AIAA AVIATION Forum
16-20 June 2014, Atlanta, GA
15th AIAA/ISSMO Multidisciplinary Analysis and Optimization Conference

10.2514/6.2014-2429

Multidisciplinary Design Optimization of an Aircraft Wing via a Matrix-Free Approach

Andrew B. Lambe,*
University of Toronto Institute for Aerospace Studies, Toronto, ON, Canada

Graeme J. Kennedy,†
School of Aerospace Engineering, Georgia Institute of Technology, Atlanta, GA

Joaquim R. R. A. Martins‡
Department of Aerospace Engineering, University of Michigan, Ann Arbor, MI

When solving structural or aerostructural design optimization problems with gradient-based methods, a significant fraction of the total computational cost is computing gradient information for both objective and constraint functions. If the problem has a large number of constraints, the cost of computing all their gradients can become excessive even if adjoint methods are used. A novel "matrix-free" optimization method is proposed in which neither the Hessian nor Jacobian matrices of the optimization problem are explicitly computed. Instead, these matrices are estimated within the optimizer using quasi-Newton approximations. The quasi-Newton approximations are constructed using matrix-vector products with the original matrices. The optimization method is applied to the structural design of an aircraft wing. The results of the study indicate that, compared with traditional optimizers, the total computational cost can be reduced by as much as an order of magnitude.

I. Introduction

We consider the problems of structural and aerostructural design optimization subject to structural failure constraints. When conducted with high-fidelity computational models of the aerodynamic and structural behavior of the system, solving the design optimization problem is a computationally expensive task. This cost comes not just from running the aerodynamic and structural analyses themselves, but also from computing the gradients required for gradient-based optimization. For problems with large numbers of variables and constraints and expensive function evaluations, gradient-based optimization methods offer the only practical way to obtain optimal designs in a reasonable amount of time.

When structural failure constraints are considered in the design optimization problem, the total number of constraints in the problem can be very large. Typically, when solving high-fidelity structural and aerostructural design problems, constraint aggregation is used to reduce the number of constraints and, subsequently, the cost of the sensitivity analysis.[1,2] However, aggregation has several drawbacks depending on the method employed. Aggregation using a p-norm constraint[3] can model the feasible design space accurately but, outside of a few special cases, the p-norm function itself and its gradient are not smooth. (Note that this class of aggregation functions includes the maximum-value function by setting $p = \infty$.) The lack of smoothness acts to slow down the convergence of gradient-based optimization software and may even prevent the problem from being solved at all.

By far, the most common aggregation strategy is to use the Kreisselmeier–Steinhauser (KS) function.[4,5]

*Ph.D. Candidate, AIAA Student Member
†Assistant Professor, AIAA Member
‡Associate Professor, AIAA Senior Member

The KS function is given by

$$\mathrm{KS}[c(x)] = c_{max} + \frac{1}{\rho_{KS}} \ln \left[\sum_{i=1}^{m} \exp\left(\rho_{KS}(c_i(x, y(x)) - c_{max})\right) \right] \tag{1}$$

where c_{max} is the maximum value of the constraint set c. Aggregation using the KS function removes the issue of nonsmoothness but only provides a conservative estimate of the design space.[6,7] This means that some designs that are feasible according to the original set of constraints are deemed infeasible by the corresponding KS constraint. The amount of the feasible design space cut off by the KS function can be reduced by selecting a larger value of the KS penalty parameter, denoted ρ_{KS}. However, larger values of this parameter correspond to more ill-conditioned optimization problems, slowing down convergence. Therefore, we are necessarily forced to compromise between the quality of the optimal solution and the amount of work required to obtain it.

In this work, we propose a way around this compromise so that a large number of design constraints may be handled more efficiently. Instead of computing the derivatives of all functions with respect to all variables—i.e. the full constraint Jacobian—every time we compute a new design point, we propose to only compute the product of this Jacobian with a few vectors of interest. As a result, we are able to reduce the cost of the sensitivity analysis without resorting to excessive aggregation of the constraints. However, to be able to do this requires fundamental changes in the structure of the optimizer and how it receives information from the disciplinary or multidisciplinary analysis. In short, we require an optimizer that does not require full matrices of gradient information or a "matrix-free" optimizer.

The remainder of this paper is organized as follows. In Section II, we define the optimization problem statement and review the equations for computing design sensitivities for the functions in the problem. In Section III, we give an overview of our optimization algorithm and the modifications necessary to derive a matrix-free algorithm. In Section IV, we outline our test problem, the design optimization of a wing box, and the analysis tools used to inform the optimizer about the design. Section V outlines the results of our optimization and how the matrix-free optimizer compares with a more traditional optimization approach. Finally, Section VI draws conclusions and outlines future research directions.

II. Sensitivity Analysis for Gradient-Based Optimization

All of our design optimization problems can be simplified to fall under the following model problem.

$$\begin{aligned} \text{minimize} \quad & f(x, y(x)) \\ \text{with respect to} \quad & x \\ \text{subject to} \quad & c(x, y(x)) \leq 0 \\ & x_L \leq x \leq x_U \\ \text{where} \quad & y \text{ solves } \mathcal{R}(x, y(x)) = 0 \end{aligned} \tag{2}$$

In problem (2), $\mathcal{R}$ represents the governing equations of the analysis and y represents the state variables. This problem is known as Nested Analysis and Design in the literature.[8] If the analysis consists of several disciplines, problem (2) represents a Multidisciplinary Feasible (MDF) problem formulation.[9,10] Computing the gradients of f and c with respect to x can be done in several ways but analytic approaches are by far the most accurate and cost-effective if the software is available to do so.[11]

We now briefly derive the equations for analytic sensitivity analysis and show how to obtain matrix-vector products efficiently from these expressions. While our derivation only covers sensitivity analysis for a single discipline, we note that the multidisciplinary case is a simple extension of our work and has been noted in the literature previously.[11–13]

Under problem (2), we can compute the derivative of constraint c_i with respect to variable x_j as

$$\begin{aligned} \frac{dc_i}{dx_j} &= \frac{\partial c_i}{\partial x_j} + \sum_{k=1}^{M} \frac{\partial c_i}{\partial y_k} \frac{dy_k}{dx_j} \\ &= \frac{\partial c_i}{\partial x_j} + \frac{\partial c_i}{\partial y} \frac{dy}{dx_j}, \end{aligned} \tag{3}$$

where M is the total number of state variables. The notation $d()/dx$ is used to indicate that the derivative accounts for the solution of the governing equations. We adopt the shorthand notation

$$\frac{\partial c}{\partial x} = \frac{\partial(c_1, ..., c_m)}{\partial(x_1, ..., x_n)} \in \mathbb{R}^{m \times n}$$

to describe the Jacobian of a set of functions with respect to a set of variables. For example, the $\partial c_i/\partial y$ term in Equation (3) is a row vector of length M containing all $\partial c_i/\partial y_k$ values.

The dy/dx_j term is computed by recognizing that the system $\mathcal{R}(x, y(x)) = 0$ has been solved for x_j and no change in x_j alters this fact. In other words, disciplinary feasibility is maintained by construction in our problem statement. Therefore, we can state that

$$\frac{d\mathcal{R}}{dx_j} = 0 = \frac{\partial \mathcal{R}}{\partial x_j} + \frac{\partial \mathcal{R}}{\partial y}\frac{dy}{dx_j}. \tag{4}$$

Rearranging the terms in Equation (4) yields

$$\frac{dy}{dx_j} = -\left[\frac{\partial \mathcal{R}}{\partial y}\right]^{-1}\frac{\partial \mathcal{R}}{\partial x_j}. \tag{5}$$

By direct substitution of Equation (5) into Equation (3), we obtain

$$\frac{dc_i}{dx_j} = \frac{\partial c_i}{\partial x_j} - \left[\frac{\partial c_i}{\partial y}\right]\left[\frac{\partial \mathcal{R}}{\partial y}\right]^{-1}\frac{\partial \mathcal{R}}{\partial x_j}. \tag{6}$$

Finally, we drop the subscripts on c and x to obtain an expression for the full constraint Jacobian.

$$\frac{dc}{dx} = \frac{\partial c}{\partial x} - \left[\frac{\partial c}{\partial y}\right]\left[\frac{\partial \mathcal{R}}{\partial y}\right]^{-1}\frac{\partial \mathcal{R}}{\partial x} \tag{7}$$

Note that Equation (7) describes an $m \times n$ matrix, where m is the number of constraints and n is the number of design variables.

We can use Equation (7) to derive two different schemes for computing the full constraint Jacobian. These schemes are the well-known *direct* and *adjoint* sensitivity analysis methods.[11] In the direct method, a sequence of linear systems of the form

$$\left[\frac{\partial \mathcal{R}}{\partial y}\right]\frac{dy}{dx_j} = -\frac{\partial \mathcal{R}}{\partial x_j} \tag{8}$$

is solved to yield column vectors dy/dx_j. These vectors are arranged to form a matrix, dy/dx, that is then multiplied by $\partial c/\partial y$ to compute dc/dx. In the adjoint method, an alternative sequence of linear systems of the form

$$\left[\frac{\partial \mathcal{R}}{\partial y}\right]^T \psi_{c_i} = -\left[\frac{\partial c_i}{\partial y}\right]^T \tag{9}$$

is solved to yield column vectors ψ_{c_i}. These vectors are arranged to form a matrix, ψ_c^T, that is then multiplied by $\partial \mathcal{R}/\partial x$ to compute dc/dx. For optimization problems with high-fidelity computational models, $M >> m, n$ so inverting the matrix $\partial \mathcal{R}/\partial y$ to solve (8) or (9) is the most costly operation in forming the Jacobian. Since this operation occurs either m or n times, we select either the direct or adjoint methods depending on whether the problem has more variables or more constraints. However, if both m and n are fairly large, e.g. $n, m \geq 100$, then neither method is particularly efficient because either (8) or (9) must be solved hundreds of times for different right-hand sides.

While obtaining the full Jacobian can be prohibitively expensive for a problem with both many variables and many constraints, obtaining individual matrix-vector products is still relatively inexpensive. If we multiply the Jacobian expression in (7) by an appropriate vector v, we obtain the following expression.

$$\left[\frac{dc}{dx}\right]v = \left[\frac{\partial c}{\partial x}\right]v - \left[\frac{\partial c}{\partial y}\right]\left[\frac{\partial \mathcal{R}}{\partial y}\right]^{-1}\left[\frac{\partial \mathcal{R}}{\partial x}\right]v \tag{10}$$

Similarly, if we multiply the traspose Jacobian by an appropriate vector w, we obtain the following expression.

$$\left[\frac{dc}{dx}\right]^T w = \left[\frac{\partial c}{\partial x}\right]^T w - \left[\frac{\partial \mathcal{R}}{\partial x}\right]^T \left[\frac{\partial \mathcal{R}}{\partial y}\right]^{-T} \left[\frac{\partial c}{\partial y}\right]^T w \tag{11}$$

These expressions reveal two important properties of the matrix-free approach. First, by evaluating the matrix-vector products right-to-left in both expressions, we only require a single inversion of $\partial\mathcal{R}/\partial y$ or its transpose to evaluate the product. Second, we do not need to form any of the partial derivative matrices explicitly. Therefore, provided the number of matrix-vector products can be kept small by the optimization algorithm, the matrix-free approach can provide a time- and memory-efficient way to solve optimization problems with large, expensive Jacobian matrices.

The main issue with the matrix-free approach to optimization is that specialized software is required that allows the user to supply only Jacobian-vector products instead of a full Jacobian matrix. We have sought to create our own implementation of a matrix-free algorithm due to a lack of available software suitable for the structure of our optimization problem. In the next section, we outline our matrix-free implementation of a traditional optimization algorithm.

III. Algorithm Overview

While, in principle, numerous existing optimization algorithms can be adapted as matrix-free optimizers, we have chosen to adapt the classical augmented Lagrangian algorithm. We chose this algorithm due to its long heritage in optimization, well-established convergence properties, and ease of implementation. This section provides an overview of the basic algorithm and our adaptations to make it fully matrix-free. For a more detailed discussion of the augmented Lagrangian algorithm itself, we refer the reader to one of several textbooks on optimization.[14–16] More details of our implementation, along with performance benchmarks and a scalability study are presented in a forthcoming paper.[17]

Given a nonlinear optimization problem with equality constraints and variable bounds,

$$\begin{aligned} \text{minimize} \quad & f(x) \\ \text{with respect to} \quad & x \\ \text{subject to} \quad & c(x) = 0 \\ & x_L \leq x \leq x_U, \end{aligned} \tag{12}$$

the augmented Lagrangian algorithm relaxes the problem into the form

$$\begin{aligned} \text{minimize} \quad & \phi(x;\lambda,\rho) = f(x) - \lambda^T c(x) + \frac{\rho}{2} c(x)^T c(x) \\ \text{with respect to} \quad & x \\ \text{subject to} \quad & x_L \leq x \leq x_U, \end{aligned} \tag{13}$$

where $\rho > 0$ is the penalty parameter and λ is the vector of Lagrange multipliers. (Note that a problem with inequality constraints can easily be converted into the canonical form (12) by introducing additional "slack" variables.) The algorithm solves a sequence of bound-constrained minimization problems of the form (13). After minimization k, either the penalty parameter is increased by a factor, or the multipliers are updated using the rule

$$\lambda^{k+1} = \lambda^k + \rho c(x^k) \tag{14}$$

depending on whether or not the constraint infeasibility has been reduced sufficiently. The problem is deemed to be solved when

$$||P(\nabla L(x^*, \lambda^*))||_\infty \leq \omega^*$$

where $L(x,\lambda) = f(x) - \lambda^T c(x)$ and $P()$ represents the projection of the gradient on to the feasible region, i.e.,

$$\begin{aligned} P(g_i(x)) &= 0 && \text{if } g_i(x) > 0 \text{ and } x_i = x_{Li} \\ P(g_i(x)) &= 0 && \text{if } g_i(x) < 0 \text{ and } x_i = x_{Ui} \\ P(g_i(x)) &= g_i(x) && \text{otherwise,} \end{aligned}$$

and

$$||c(x^*)||_\infty \leq \eta^*$$

for the defined constants ω^* and η^*. Rules given by Conn et al[14] allow subproblems of the form (13) to be solved inexactly while maintaining the global convergence property of the algorithm.

The nonlinear bound-constrained problems (13) are solved using an infinity-norm trust-region algorithm. The infinity-norm is preferred over the usual Euclidean-norm because of the presence of the bound constraints. Effectively, the infinity-norm imposes an additional set of bound constraints on the trust region subproblem, allowing us to consider all bound constraints directly at the same time. The trust region subproblem has the following form.

$$\begin{aligned} \text{minimize} \quad & \frac{1}{2}p^T H p + g^T p \\ \text{with respect to} \quad & p \\ \text{subject to} \quad & p_L \leq p \leq p_U \end{aligned} \tag{15}$$

This bound-constrained, nonconvex, quadratic subproblem can be solved by well-known iterative methods, such as the algorithm of Moré and Toraldo.[18] Because this algorithm uses a combination of gradient projection and conjugate gradient iterations, it does not explicitly require the Hessian matrix H in (15). Therefore, as long as we can provide products with the matrix H, the entire augmented Lagrangian algorithm can be adapted as a matrix-free optimizer. The solution to problem (15) is a design variable step that is used to update the current point. The updated point is accepted or rejected, and the trust region is expanded or restricted, following the usual rules.[19]

Key to the success of the matrix-free optimizer is the ability to compute derivative information without forming large matrices. Examining the first and second derivatives of the augmented Lagrangian function from (13), we obtain the following expressions.

$$\nabla\phi = \nabla f + J^T(\rho c(x) - \lambda) \tag{16}$$

$$\nabla^2\phi = \nabla^2 f - \sum_{i=1}^{m} \lambda_i \nabla^2 c_i + \sum_{i=1}^{m} \rho c_i(x) \nabla^2 c_i + \rho J^T J \tag{17}$$

Here, we have used J to denote the Jacobian of c with respect to x. Note that the gradient of the augmented Lagrangian is easy to obtain without forming the original constraint Jacobian; only a single matrix-vector product with the transpose Jacobian is required. Therefore, we can take g in Problem (15) to be exactly (16) and optimizer convergence can be checked using exact gradient information. Obtaining the Hessian of the augmented Lagrangian requires second derivatives of both the objective and constraint functions, which are not accessible. However, we can use a quasi-Newton method to obtain an approximation to this matrix for use in the trust region subproblem (15). In our code, we have the option of using limited-memory versions of the BFGS or SR1 approximation methods.[16,20]

Through experiments, we found that naïvely applying a quasi-Newton approximation to (17) resulted in poor performance of the optimization algorithm. Much better performance can be obtained by exploiting the structure of the Hessian in (17). One approach to exploiting problem structure is to separate the augmented Lagrangian function into two terms—the Lagrangian function $f(x) - \lambda^T c(x)$, and the constraint infeasibility $(\rho/2)c(x)^T c(x)$—and maintain a separate quasi-Newton approximation to the Hessian of each. Since the infeasibility function is convex near a local minimum of (13), the BFGS approximation is preferred for this term. Another way of exploiting the Hessian structure (17) is to observe that the second-order term $\sum_{i=1}^{m} \rho c_i(x) \nabla^2 c_i$ vanishes at a feasible solution. The remaining term, $\rho J^T J$ can be estimated by directly approximating J and using that approximation in both forward and transpose matrix-vector products. In general, this approach requires us to form a quasi-Newton approximation of a nonsquare matrix.

In this last strategy, the question arises of whether or not to approximate J at all. In principle, because we only need Hessian-vector products with our model Hessian H, we could form products with $\rho J^T J$ without substituting in an approximation to J. Based on our testing, this approach is effective but expensive. Under this strategy, the Jacobian-vector products are required within the conjugate gradient method, rather than the trust-region method, and the resulting number of expensive matrix-vector products is extremely high. To achieve our initial goal of keeping the number of these expensive products down, we have to resort to forming an approximate Jacobian via a quasi-Newton method.

While quasi-Newton approximations to nonsquare matrices have been known in the literature for many years,[21,22] they have not frequently been applied to optimization problems. In our matrix-free algorithm implementation, we use the adjoint Broyden method discussed by Schlenkrich et al.[23] For a given Jacobian approximation A_k, the adjoint Broyden update is given by

$$A_{k+1} = A_k + \frac{\sigma_k \sigma_k^T}{\sigma_k^T \sigma_k} \left(J_{k+1} - A_k \right), \tag{18}$$

where J_{k+1} is the current Jacobian matrix and σ_k is an adjoint search direction. We choose the adjoint search direction to be the difference between the true and approximate matrix-vector product with the most recent step taken by the optimization algorithm, i.e.,

$$\sigma_k = (J_{k+1} - A_k)(x_{k+1} - x_k). \tag{19}$$

With this choice of σ, the update (18) requires two matrix-vector products with the true Jacobian each time the trust region subproblem is updated. Like quasi-Newton updates for symmetric and square matrices, the adjoint Broyden method posesses a bounded deterioration property to ensure that the approximate matrix converges to the true matrix as more information about the problem is gathered.[23]

For our study, we initialize the approximate Jacobian to be the true Jacobian and store the full approximation matrix. Unlike the limited-memory Hessian approximations, there is no literature establishing the convergence properties of a limited-memory Jacobian estimate. While this leads to a more memory-intensive algorithm, we can make use of the existing distributed-memory parallel-processing environment to store the matrix and compute the matrix-vector products efficiently. Furthermore, because we never need to factorize this matrix, memory savings are still substantial compared with a full-Jacobian SQP approach.

To summarize, we have developed two distinct approaches for approximating the Hessian of the augmented Lagrangian without explicitly forming any Hessian or Jacobian matrices. In the first approach, we choose our model Hessian H to be

$$H = B_L + \rho B_I$$

where B_L is the approximate Hessian of the Lagrangian and B_I is the approximate Hessian of the infeasibility function $(1/2)c(x)^T c(x)$. We refer to this strategy as the "split" quasi-Newton approach. In the second approach we choose

$$H = B_L + \rho A^T A$$

where A is an approximate Jacobian computed by (18). We refer to this strategy as the "approximate J" approach. In both cases, the approximations are restricted to the trust region subproblem (15) so that precise information is used to establish convergence to a local minimum.

IV. Wing Model and Optimization Environment

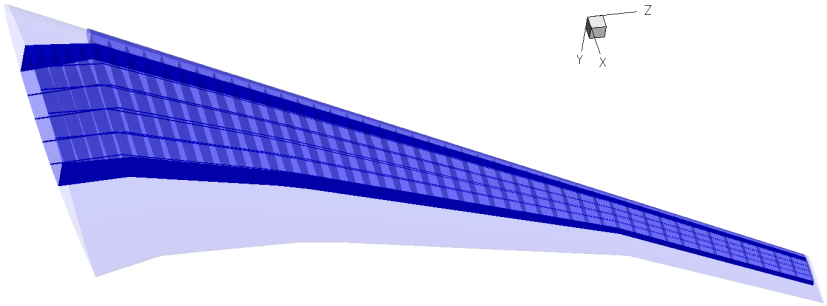

Figure 1. Outer geometry and layout of the wing structure to be optimized.

The wing box model we will design is loosely derived from the wing of a Boeing 777-200 civil transport. We have created a simple structural layout including top and bottom skins, spars, ribs, and the leading edge structure of the wing. Both skins have stiffeners running spanwise through the middle of the wing box.

Figure 1 provides an overview of the geometry. The structural design variables are simply the thicknesses of each component. A total of 1416 design variables are used to parametrize the wing structure. The wing is analyzed for two load conditions: a 2.5g climb and a 1g dive at Mach 0.85. The maximum number of stress constraints in the problem is 2832: 1416 stress values tested for two loading conditions. Finally, a minimum thickness of 0.096 inches and a maximum thickness of 3 inches is defined for each design variable. In all the forthcoming optimization results, the objective is to minimize mass subject to thickness bounds and failure constraints.

The structure is analyzed using the Toolkit for the Analysis of Composite Structures, (TACS,) an in-house finite-element code designed specifically for thin-walled structures.[24] We use a third-order finite-element mesh to obtain accurate stress and strain distributions within the structure. While TACS is capable of analyzing composite structures, our test cases assume that the wing structure is metallic. The final finite-element wing model contains nearly 46 000 elements and 250 000 degrees of freedom.

The aerodynamic analysis is carried out using the unstructured panel code TriPan.[25] While panel codes are not the highest-fidelity analysis tools, they are accurate enough for the purpose of testing the optimization algorithm. Aerodynamic forces on the structure are computed by simple integration of the pressure distribution. To transfer these aerodynamic loads to the structure, we use a consistent and conservative transfer scheme derived from the method of virtual work. This scheme uses a system of rigid links to connect the nodes on the aerodynamic mesh with the nearest point on the structural mesh.[26] The interested reader may refer to the journal paper by Kennedy and Martins[25] for further details on the aerostructural analysis. While the present results concern structural optimization only, the availability of this transfer scheme makes extending the results to aerostructural problems in the near future easy.

Our matrix-free optimizer is implemented in Python in the open-source NLPy environment.[27] NLPy is an object-oriented toolkit that provides common implementations of a variety of optimization algorithms. By design, NLPy allows for the reuse of common building blocks to allow users to rapidly develop and test optimization algorithms. Our results are benchmarked against the active-set sequential quadratic programming software SNOPT,[28] which is accessed through the pyOpt framework.[29]

V. Results and Discussion

Before studying the impact of using a matrix-free optimizer, we want to quantify what we give up by using constraint aggregation on the minimum mass structural design problem. For this series of tests, we optimize the structure using SNOPT, a varying number of KS functions, and a varying value of ρ_{KS}. The starting point for all test cases is a structure with a uniform thickness of 0.25 inches in all components. This is an infeasible design point. The feasibility and optimality tolerances for SNOPT were set to 10^{-5} if the total number of constraints was 100 or less and 10^{-3} for the case of 2832 constraints. A looser constraint tolerance was necessary to obtain solutions in a reasonable run time, even with high-performance computing facilities available.

Figure 2 shows the comparison in the optimum mass of the wing for ρ_{KS} values between 20 and 100, using four different aggregation schemes. The masses are normalized with respect to the optimum mass computed for the case of 2832 constraints and $\rho_{KS} = 100$ — about 26 000 pounds. In the most-restrictive aggregation scheme, (ten KS constraints,) the wing is subdivided into five domains, (spars, ribs, lower skin, upper skin, and leading-edge structure,) and all of the constraints for each domain are aggregated for each load condition. For the 50-KS-constraint and 100-KS-constraint cases, these domains are further subdivided into five and ten smaller domains, and constraint aggregation occurs over these smaller domains. For the least-restrictive aggregation scheme, the KS function is only applied over individual components of the wing, i.e. the same regions over which the design variables are defined.

The most striking feature of Figure 2 is the spread of the optimum mass values, even as ρ_{KS} increases. The "diminishing returns" trend in the optimum mass values in Figure 2 is to be expected given the exponential nature of the KS function itself. However, even at $\rho_{KS} = 100$, the spread in the optimum masses between the four cases is about 5% of the lowest optimum mass. In some literature,[6,7,30] an upper limit of $\rho_{KS} = 50$ is recommended due to the ill-conditioning imposed on the optimization problem by the aggregation. However, for this parameter setting in Figure 2, the spread in the optimum mass is about 10% of the lowest optimum mass. While we do not claim that such behavior is typical for minimum-mass structural design, we do emphasize that such a large overestimation of the optimum mass can happen if constraint aggregation is not applied judiciously.

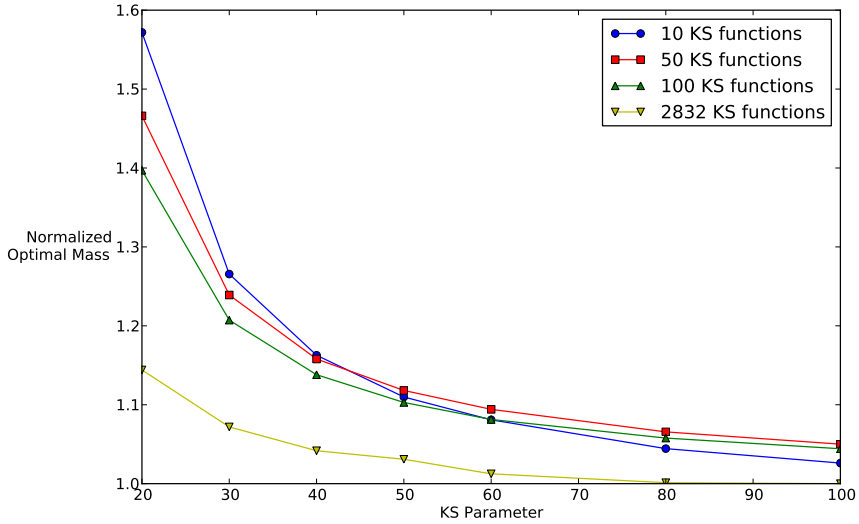

Figure 2. Normalized optimal mass of the test wing for various aggregation schemes.

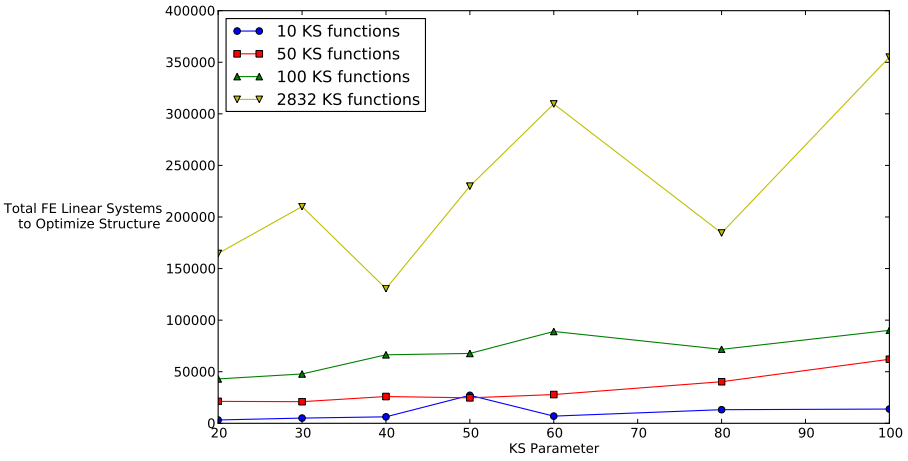

Figure 3. Computational work to optimize test wing for various aggregation schemes.

The primary motivation for using aggregation is to reduce the amount of computational work, in the form of direct and adjoint linear-system solves, required to complete the optimization. By this metric, constraint aggregation is successful. Figure 3 shows the computational work in terms of the total number of finite-element linear-system solves, both direct and adjoint, needed to optimize the structure. Even though three of the four aggregation schemes are optimized to a tighter tolerance, they still only require a fraction of the work required by the test cases using the maximum number of constraints. Surprisingly, the computational work does not increase drastically with an increase in ρ_{KS}. This is likely due to the robustness of the SNOPT optimizer and its ability to solve ill-conditioned optimization problems.

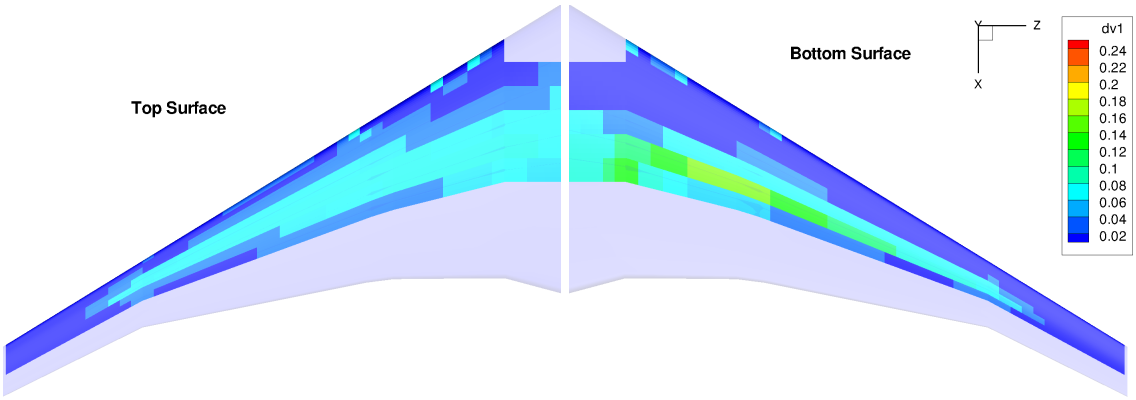

Figure 4. Optimal thickness of top and bottom skins for problem formulation with $\rho_{KS} = 100$ and 2832 constraints.

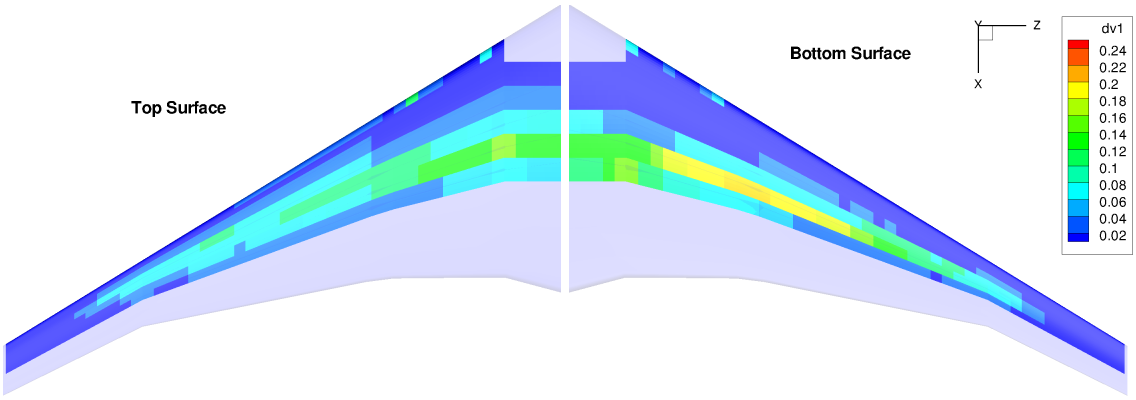

Figure 5. Optimal thickness of top and bottom skins for problem formulation with $\rho_{KS} = 20$ and 2832 constraints.

The lowest-mass wing computed by SNOPT, the one using $\rho_{KS} = 100$ and 2832 constraints in the problem formulation is presented in Figures 4 and 6. The colors represent the thickness of each component, in feet. As we might expect, most of the load is carried in the middle of the structure. The thickest components of the wing are the skin stiffeners on both the top and bottom skins. In a few cases, these components are designed to be the maximum allowed thickness, likely because we fix the depth of the stiffener in our problem formulation. The effect of altering the KS parameter can be seen by comparing Figures 4 and 6 with Figures 5 and 7, which were generated for the case of $\rho_{KS} = 20$. Using a smaller KS parameter causes the outer skins to be thickened, which increases the mass of the optimal wing design by over 3000 pounds.

The stress distributions corresponding to the optimal solutions in Figures 4 and 5 are plotted in Figures 8 and 9. Stress values are normalized with respect to the local Von Mises yield stress. Both designs satisfy the feasibility tolerance for their respective problem formulations, but the wing in Figure 8 is clearly more fully-stressed. As expected, using a larger value of ρ_{KS} leads to a more fully-stressed design, one of the hallmarks of a more efficient structure.

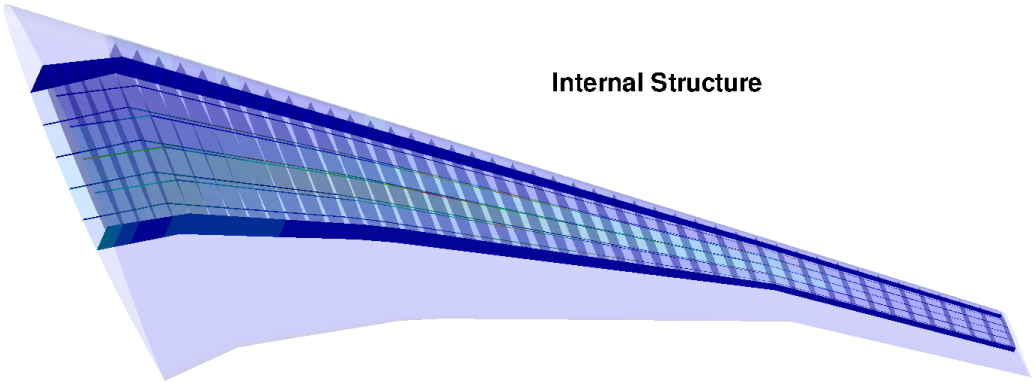

Figure 6. Optimal thickness of internal structure for problem formulation with $\rho_{KS} = 100$ and 2832 constraints.

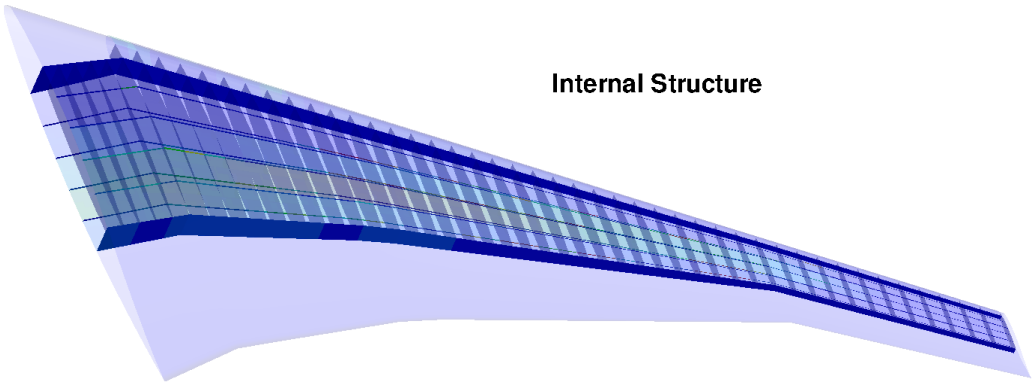

Figure 7. Optimal thickness of internal structure for problem formulation with $\rho_{KS} = 20$ and 2832 constraints.

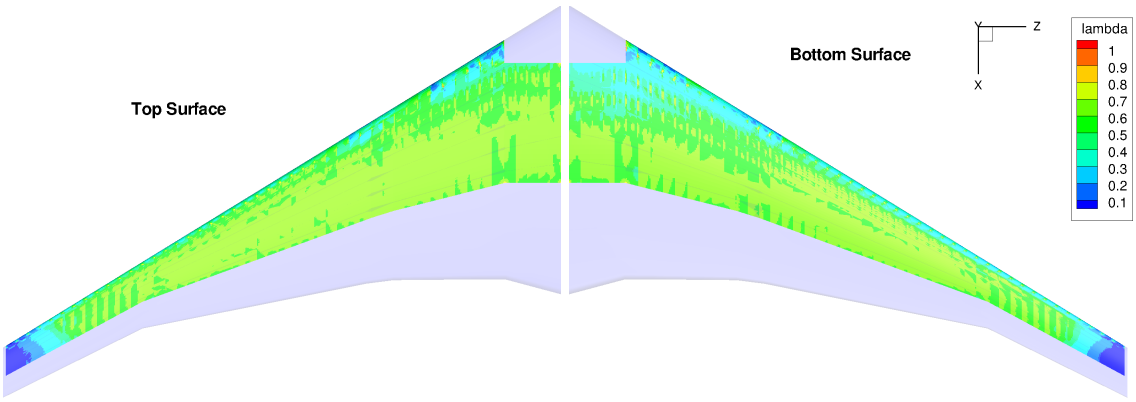

Figure 8. Von Mises stress distribution of the optimal solution for problem formulation with $\rho_{KS} = 100$ and 2832 constraints.

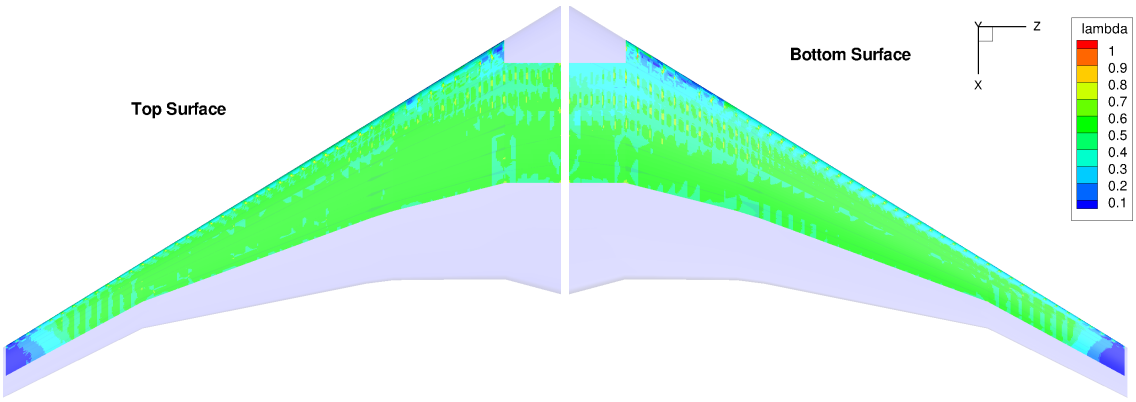

Figure 9. Von Mises stress distribution of the optimal solution for problem formulation with $\rho_{KS} = 20$ and 2832 constraints.

By using the matrix-free optimizer outlined in Section III, we hope to reduce the computational cost of solving an optimization problem with a large number of variables and constraints. First, we verify that our matrix-free optimizer obtains the correct solution for the case of 2832 constraints. To provide a fair comparison with SNOPT, the feasibility and optimality tolerances of the matrix-free optimizer are also set to 10^{-3}. The mass results are plotted in Figure 10. The case of 50 KS constraints is also plotted as a reference. Both matrix-free strategies successfully compute an optimal design on all test cases when using 2832 design constraints. The optimal structure computed by the matrix-free strategies seems to have a slightly lower mass (by about 200 pounds) than the optimal structure computed by SNOPT. We attribute this to the loose convergence tolerance and the difference in optimization algorithms. Whereas an SQP algorithm like SNOPT tries to satisfy feasibility and optimality simultaneously, the augmented Lagrangian algorithm aggressively seeks out optimality and is pushed back to feasibility by infrequent updates of the penalty parameter and Lagrange multiplers. Therefore, the augmented Lagrangian will approach the optimal solution from an infeasible direction in the design space.

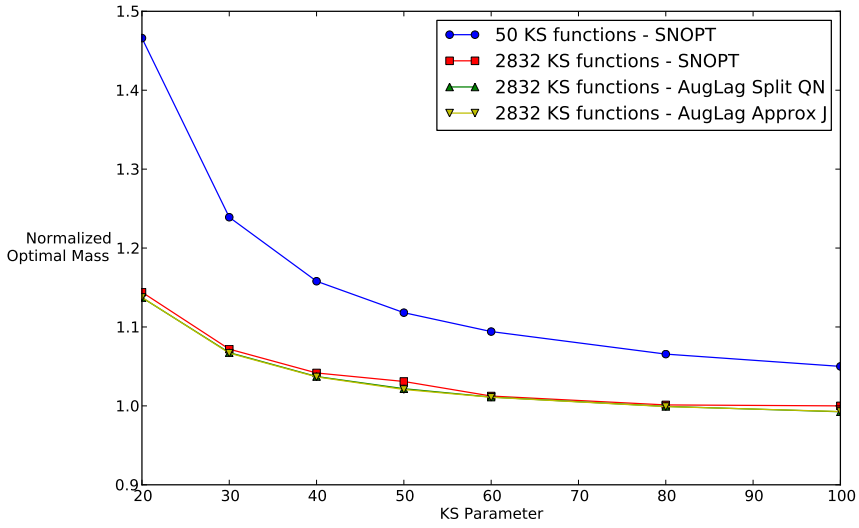

Figure 10. Normalized optimal mass comparison between SNOPT and the matrix-free optimizers.

Figure 11 compares the computational effort of SNOPT and the matrix-free optimizers to obtain these solutions. In terms of the number of linear systems to solve to obtain an optimal structure, the matrix-free approach is far less expensive than SNOPT. For $\rho_{KS} = 60$ and $\rho_{KS} = 100$, we observed a reduction in

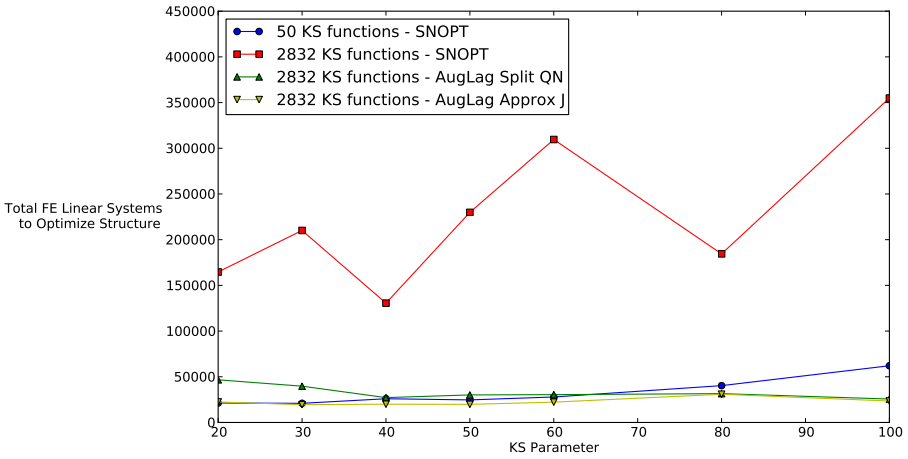

Figure 11. Computational work comparison between SNOPT and the matrix-free optimizers.

computational work by a full order of magnitude! For most values of ρ_{KS}, the matrix-free optimizer solving the formulation with 2832 constraints is cost-competitive with SNOPT solving the formulation with 50 constraints, albeit with a tighter convergence tolerance. The drastic reductions in cost observed in Figure 11 are caused by the fact that the matrix-free optimizer requires only a few linear systems to be solved at each iteration, while SNOPT and other SQP algorithms require the complete recomputation of the Jacobian.

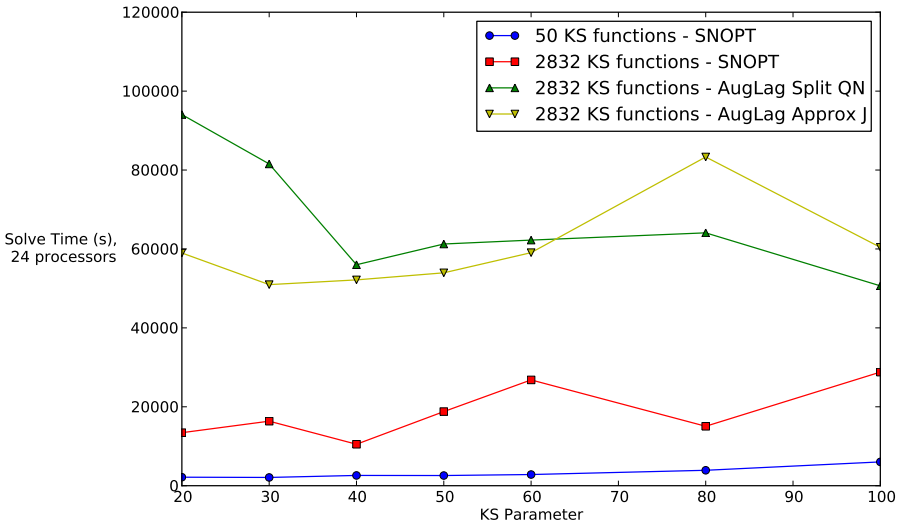

Figure 12. Run time comparison between SNOPT and the matrix-free optimizers using 24 processors.

While the cost reductions of Figure 11 look extremely promising, they are not reflected in the run time required for the optimization. Figure 12 shows this comparison. Clearly, even for a large number of constraints, SNOPT can solve the problem in less time than the matrix-free optimizer. It is tempting to attribute this behavior to the optimizer implementation, given that our optimizer is written in Python while SNOPT is written in Fortran. However, a breakdown of the run time shows that only 5-25% of the run time is actually spent in the matrix-free optimizer. Table 1 shows the results for the hard case of $\rho_{KS} = 100$ and the maximum number of stress constraints. Even if the matrix-free optimizer implementation were ten times faster, it would still not be enough to catch SNOPT.

We believe the discrepancy between Figures 11 and 12 is due to the number of iterations taken by

Table 1. Optimization statistics for the case of $\rho_{KS} = 100$, 2832 constraints.

	SNOPT	AugLag + Approx J	AugLag + Split QN
Wall Time (hours)	8.00	16.78	14.07
Optimizer Wall-Time Fraction	12.7%	25.9%	5.5%
Iterations	119	1675	2409
c calls	471	1913	3220
J calls	125	-	-
J products	-	3091	24
J^T products	-	6746	9574
Total FE solves	354 942	23 500	25 636

each optimizer and the design of the TACS solver. Table 1 lists statistics for the number of iterations and function calls taken by each optimizer. SNOPT and other SQP algorithms generally require a small number of high-cost iterations to solve the problem. In this wing design problem, the cost is amplified by the need to recompute the constraint Jacobian at each iteration. However, to solve the finite element linear systems, TACS uses a specialized direct factorization method and can reuse the matrix factors for multiple right-hand sides. This means that while computing the full Jacobian is expensive, it is much less than 2832 times more expensive than evaluating the stress constraints. Because the matrix-free optimizer only requires a few direct or adjoint solves at each iteration, it cannot reuse the matrix factors for each new design point nearly as much as SNOPT. Therefore, the overhead cost of changing design points causes the matrix-free algorithm to require more time than SNOPT.

Based on our current results, we envision two scenarios in which the matrix-free approach can be competitive in terms of run time. The first is that iterative methods are used to perform the structural analysis rather than direct methods. Under this scenario, the computation of the full Jacobian is less attractive because the advantage of reusing a full matrix factorization is reduced to reusing a preconditioner for each right-hand side. The second scenario involves developing a matrix-free SQP solver. It is well-known within the nonlinear optimization community that, when close to an optimal solution, SQP methods converge more quickly than the augmented Lagrangian method. If a matrix-free SQP method were developed, this method would likely require fewer iterations than the augmented Lagrangian method and the total overhead cost would be reduced.

VI. Conclusions and Outlook

In this work we have outlined a strategy for performing structural design optimization without explicitly computing the full constraint Jacobian. Our strategy is applicable in cases where there are a large number of design variables and a large number of expensive design constraints in the problem. In particular, we also showed how this approach may be used to alleviate the shortcomings of constraint aggregation. Our test results show that optimal solutions to design problems with thousands of variables and constraints can be computed using a matrix-free optimization algorithm. The computational cost of this approach is competitive with using constraint aggregation to reduce the number of constraints to a few dozen. However, because the matrix-free algorithm works with the original, non-aggregated constraint set, the final designs are of a lower mass, while still satisfying all stress constraints.

We aim to extend this approach to testing aerostructural design problems. The main difference between structural and aerostructural design is the complexity of the multidisciplinary analysis and the size of the resulting linear systems in the direct and adjoint gradient computation methods. In general, it will not be possible to perform direct factorization on these linear systems because the load-displacement transfer operators may be too costly to compute explicitly. This is one of the scenarios under which we expect the matrix-free approach to be particularly attractive for multidisciplinary design optimization.

Acknowledgements

The authors wish to thank Sylvain Arreckx and Dominique Orban of École Polytechnique de Montréal for their help in setting up the optimizer in the NLPy package. Funding for this work was provided in part by a Postgraduate Scholarship from the Natural Science and Engineering Research Council of Canada. All HPC computations were performed on the SciNet GPC cluster at the University of Toronto.

References

[1]Kenway, G. K. W., Kennedy, G. J., and Martins, J. R. R. a., "Scalable Parallel Approach for High-Fidelity Steady-State Aeroelastic Analysis and Adjoint Derivative Computations," *AIAA Journal*, Vol. 52, No. 5, May 2014, pp. 935–951.

[2]Kenway, G. K. W. and Martins, J. R. R. a., "Multipoint High-Fidelity Aerostructural Optimization of a Transport Aircraft Configuration," *Journal of Aircraft*, Vol. 51, No. 1, Jan. 2014, pp. 144–160.

[3]Qiu, G. Y. and Li, X. S., "A note on the derivation of global stress constraints," *Structural and Multidisciplinary Optimization*, Vol. 40, 2010, pp. 625–628.

[4]Kreisselmeier, G. and Steinhauser, R., "Systematic Control Design by Optimizing a Vector Performance Indicator," *Symposium on Computer-Aided Design of Control Systems*, IFAC, Zurich, Switzerland, 1979, pp. 113–117.

[5]Kreisselmeier, G. and Steinhauser, R., "Application of Vector Performance Optimization to a Robust Control Loop Design for a Fighter Aircraft," *International Journal of Control*, Vol. 37, No. 2, Feb. 1983, pp. 251–284.

[6]Poon, N. M. K. and Martins, J. R. R. A., "An adaptive approach to constraint aggregation using adjoint sensitivity analysis," *Structural and Multidisciplinary Optimization*, Vol. 34, 2007, pp. 61–73.

[7]Akgün, M. A., Haftka, R. T., Wu, K. C., Walsh, J. L., and Garcelon, J. H., "Efficient Structural Optimization for Multiple Load Cases Using Adjoint Sensitivities," *AIAA Journal*, Vol. 39, No. 3, 2001, pp. 511–516.

[8]Balling, R. J. and Sobieszczanski-Sobieski, J., "Optimization of Coupled Systems: A Critical Overview of Approaches," *AIAA Journal*, Vol. 34, No. 1, 1996, pp. 6–17.

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

[10]Martins, J. R. R. A. and Lambe, A. B., "Multidisciplinary Design Optimization: Survey of Architectures," *AIAA Journal*, Vol. 51, No. 9, 2013, pp. 2049–2075.

[11]Martins, J. R. R. A. and Hwang, J. T., "Review and Unification of Methods for Computing Derivatives of Multidisciplinary Computational Models," *AIAA Journal*, Vol. 51, No. 11, Nov. 2013, pp. 2582–2599.

[12]Sobieszczanski-Sobieski, J., "Sensitivity of Complex, Internally Coupled Systems," *AIAA Journal*, Vol. 28, No. 1, April 1990, pp. 153–160.

[13]Martins, J. R. R. A., Alonso, J. J., and Reuther, J. J., "A Coupled-Adjoint Sensitivity Analysis Method for High-Fidelity Aero-Structural Design," *Optimization and Engineering*, Vol. 6, No. 1, March 2005, pp. 33–62.

[14]Conn, A. R., Gould, N. I. M., and Toint, P. L., *LANCELOT: a Fortran package for large-scale nonlinear optimization (release A)*, Springer-Verlag, Berlin, 1992.

[15]Bertsekas, D. P., *Constrained Optimization and Lagrange Multiplier Methods*, Athena Scientific, 1996.

[16]Nocedal, J. and Wright, S. J., *Numerical Optimization*, Springer-Verlag, 2nd ed., 2006.

[17]Arreckx, S., Lambe, A. B., Martins, J. R. R. A., and Orban, D., "Implementation of a Matrix-Free Augmented Lagrangian Algorithm with Application to Aircraft Structural Design," Tech. rep., 2014.

[18]Moré, J. J. and Toraldo, G., "On the Solution of Large Quadratic Programming Problems with Bound Constraints," *SIAM Journal on Optimization*, Vol. 1, No. 1, 1991, pp. 93–113.

[19]Conn, A. R., Gould, N. I. M., and Toint, P. L., *Trust Region Methods*, SIAM, Philadelphia, PA, 2000.

[20]Byrd, R. H., Nocedal, J., and Schnabel, R. B., "Representations of quasi-Newton matrices and their use in limited memory methods," *Mathematical Programming*, Vol. 63, 1994, pp. 129–156.

[21]Bourji, S. K. and Walker, H. F., "Least-Change Secant Updates of Nonsquare Matrices," *SIAM Journal on Numerical Analysis*, Vol. 27, No. 5, 1990, pp. 1263–1294.

[22]Griewank, A. and Walther, A., "On Constrained Optimization by Adjoint based Quasi-Newton Methods," *Optimization Methods and Software*, Vol. 17, 2002, pp. 869–889.

[23]Schlenkrich, S., Griewank, A., and Walther, A., "On the local convergence of adjoint Broyden methods," *Mathematical Programming*, Vol. 121, 2010, pp. 221–247.

[24]Kennedy, G. J. and Martins, J. R. R. A., "A parallel finite-element framework for large-scale gradient-based design optimization of high-performance structures," *Finite Elements in Analysis and Design*, Vol. 87, Sept. 2014, pp. 56–73.

[25]Kennedy, G. J. and Martins, J. R. R. A., "A parallel aerostructural optimization framework for aircraft design studies," *Structural and Multidisciplinary Optimization*, 2014.

[26]Brown, S., "Displacement extrapolation for CFD+CSM aeroelastic analysis," Tech. rep., AIAA Paper 97-1090, 1997.

[27]Orban, D., "NLPy - a large-scale optimization toolkit in Python," Tech. rep., GERAD, 2014.

[28]Gill, P. E., Murray, W., and Saunders, M. A., "SNOPT: An SQP Algorithm for Large-Scale Constrained Optimization," *SIAM Journal on Optimization*, Vol. 12, No. 4, 2002, pp. 979–1006.

[29]Perez, R. E., Jansen, P. W., and Martins, J. R. R. A., "pyOpt: A Python-Based Object-Oriented Framework for Nonlinear Constrained Optimization," *Structural and Multidisciplinary Optimization*, Vol. 45, No. 1, 2012, pp. 101–118.

[30]Raspanti, C. G., Bandoni, J. A., and Biegler, L. T., "New strategies for flexibility analysis and design under uncertainty," *Computers & Chemical Engineering*, Vol. 24, Oct. 2000, pp. 2193–2209.