partial \mathbf{x}} - \frac{\partial \mathbf{f}_i}{\partial \mathbf{u}}\left[\frac{\partial \mathbf{R}_i}{\partial \mathbf{u}}\right]^{-1}\frac{\partial \mathbf{R}_i}{\partial \mathbf{x}}. \tag{19}$$

There are two methods that may be used to evaluate the gradient: the adjoint method or the direct method (Martins and Hwang, 2013). The adjoint sensitivity method can be obtained by introducing the matrix of adjoint variables $\boldsymbol{\psi}$, such that

$$\left[\frac{\partial \mathbf{R}_i}{\partial \mathbf{u}}\right]^T \boldsymbol{\psi}_i = \frac{\partial \mathbf{f}_i}{\partial \mathbf{u}}^T. \tag{20}$$

Once the adjoint variables $\boldsymbol{\psi}$ are determined, the total derivative (19) reduces to:

$$\nabla_{\mathbf{x}}\mathbf{f}_i = \frac{\partial \mathbf{f}_i}{\partial \mathbf{x}} - \boldsymbol{\psi}_i^T\frac{\partial \mathbf{R}_i}{\partial \mathbf{x}}. \tag{21}$$

The direct sensitivity method can be obtained by introducing the matrix of state-variable derivatives $\boldsymbol{\phi}$ such that

$$\left[\frac{\partial \mathbf{R}_i}{\partial \mathbf{u}}\right]\boldsymbol{\phi}_i = \frac{\partial \mathbf{R}_i}{\partial \mathbf{x}}. \tag{22}$$

Once the state-variable derivatives $\boldsymbol{\phi}$ are determined, total derivative can be evaluated as follows:

$$\nabla_{\mathbf{x}}\mathbf{f}_i = \frac{\partial \mathbf{f}_i}{\partial \mathbf{x}} - \frac{\partial \mathbf{f}_i}{\partial \mathbf{u}}\boldsymbol{\phi}_i. \tag{23}$$

It is important to understand the relative computational time and memory costs of both the adjoint and direct methods. Both methods require mathematically equivalent terms, however, the sequence of computations and storage of the terms also strongly impacts the relative memory costs.

As many authors have identified, for instance (Martins and Hwang, 2013), the most significant difference between the methods, in terms of computational cost, is the relative number of functions and design variables. If there are more design variables than functions $n_x > n_f$, then the adjoint method requires fewer adjoint solutions (20) than the direct method (22). As a result, the adjoint method requires less computational time. If there are more functions than design variables $n_f > n_x$, then the direct method requires less computational time. However, if the number of functions and design variables are both large, then both methods require significant computational time. By using the constraint aggregation techniques presented in Section 4.4, we ensure that the number of functions per load case can be reduced to a manageable number. Therefore, we focus on the adjoint method. In the following section, we describe how we implemented the adjoint in a way that seeks to minimize the computational costs when there are large numbers of geometric and material design variables. We then evaluate the computational performance and scalability of our implementation.

5.1. Adjoint implementation

Within our framework, we treat the design variables as a global vector, rather than a distributed vector, in the sense that any finite-element on any processor may access any design variable. Therefore, the design variable vector is duplicated on all processors. For the gradient implementation, we likewise store the full gradient, $\nabla_{\mathbf{x}}\mathbf{f}_i$, on all processors. Both the adjoint and direct methods require the partial derivatives of the functions of interest with respect to the design variables, $\partial \mathbf{f}_i / \partial \mathbf{x}$. We compute this term by adding the contributions from the material and geometric design variables that are evaluated using two separate techniques. For the material design variables, we compute the derivative of the function with respect to all material design variables over each element and sum the results on all processors. The implementation for the geometric design variables is more complex. First, we compute the derivative of the functions with respect to all nodes. Next, we multiply this term by the derivative of the nodes with respect to the geometric design variables. This calculation can be written as follows:

$$\frac{\partial \mathbf{f}_i}{\partial \mathbf{x}_G} = \left[\sum_{j=1}^{N_e} \frac{\partial \mathbf{f}_i}{\partial \mathbf{X}_j^e} \left(\mathbf{P}_j^T \otimes \mathbf{I}_3 \right) \right] \frac{\partial \mathbf{X}^N}{\partial \mathbf{x}_G}. \tag{24}$$

As with all gradient operations, we sum the contributions locally on each processor and, after each processor has finished, sum the results across all processors.

The first step in the adjoint method is to obtain the adjoint variables $\boldsymbol{\psi}_i$ by evaluating the gradients $\partial \mathbf{f}_i / \partial \mathbf{u}$ for $j = 1, \ldots, n_f$ and solving Equation (20). We compute the derivatve, $\partial \mathbf{f}_i / \partial \mathbf{u}$, by evaluating the contribution from each

element, and assemblying the global vector as follows:

$$\frac{\partial \mathbf{f}_i}{\partial \mathbf{u}} = \sum_{j=1}^{N_e} \frac{\partial \mathbf{f}_i}{\partial \mathbf{u}_j^e} \left(\mathbf{P}_j^T \otimes \mathbf{I}_6 \right). \tag{25}$$

The load balancing during the computation of the terms in Equations (24) and (25) depends on the size and distribution of the aggregation domains for the functions $\mathbf{f}_i$. Since the union of all the aggregation domains is the entire structure, this calculation tends to scale well, as long as the aggregation domains are approximately of equal size.

To complete the gradient calculation, we must multiply the adjoint variables $\boldsymbol{\psi}_i$ by the derivative of the residuals with respect to the design variables. One approach to this calculation would be to compute the derivative $\partial \mathbf{R}_i / \partial x_i$, for $i = 1, \ldots, n_x$ one column vector at a time and then compute the inner product with each adjoint vector. However, this approach ignores the sparsity pattern of $\partial \mathbf{R}_i / \partial \mathbf{x}$, and would be equivalent to a dense matrix-vector product. Instead, we have implemented a matrix-free method that computes the matrix-vector product $\boldsymbol{\psi}_i^T \left[\partial \mathbf{R}_i / \partial \mathbf{x} \right]$ directly and exploits sparsity. As before, we use separate calculations for the geometric and material design variable contributions. We compute the contribution from the geometric design variables as follows:

$$\boldsymbol{\psi}_i^T \frac{\partial \mathbf{R}_i}{\partial \mathbf{x}_G} = \boldsymbol{\psi}_i^T \frac{\partial \mathbf{R}_i}{\partial \mathbf{X}^N} \frac{\partial \mathbf{X}^N}{\partial \mathbf{x}_G} = \left[\sum_{j=1}^{N_e} \left(\boldsymbol{\psi}_i^T \left(\mathbf{P}_j \otimes \mathbf{I}_6 \right) \right) \frac{\partial \mathbf{R}_j^e}{\partial \mathbf{X}_j^e} \left(\mathbf{P}_j^T \otimes \mathbf{I}_3 \right) \right] \frac{\partial \mathbf{X}^N}{\partial \mathbf{x}_G}. \tag{26}$$

This term is calculated in three stages. First, we compute the term in Equation (26) between the square brackets independently on each processor by evaluating the derivative, $\partial \mathbf{R}_j^e / \partial \mathbf{X}_j^e$ for each element, and then take a local matrix-matrix product with all adjoint variables corresponding to the local element. The result from each element-wise calculation is accumulated locally without communication on each processor in a temporary array. Next, after all element contributions have been added locally, we evaluate the product of the term within the square brackets with the derivative of the nodal locations with respect to the geometric design variables, $\partial \mathbf{X}^N / \partial \mathbf{x}_G$. Finally, we perform a global reduction by adding the local contribution from each processor to a global array. The calculation of the term $\boldsymbol{\psi}_i^T \left[\partial \mathbf{R}_i / \partial \mathbf{x} \right]$, is much simpler for the material design variables. This contribution can be written as follows:

$$\boldsymbol{\psi}_i^T \frac{\partial \mathbf{R}_i}{\partial \mathbf{x}_M} = \sum_{j=1}^{N_e} \left(\boldsymbol{\psi}_i^T \left(\mathbf{P}_j \otimes \mathbf{I}_6 \right) \right) \frac{\partial \mathbf{R}_j^e}{\partial \mathbf{x}_M}, \tag{27}$$

where we sum the contribution from each element on each processor and then perform a global reduction by adding the local contribution from each processor to a global array.

5.2. Gradient accuracy verification

In this section, we verify the accuracy of the gradient calculations using the complex-step method (Squire and Trapp, 1998; Martins et al., 2003). The complex-step method can be used to obtain high-accuracy derivate approximations, since this technique is not susceptible to subtractive cancellation. To the projected derivative with the complex-step method, we evaluate the imaginary part of the function that is perturbed by a small complex step:

$$\mathbf{p}^T \nabla_{\mathbf{x}} f = \frac{\mathrm{Im}(f(\mathbf{x} + i\, h\mathbf{p}))}{h} + O(h^2), \tag{28}$$

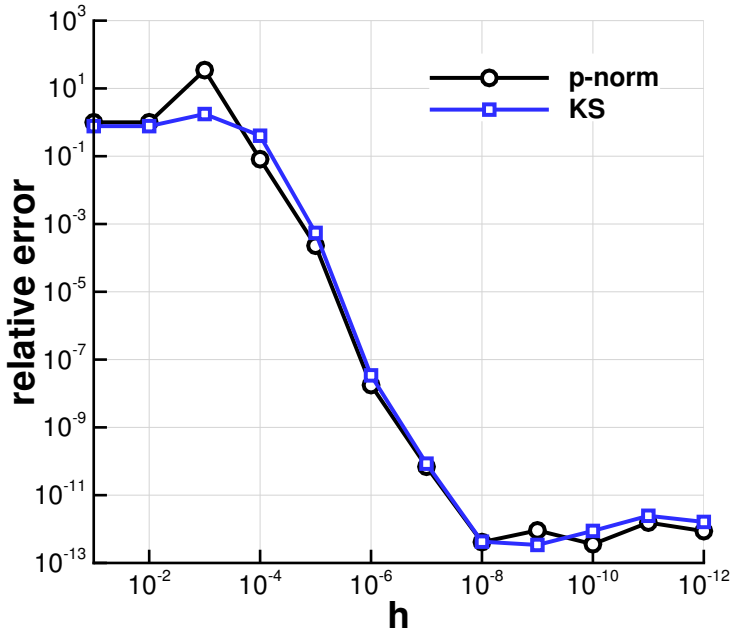


Figure 11: Relative difference between the projected derivatives of the p-norm and KS functionals for increasing complex-step size. Both p-norm and KS gradients agree to 10^{-11} for a sufficiently small step size.

where each component of the vector $\mathbf{p}$ is either 1 or -1 such that $\mathbf{p} = \text{sign}(\nabla_{\mathbf{x}} f)$. This selection ensures that all derivative components add a positive contribution to the inner product $\mathbf{p}^T \nabla_{\mathbf{x}} f$.

To verify the derivatives, we use a small third-order mesh of the wing described above with 576 elements and 2130 nodes, with the isotropic design parametrization described above. Figure 11 shows the relative error between the projected gradient calculated using the complex step and the projected gradient obtained using the adjoint method implementation for both the p-norm and the KS functional. The relative error between the complex step approximation and the adjoint implementation is less than 10^{-11} for complex step sizes less than 10^{-8}.

5.3. Gradient computational cost

In this section, we study the computational cost of the adjoint gradient evaluation method. We examine the scalability of the gradient with number of processors, increasing numbers of functions and increasing numbers of design variables. Our objective is to ensure that problems with large numbers of constraints and large numbers of design variables scale well with increasing numbers of processors.

In this study we examine the four computational steps required to evaluate the total derivative: the evaluation of the derivative $\partial \mathbf{f}_i / \partial \mathbf{u}$, labeled "SV time", the solution of the adjoint equations (20), the evaluation of the derivative $\partial \mathbf{f}_i / \partial \mathbf{x}$, which we label "DV time", and the evaluation of the inner product $\boldsymbol{\psi}_i^T \partial \mathbf{R}_i / \partial \mathbf{u}$. For these comparisons, we use the finite-element wing models from Section 3.2 and in all cases we use an ND ordering of the unknowns. Note

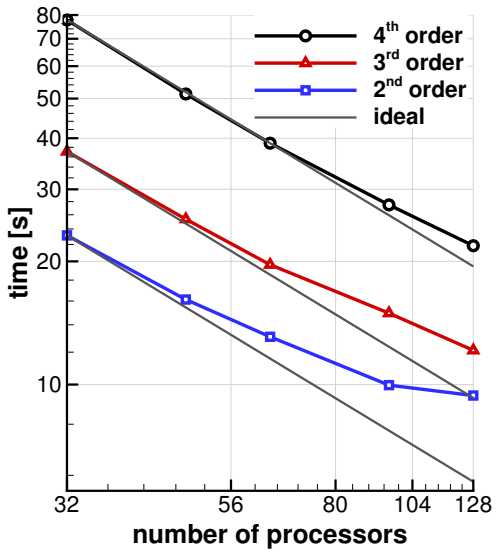
that performance results in this section are primarily dependent on the number of finite-elements within the model and should hold for all models of similar size. However, the adjoint solution time depends on the connectivity of the underlying mesh and may vary for different finite-element models with equal numbers of elements.

For this study, we examine the computational costs of both geometric and material design variables. The geometric variables are the span and chord-length of the wing, as well as the twist at a series of sections spaced evenly out the span of the wing using the geometric parametrization presented in Kennedy and Martins (2012a). In this study we denote the number of geometric design variables n_{xg}, where there are either 11 or 201 geometric design variables. The material design variables are either 220 thicknesses for the metallic wing, or 1320 design variables for the composite wing with 220 thickness variables, 660 ply fraction variables and 440 lamination parameters using the parametrization of Liu et al. (2004). We also examine the effect of varying the number of constraint aggregation domains. We study cases with n_d = 1, 5, 15, and 30, where we enforce the properties of the domain outlined in Equation (16).

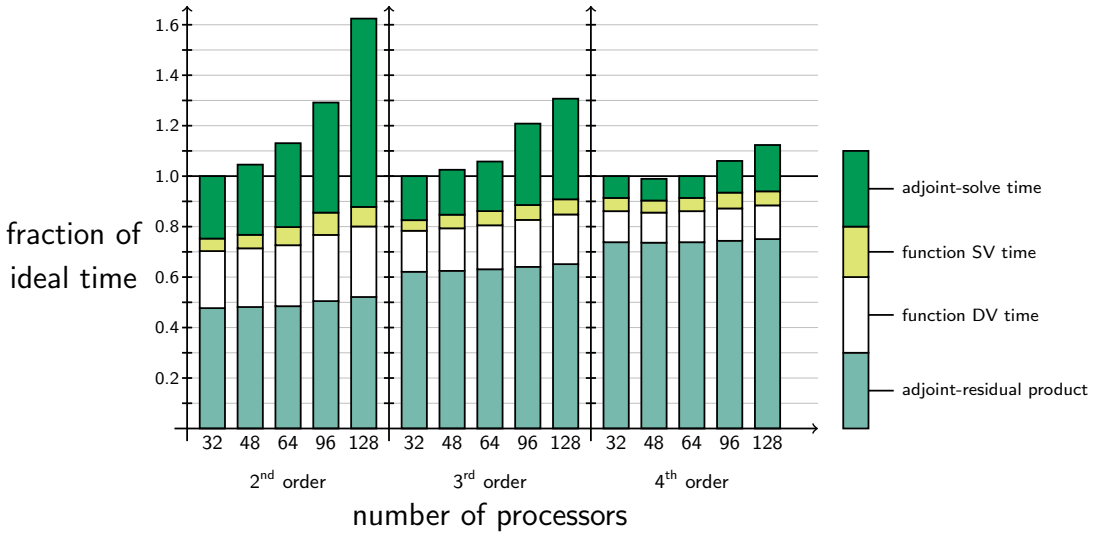
Figure 12 shows the parallel speed up of the gradient computation for second, third and fourth order elements on 32, 48, 64, 96 and 128 processors. For this study, we use 15 KS functionals with 221 geometric variables and the composite material parametrization with 1320 material design variables. Figure 12a shows the parallel scalability of the computation, while Figure 12b shows the fraction of time spent in each operation as a fraction of the ideal lines shown in Figure 12a. The gradient evaluation for the higher-order elements requires more computational time, with the fourth order elements requiring more than 3 times longer than the second order elements. Unfortunately, the gradient computational time does not scale ideally, with the second order elements exhibiting the poorest parallel performance between 96 and 128 processors. This behavior is primarily due to the adjoint solution time, which increases as a fraction of the ideal time as shown in Figure 12b. The fraction of time spent in the remaining operations increases by roughly 12, 6, and 3% for the second, third and fourth order elements, respectively.

Figure 13 shows the computational cost of evaluating the gradient with for 1, 5, 15 and 30 KS functionals with domains that satisfy condition (16). For this study, we use 64 processors, with a design parametrization that includes 221 geometric variables and the composite material parametrization with 1320 material design variables. Figure 13a shows the computational time required for the second, third and fourth order elements and the closest linear approximation of the gradient cost with number of functions. The cost of the gradient evaluation increases at a rate of approximately 0.3 seconds per function for elements of all orders. Figure 13b shows the proportion of computational time spent in each operation normalized to the time required for the evaluation of a single gradient. The increase in the computational time is primarily a result of the additional adjoint solutions required, while the computational time for all other terms increases modestly for increasing function count. For all but the second order case with 30 functions, the most computational time is spent in the evaluation of the term $\boldsymbol{\psi}^T \partial \mathbf{R}_i / \partial \mathbf{x}$. However the computational cost of this term scales only weakly with increasing numbers of functions.

Figure 14 shows the computational time required for evaluating the derivative of 15 KS functionals for increasing numbers of geometric and material design variables. Figure 14a shows the time required to evaluate the gradient with n_{xm} = 220, and 1320 material design variables as well as cases with n_{xm} = 220 material design variables as well as

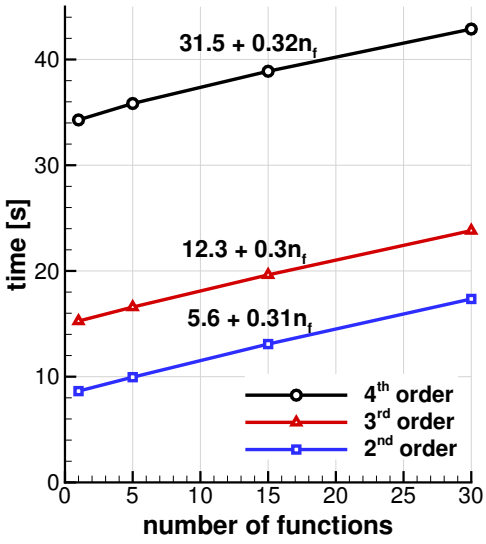


(a) Scalability the adjoint

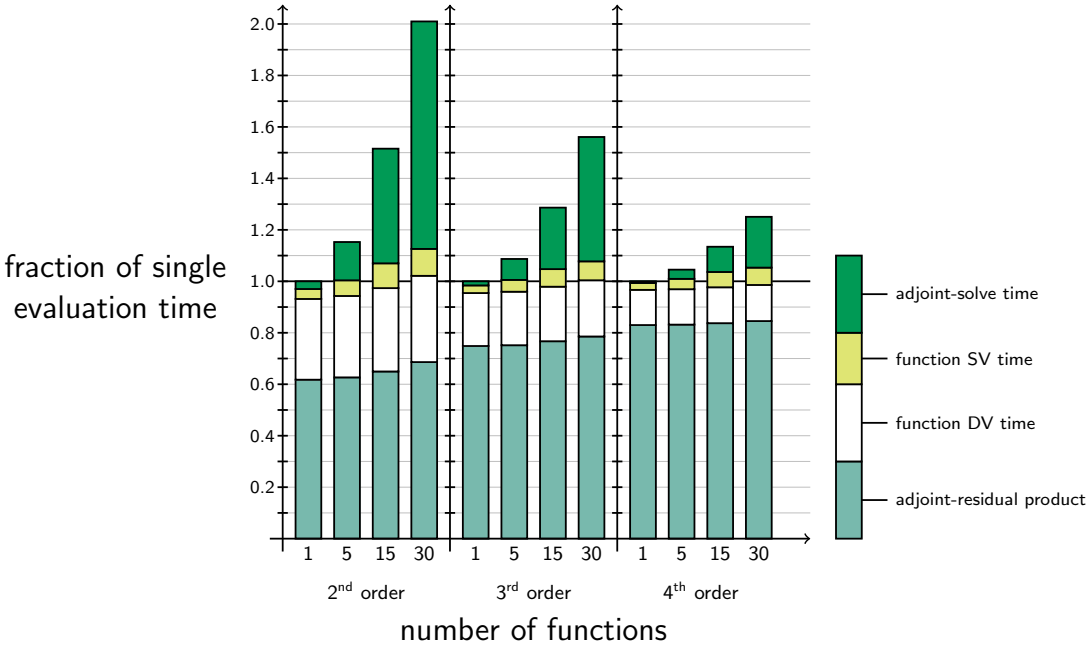


(b) Breakdown of the adjoint

Figure 12: Gradient computational time for increasing numbers of processors. Sub-optimal scaling is due to poor LU-solve performance.

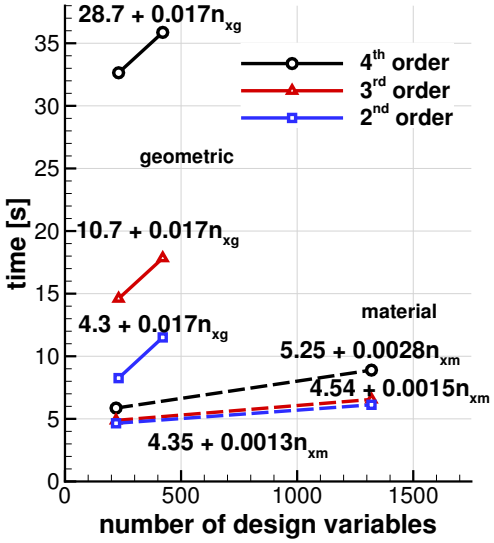


(a) Computational time

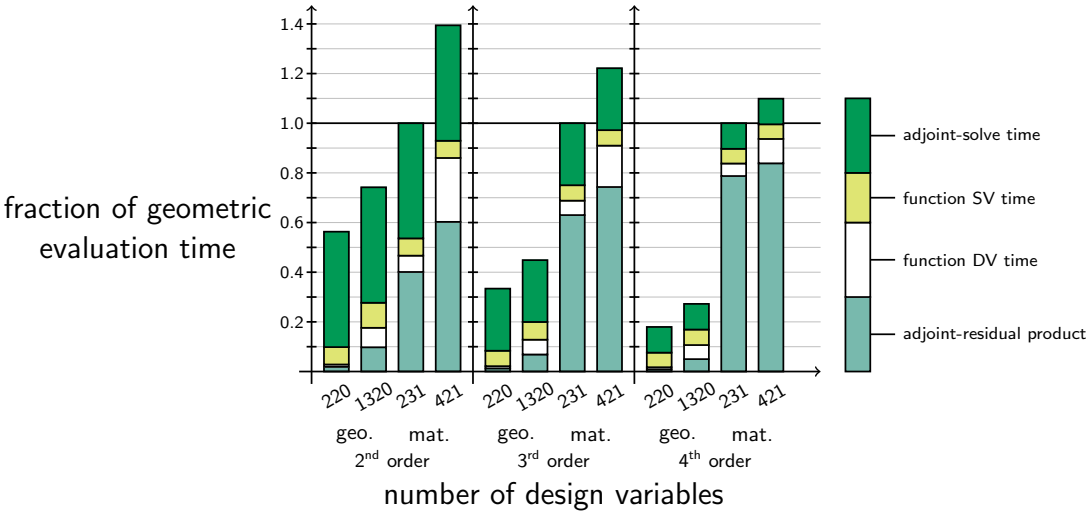


(b) Proportion of computational time

Figure 13: Gradient evaluation time for increasing numbers of KS functionals with 1521 design material and geometric design variables for the 64 processor case with ND ordering.



(a) Design variable scaling



(b) Proportion of computational time

Figure 14: Gradient evaluation time for increasing numbers of material and geometric design variables for the 64 processor ND ordering case with 15 functions.

either $n_{xg} = 11$, or $n_{xg} = 201$ geometric design variables. In addition, Figure 14a shows the linear fit between these design variable groups. Note that the cost of evaluating the derivative with additional geometric design variables is approximately equal for all element orders with 0.017 seconds per geometric design variable. The cost of additional material design variables scales at a rate that is approximately an order of magnitude lower than the geometric design variables, but depends on the element order. Figure 14b shows the proportion of computational time spent in each operation normalized to the $n_{xm} = 220$, $n_{xg} = 11$ case for each element order. Note that the "adjoint-residual product" and "DV time" vary significantly with number of design variables, while the remaining operations take roughly the same fraction of overall time. Figure 14b illustrates that the computational times for the term $\boldsymbol{\psi}^T \partial \mathbf{R}_i / \partial \mathbf{u}$ are much more significant for geometric design variables, especially for the higher-order elements. For the second order and third order elements, a larger proportion of time is spent in the solution of the adjoint vectors.

6. Conclusions

In this paper we presented a fully verified, integrated framework for parallel analysis and gradient-evaluation of shell structures. We described the implementation of a parallel direct matrix factorization method using a domain-decomposition approach that is well suited for both factorization and gradient evaluation tasks. We demonstrated that this direct factorization technique achieves excellent parallel scalability for a large-scale finite-element problem with 5.44 million degrees of freedom on between 24 and 128 processors. We examined the effect of ordering on the performance of the method and found that the AMD, ND, and AMD-OD methods all scale well over this range of processors and can be used to solve the large transonic wing case in less than 40 seconds on 128 processors for second, third and fourth-order shell elements. In order to demonstrate the correctness of our implementation, we also presented a series of verification studies to show that our methods achieve the optimal solution and functional accuracy for each corresponding element. We also presented a gradient evaluation technique that is designed to scale very weakly with increasing numbers of design variables, and scale moderately with number of functions, and exhibits good parallel scalability for all element orders. In addition, we have verified our gradient evaluation methods using a complex-step derivative approximation, demonstrating an accuracy of 10^{-11} in all gradient components. We have integrated the developments presented in this paper into a sophisticated parallel finite-element code that we call the Toolkit for the Analysis of Composite Structures (TACS). These developments make our framework well suited for large-scale high-fidelity structural design optimization problems.

References

Abrate, S., 1994. Optimal design of laminated plates and shells. Composite Structures 29, 269 – 286. doi:10.1016/0263-8223(94)90024-8.

Adams, D.B., Watson, L.T., Gurdal, Z., Anderson-Cook, C.M., 2004. Genetic algorithm optimization and blending of composite laminates by locally reducing laminate thickness. Advances in Engineering Software 35, 35 – 43. doi:10.1016/j.advengsoft.2003.09.001.

Akgun, M.A., Haftka, R.T., Wu, K.C., Walsh, J.L., Garcelon, J.H., 2001. Efficient structural optimization for multiple load cases using adjoint sensitivities. AIAA Journal 39, 511–516. doi:10.2514/2.1336.

Amestoy, P.R., Davis, T.A., Duff, I.S., 1996. An approximate minimum degree ordering algorithm. SIAM Journal on Matrix Analysis and Applications 17, 886–905. doi:10.1137/S0895479894278952.

Amestoy, P.R., Duff, I.S., L'Excellent, J.Y., 2000. Multifrontal parallel distributed symmetric and unsymmetric solvers. Computer Methods in Applied Mechanics and Engineering 184, 501 – 520. doi:10.1016/S0045-7825(99)00242-X.

Ashcraft, C., Grimes, R., 1999. SPOOLES: An object-oriented sparse matrix library, in: Proceedings of the 1999 SIAM Conference on Parallel Processing for Scientific Computing.

Bathe, K.J., Iosilevich, A., Chapelle, D., 2000. An inf-sup test for shell finite elements. Computers & Structures 75, 439 – 456. doi:10.1016/S0045-7949(99)00213-8.

Blackford, L.S., Choi, J., Cleary, A., Petitet, A., Whaley, R.C., Demmel, J., Dhillon, I., Stanley, K., Dongarra, J., Hammarling, S., Henry, G., Walker, D., 1996. Scalapack: a portable linear algebra library for distributed memory computers - design issues and performance, in: Proceedings of the 1996 ACM/IEEE conference on Supercomputing (CDROM), IEEE Computer Society, Washington, DC, USA. doi:10.1145/369028.369038.

Borrvall, T., Petersson, J., 2001. Large-scale topology optimization in 3D using parallel computing. Computer Methods in Applied Mechanics and Engineering 190, 6201 – 6229. doi:10.1016/S0045-7825(01)00216-X.

Bucalem, M.L., Bathe, K.J., 1993. Higher-order MITC general shell elements. International Journal for Numerical Methods in Engineering 36, 3729–3754. doi:10.1002/nme.1620362109.

Buechter, N., Ramm, E., 1992. Shell theory versus degeneration - a comparison of large rotation finite element analysis. International Journal for Numerical Methods in Engineering 34, 39–59. doi:10.1002/nme.1620340105.

Carter, R.G., 1991. On the global convergence of trust region algorithms using inexact gradient information. SIAM Journal on Numerical Analysis 28, 251–265.

Dvorkin, E.N., Bathe, K.J., 1984. A continuum mechanics based four-node shell element for general nonlinear analysis. Engineering Computations 1, 77–88.

Elishakoff, I., 2005. Controversy associated with the so-called "follower forces": Critical overview. Applied Mechanics Reviews 58, 117–142.

Fox, D., Simo, J., 1992. A drill rotation formulation for geometrically exact shells. Computer Methods in Applied Mechanics and Engineering 98, 329 – 343. doi:10.1016/0045-7825(92)90002-2.

Gill, P.E., Murray, W., Saunders, M.A., 2005. SNOPT: An SQP algorithm for large-scale constrained optimization. SIAM Review 47, pp. 99–131.

Herencia, J., Weaver, P., Friswell, M., 2008. Optimization of anisotropic composite panels with T-shaped stiffeners including transverse shear effects and out-of-plane loading. Structural and Multidisciplinary Optimization 37, 165–184. doi:10.1007/s00158-008-0227-6.

Hughes, T.J.R., Brezzi, F., 1989. On drilling degrees of freedom. Computer Methods in Applied Mechanics and Engineering 72, 105–121.

Hvejsel, C., Lund, E., 2011. Material interpolation schemes for unified topology and multi-material optimization. Structural and Multidisciplinary Optimization 43, 811–825. 10.1007/s00158-011-0625-z.

Hvejsel, C., Lund, E., Stolpe, M., 2011. Optimization strategies for discrete multi-material stiffness optimization. Structural and Multidisciplinary Optimization 44, 149–163. doi:10.1007/s00158-011-0648-5.

Jansen, P., Perez, R., 2011. Constrained structural design optimization via a parallel augmented Lagrangian particle swarm optimization approach. Computers & Structures 89, 1352 – 1366. doi:10.1016/j.compstruc.2011.03.011.

Jones, R.M., 1996. Mechanics of Composite Materials. Technomic Publishing Co.

Karypis, G., Kumar, V., 1998. A fast and high quality multilevel scheme for partitioning irregular graphs. SIAM Journal on Scientific Computing 20, 359–392.

Kennedy, G.J., Martins, J.R.R.A., 2010. Parallel solution methods for aerostructural analysis and design optimization, in: Proceedings of the 13th AIAA/ISSMO Multidisciplinary Analysis Optimization Conference, Fort Worth, TX. AIAA 2010-9308.

Kennedy, G.J., Martins, J.R.R.A., 2012a. A comparison of metallic and composite aircraft wings using aerostructural design optimization, in: 14th AIAA/ISSMO Multidisciplinary Analysis and Optimization Conference, Indianapolis, IN.

Kennedy, G.J., Martins, J.R.R.A., 2012b. A homogenization-based theory for anisotropic beams with accurate through-section stress and strain

prediction. International Journal of Solids and Structures 49, 54 – 72. doi:10.1016/j.ijsolstr.2011.09.012.

Kennedy, G.J., Martins, J.R.R.A., 2013a. An adjoint-based derivative evaluation method for time-dependent aeroelastic optimization of flexible aircraft, in: Proceedings of the 54th AIAA/ASME/ASCE/AHS/ASC Structures, Structural Dynamics, and Materials Conference, Boston, MA.

Kennedy, G.J., Martins, J.R.R.A., 2013b. A laminate parametrization technique for discrete ply-angle problems with manufacturing constraints. Structural and Multidisciplinary Optimization , 1–15doi:10.1007/s00158-013-0906-9.

Kenway, G.K.W., Kennedy, G.J., Martins, J.R.R.A., 2013. A scalable parallel approach for high-fidelity steady-state aeroelastic analysis and adjoint derivative computations. AIAA Journal In press.

Le, C., Norato, J., Bruns, T., Ha, C., Tortorelli, D., 2010. Stress-based topology optimization for continua. Structural and Multidisciplinary Optimization 41, 605–620. doi:10.1007/s00158-009-0440-y.

Le Riche, R., Haftka, R.T., 1993. Optimization of laminate stacking sequence for buckling load maximization by genetic algorithm. AIAA Journal 31, 951–956.

Lee, E., James, K.A., Martins, J.R.R.A., 2012. Stress-constrained topology optimization with design-dependent loading. Structural and Multidisciplinary Optimization 46, 647–661. doi:10.1007/s00158-012-0780-x.

Lee, E., Martins, J.R.R.A., 2012. Structural topology optimization with design-dependent pressure loads. Computer Methods in Applied Mechanics and Engineering 233–236, 40–48. doi:10.1016/j.cma.2012.04.007.

Li, X.S., Demmel, J.W., 2003. SuperLU_DIST: A scalable distributed-memory sparse direct solver for unsymmetric linear systems. ACM Trans. Mathematical Software 29, 110–140.

Liem, R.P., Kenway, G.K.W., Martins, J.R.R.A., 2012. Multi-point, multi-mission, high-fidelity aerostructural optimization of a long-range aircraft configuration, in: 14th AIAA/ISSMO Multidisciplinary Analysis and Optimization Conference, Indianapolis, IN.

Liu, B., Haftka, R.T., 2004. Single-level composite wing optimization based on flexural lamination parameters. Structural and Multidisciplinary Optimization 26, 111–120. doi:10.1007/s00158-003-0315-6.

Liu, B., Haftka, R.T., Akgun, M.A., 2000a. Two-level composite wing structural optimization using response surfaces. Structural and Multidisciplinary Optimization 20, 87–96. doi:10.1007/s001580050140.

Liu, B., Haftka, R.T., Akgn, M.A., Todoroki, A., 2000b. Permutation genetic algorithm for stacking sequence design of composite laminates. Computer Methods in Applied Mechanics and Engineering 186, 357 – 372. doi:10.1016/S0045-7825(99)90391-2.

Liu, B., Haftka, R.T., Trompette, P., 2004. Maximization of buckling loads of composite panels using flexural lamination parameters. Structural and Multidisciplinary Optimization 26, 28–36. doi:10.1007/s00158-003-0314-7.

Liu, W., Butler, R., Kim, H.A., 2008. Optimization of composite stiffened panels subject to compression and lateral pressure using a bi-level approach. Structural and Multidisciplinary Optimization 36, 235–245. doi:10.1007/s00158-007-0156-9.

Loken, C., Gruner, D., Groer, L., Peltier, R., Bunn, N., Craig, M., Henriques, T., Dempsey, J., Yu, C.H., Chen, J., Dursi, L.J., Chong, J., Northrup, S., Pinto, J., Knecht, N., Zon, R.V., 2010. SciNet: Lessons learned from building a power-efficient top-20 system and data centre. Journal of Physics: Conference Series 256, 12–26.

MacNeal, R.H., Harder, R.L., 1985. A proposed standard set of problems to test finite element accuracy. Finite Elements in Analysis and Design 1, 3–20.

Mader, C.A., Martins, J.R.R.A., Alonso, J.J., van der Weide, E., 2008. ADjoint: An approach for the rapid development of discrete adjoint solvers. AIAA Journal 46, 863–873. doi:10.2514/1.29123.

Martins, J.R.R.A., Alonso, J.J., Reuther, J.J., 2004. High-fidelity aerostructural design optimization of a supersonic business jet. Journal of Aircraft 41, 523–530. doi:10.2514/1.11478.

Martins, J.R.R.A., Hwang, J.T., 2013. Review and unification of methods for computing derivatives of multidisciplinary computational models. AIAA Journal 51, 2582–2599. doi:10.2514/1.J052184.

Martins, J.R.R.A., Sturdza, P., Alonso, J.J., 2003. The complex-step derivative approximation. ACM Transactions on Mathematical Software 29, 245–262. doi:10.1145/838250.838251.

Milford, R.V., Schnobrich, W.C., 1986. Degenerated isoparametric finite elements using explicit integration. International Journal for Numerical

Methods in Engineering 23, 133–154. doi:10.1002/nme.1620230111.

Padula, S., Stone, S., 1998. Parallel implementation of large-scale structural optimization. Structural optimization 16, 176–185. doi:10.1007/BF01202828.

Papadrakakis, M., Lagaros, N., Tsompanakis, Y., Plevris, V., 2001. Large scale structural optimization: Computational methods and optimization algorithms. Archives of Computational Methods in Engineering 8, 239–301. doi:10.1007/BF02736645.

Papadrakakis, M., Lagaros, N.D., Fragakis, Y., 2003. Parallel computational strategies for structural optimization. International Journal for Numerical Methods in Engineering 58, 1347–1380. doi:10.1002/nme.821.

Pars, J., Navarrina, F., Colominas, I., Casteleiro, M., 2009. Topology optimization of continuum structures with local and global stress constraints. Structural and Multidisciplinary Optimization 39, 419–437. doi:10.1007/s00158-008-0336-2.

Pereira, J., Fancello, E., Barcellos, C., 2004. Topology optimization of continuum structures with material failure constraints. Structural and Multidisciplinary Optimization 26, 50–66. doi:10.1007/s00158-003-0301-z.

Perez, R.E., Jansen, P.W., Martins, J.R.R.A., 2012. pyOpt: a Python-based object-oriented framework for nonlinear constrained optimization. Structural and Multidisciplinary Optimization 45, 101–118. doi:10.1007/s00158-011-0666-3.

Poon, N., Martins, J.R.R.A., 2007. An adaptive approach to constraint aggregation using adjoint sensitivity analysis. Structural and Multidisciplinary Optimization 34, 61–73. doi:10.1007/s00158-006-0061-7.

Saad, Y., 2003. Iterative Methods for Sparse Linear Systems. 2nd ed., PWS Pub. Co.

Simo, J.C., 1993. On a stress resultant geometrically exact shell model. Part VII: Shell intersections with 5/6-dof finite element formulations. Computer Methods in Applied Mechanics and Engineering 108, 319 – 339. doi:10.1016/0045-7825(93)90008-L.

Squire, W., Trapp, G., 1998. Using complex variables to estimate derivatives of real functions. SIAM Review 40, 110–112. doi:10.1137/S003614459631241X.

Stegmann, J., Lund, E., 2005. Discrete material optimization of general composite shell structures. International Journal for Numerical Methods in Engineering , 2009–2027doi:10.1002/nme.1259.

Venkataraman, S., Haftka, R., 2004. Structural optimization complexity: What has Moores law done for us? Structural and Multidisciplinary Optimization 28, 375–387. doi:10.1007/s00158-004-0415-y.

Wang, S., de Sturler, E., Paulino, G.H., 2007. Large-scale topology optimization using preconditioned Krylov subspace methods with recycling. International Journal for Numerical Methods in Engineering 69, 2441–2468. doi:10.1002/nme.1798.

Wrenn, G., 1989. An indirect method for numerical optimization using the Kreisselmeier-Steinhauser function. NASA Technical Report CR-4220.

Appendix A. Cylindrical shell solution

The strain expression for a moderately deep cylinder are:

$$\begin{aligned}
\epsilon_x &= u_{,x} \\
\epsilon_y &= v_{,y} + \frac{w}{a} \\
\gamma_{xy} &= u_{,y} + v_{,x} \\
\kappa_x &= \psi_{x,x} \\
\kappa_y &= \psi_{y,y} - \frac{v_{,y}}{a} - \frac{w}{a^2} \\
\kappa_{xy} &= \psi_{x,y} + \psi_{y,x} - \frac{u_{,y}}{a} \\
\gamma_{yz} &= w_{,y} + \psi_y - \frac{v}{a} \\
\gamma_{xz} &= w_{,x} + \psi_x
\end{aligned} \tag{A.1}$$

where u is the axial displacement, v is the displacement along the circumferential direction and w is the normal displacement. In addition, $x \in [0,\ L]$ is the axial position while $y \in [0,\ 2\pi a]$ is the circumferential position.

If the cylinder is subjected to a sinusoidally varying pressure distribution $p(x,y) = \sum_{n,m} p_{mn} \sin(\alpha_m y) \sin(\beta_n x)$, a displacement solution can be obtained in the following form:

$$
\begin{aligned}
u(x,y) &= \sum_{n,m} U_{mn} \sin(\alpha_m y) \cos(\beta_n x) \quad & \psi_x(x,y) &= \sum_{n,m} \theta_{mn} \sin(\alpha_m y) \cos(\beta_n x) \\
v(x,y) &= \sum_{n,m} V_{mn} \cos(\alpha_m y) \sin(\beta_n x) \quad & \psi_y(x,y) &= \sum_{n,m} \phi_{mn} \cos(\alpha_m y) \sin(\beta_n x) \\
w(x,y) &= \sum_{n,m} W_{mn} \sin(\alpha_m y) \sin(\beta_n x)
\end{aligned}
$$

The principle of minimum potential energy in combination with the strain expressions (A.1) and the assumed displacement distributions yield the following coupled algebraic equations for U_{mn}, V_{mn}, W_{mn}, θ_{mn} and ϕ_{mn}:

$$
\begin{aligned}
-(A_{11}\beta_n^2 + A_{33}\alpha_m^2)U_{mn} - (A_{33} + A_{12})\alpha_m\beta_n V_{mn} + A_{12}\frac{\beta_n}{a}W_{mn} + \frac{D_{33}}{a}\left(\alpha_m^2\theta_{mn} + \alpha_m\beta_n\phi_{mn} - \frac{\alpha_m^2}{a}U_{mn}\right) = 0 \\
-(A_{12} + A_{33})\alpha_m\beta_n U_{nm} - (A_{33}\beta_n^2 + A_{22}\alpha_m^2)V_{mn} + A_{22}\frac{\alpha_m}{a}W_{mn} + \frac{D_{12}}{a}\alpha_m\beta_n\theta_{mn} + \\
\frac{D_{22}}{a}\left(\alpha_m^2\left(\phi_{mn} - \frac{V_{mn}}{a}\right) - \alpha_m\frac{W_{mn}}{a^2}\right) + \frac{\bar{A}_{11}}{a}\left(\alpha_m W_{mn} + \phi_{mn} - \frac{V_{mn}}{a}\right) = 0 \\
-(\bar{A}_{11}\alpha_m^2 + \bar{A}_{22}\beta_n^2)W_{mn} - \bar{A}_{22}\beta_n\theta_{mn} - \bar{A}_{11}\alpha_m\left(\phi_{mn} - \frac{V_{mn}}{a}\right) + \\
\frac{A_{12}}{a}\beta_n U_{mn} - \frac{A_{22}}{a}\left(\frac{W_{mn}}{a} - \alpha_m V_{mn}\right) + \beta_n\frac{D_{12}}{a^2}\theta_{mn} - \frac{D_{22}}{a^2}\left(\frac{W_{mn}}{a^2} - \alpha_n\phi_{mn}\right) + p_{nm} = 0 \\
-(D_{11}\beta_n^2 + D_{33}\alpha_m^2)\theta_{mn} - (D_{12} + D_{33})\alpha_m\beta_n\phi_{mn} - D_{12}\left(\beta_n\frac{W_{mn}}{a^2} - \alpha_m\beta_n\frac{V_{mn}}{a}\right) + \\
\frac{D_{33}}{a}\alpha_m^2 U_{mn} - \bar{A}_{22}(\beta_n W_{mn} + \theta_{mn}) = 0 \\
-(D_{33} + D_{12})\alpha_m\beta_n\theta_{mn} - (D_{33}\beta_n^2 + D_{22}\alpha_m^2)\phi_{mn} + \alpha_m\frac{D_{22}}{a^2}W_{mn} - D_{22}\left(\alpha_m\frac{W_{mn}}{a^2} - \alpha_m^2\frac{V_{mn}}{a}\right) + \\
\frac{D_{33}}{a}\alpha_m\beta_n U_{mn} - \bar{A}_{11}\left(\alpha_m W_{mn} + \phi_{mn} - \frac{V_{mn}}{a}\right) = 0
\end{aligned}
\tag{A.2}
$$

Once the coefficients have been determined from Equation (A.2), the stresses in the cylinder can be obtained as follows:

$$
\begin{aligned}
\sigma_1 &= \sum_{n,m} a_{mn} \sin(\alpha_m y) \sin(\beta_n x) \\
\sigma_2 &= \sum_{n,m} b_{mn} \sin(\alpha_m y) \sin(\beta_n x) \\
\sigma_{12} &= \sum_{n,m} c_{mn} \cos(\alpha_m y) \cos(\beta_n x)
\end{aligned}
$$

where the coefficients a_{mn}, b_{mn} and c_{mn} are as follows:

$$
\begin{aligned}
a_{mn} &= -\beta_n Q_{11}(U_{mn} + z\theta_{mn}) - Q_{12}\left(\alpha_m\left(V_{mn}\left(1 - \frac{z}{a}\right) + z\phi_{mn}\right) - W_{mn}\frac{a - z}{a^2}\right) \\
b_{mn} &= -\beta_n Q_{12}(U_{mn} + z\theta_{mn}) - Q_{22}\left(\alpha_m\left(V_{mn}\left(1 - \frac{z}{a}\right) + z\phi_{mn}\right) - W_{mn}\frac{a - z}{a^2}\right) \\
c_{mn} &= Q_{33}\left(\alpha_m\left(U_{mn}\left(1 - \frac{z}{a}\right) + z\theta_{mn}\right) + \beta_n(V_{mn} + z\phi_{mn})\right)
\end{aligned}
\tag{A.3}
$$

Appendix A.1. p-norm evaluation

To evaluate the p norm, we assume that the cylinder is subject to a load such that only one term $p_{mn} \neq 0\ \forall m, n$.

For an isotropic cylinder, the von Mises stress distribution in the cylinder becomes:

$$\sigma_{vm}^2 = (a_{mn}^2 + b_{mn}^2 - a_{mn}b_{mn}) \sin^2(\alpha_m y) \sin^2(\beta_n x) + 3c_{mn}^2 \cos^2(\alpha_m y) \cos^2(\beta_n x). \tag{A.4}$$

Note that the maximum value of the von Mises stress in the cylinder is:

$$\|\sigma_{vm}\|_\infty = \max \left\{ \sqrt{3}c_{mn},\ \sqrt{a_{mn}^2 + b_{mn}^2 - a_{mn}b_{mn}},\ \sqrt{\frac{3c_{mn}^2(a_{mn}^2 + b_{mn}^2 - a_{mn}b_{mn})}{a_{mn}^2 + b_{mn}^2 - a_{mn}b_{mn} + 3c_{mn}^2}} \right\}. \tag{A.5}$$

The p-norm can be evaluated if $p = 2k$ is an even integer. The p-norm of the von Mises stress can be expressed in the following binomial expansion:

$$\|\sigma_{vm}\|_p = \left(\sum_{i=0}^{k} \binom{k}{i} sc(2i, 2(k-i))(a_{mn}^2 + b_{mn}^2 - a_{mn}b_{mn})^i (3c_{mn}^2)^{k-i} \right)^{\frac{1}{p}}, \tag{A.6}$$

where $sc(p, q)$ is the value of the following integral:

$$\begin{aligned} sc(p, q) &= \int_\Omega \sin^p(\alpha_m y) \sin^p(\beta_n x) \cos^q(\alpha_m y) \cos^q(\beta_n x)\, d\Omega \\ &= \int_0^L \sin^p(\beta_n x) \cos^q(\beta_n x)\, dx \int_0^{2\pi a} \sin^p(\alpha_m y) \cos^q(\alpha_m y)\, dy \end{aligned} \tag{A.7}$$

The value of the integral $sc(p, q)$ is,

$$sc(p, q) = \begin{cases} A \displaystyle\prod_{k=1}^{p/2} \left(\frac{2k-1}{q+2k} \right)^2 \prod_{k=1}^{q/2} \left(\frac{2k-1}{2k} \right)^2 & p \text{ and } q \text{ even} \\ \dfrac{2}{\alpha_m \beta_n (q+1)} \displaystyle\prod_{k=1}^{(p-1)/2} \left(\frac{p-2k-1}{p+q-2k} \right)^2 & p \text{ odd } q \text{ even} \end{cases} \tag{A.8}$$

where $A = 2\pi aL$.

The Tsai–Wu failure criterion is more appropriate for specially orthotropic laminates than the von Mises stress. The Tsai–Wu failure criterion is:

$$F_{TW}(\boldsymbol{\sigma}) = F_1\sigma_1 + F_2\sigma_2 + F_{11}\sigma_1^2 + 2F_{12}\sigma_1\sigma_2 + F_{22}\sigma_2^2 + F_{66}\sigma_{12}^2 \leq 1, \tag{A.9}$$

where the coefficients F_* are selected such that the boundary is a closed convex surface.

In this case, the p-norm can be expressed as follows:

$$\|F_{TW}\|_p = \left(\sum_{i+j+k=p} \binom{p}{i\,j\,k} (f_{mn})^i (g_{mn})^j (h_{mn})^k sc(i + 2j, 2k) \right)^{\frac{1}{p}} \tag{A.10}$$

where the coefficients f_{mn}, g_{mn} and h_{mn} are defined as follows:

$$\begin{aligned} f_{mn} &= F_1 a_{mn} + F_2 b_{mn} \\ g_{mn} &= F_{11} a_{mn}^2 + 2F_{12} a_{mn} b_{mn} + F_{22} b_{mn}^2 \\ h_{mn} &= F_{66} c_{mn}^2 \end{aligned}$$

Algorithm 1: Factorization of a matrix stored in a block-cyclic data format

Given $\mathbf{S}$, compute $\mathbf{L}$ and $\mathbf{U}$ such that $\mathbf{S} = \mathbf{LU}$;

for $i = 1$ **to** n **do**

Diagonal update

if *is_block_owner(i, i)* **then**

compute $\mathbf{L}_{ii}\mathbf{U}_{ii} = \mathbf{S}_{ii}$;

send $\mathbf{U}_{ii}$ to each processor in column i;

send $\mathbf{L}_{ii}$ to each processor in row i;

Column update

if *processor in column i* **then**

receive $\mathbf{U}_{ii}$;

for $j = i + 1$ **to** n **do**

if *is_block_owner(j, i)* **then**

compute $\mathbf{L}_{ji} = \mathbf{S}_{ji}\mathbf{U}_{ii}^{-1}$ and copy $\mathbf{L}_{ji}$ to row_buffer;

send row_buffer to the row processors;

else

receive row_buffer;

Row update

if *processor in row i* **then**

receive $\mathbf{L}_{ii}$;

for $j = i + 1$ **to** n **do**

if *is_block_owner(i, j)* **then**

compute $\mathbf{U}_{ij} = \mathbf{L}_{ii}^{-1}\mathbf{S}_{ij}$ and copy $\mathbf{U}_{ij}$ to column_buffer;

send column_buffer to the column processors;

else

receive column_buffer;

GEMM update

for $j = i + 1$ **to** n **do**

for $k = i + 1$ **to** n **do**

if *is_block_owner(j, k)* **then**

extract $\mathbf{L}_{ji}$ from column_buffer and $\mathbf{U}_{ik}$ from row_buffer;

compute $\mathbf{S}_{jk} \leftarrow \mathbf{S}_{jk} - \mathbf{L}_{ji}\mathbf{U}_{ik}$;

Property	Value
E_{11}	164 GPa
E_{22}	8.3 GPa
G_{12}, G_{13}	21.0 GPa
G_{23}	12 GPa
ν_{12}	0.34
t_{ply}	0.125 mm

Table 1: Representative IM7/3501-6 stiffness and strength properties.