GERAD

GROUPE D'ÉTUDES ET DE RECHERCHE
EN ANALYSE DES DÉCISIONS

Les Cahiers du GERAD

G-2014-71
September 2014

CITATION ORIGINALE / ORIGINAL CITATION

A matrix-free augmented Lagrangian algorithm with application to large-scale structural design optimization. Optimization and Engineering, 1-26, 2015, dx.doi.org/10.1007/s11081-015-9287-9.

GERAD HEC Montréal
3000, ch. de la Côte-Sainte-Catherine
Montréal (Québec) Canada H3T 2A7

Tél. : 514 340-6053
Téléc. : 514 340-5665
info@gerad.ca
www.gerad.ca

Les Cahiers du GERAD ISSN: 0711–2440

A matrix-free augmented Lagrangian algorithm with application to large-scale structural design optimization

S. Arreckx, A. Lambe,
J.R.R.A. Martins, D. Orban

G–2014–71

September 2014

GERAD HEC Montréal
3000, chemin de la Côte-Sainte-Catherine
Montréal (Québec) Canada H3T 2A7

Tél. : 514 340-6053
Téléc. : 514 340-5665
info@gerad.ca
www.gerad.ca

A matrix-free augmented Lagrangian algorithm with application to large-scale structural design optimization

Sylvain Arreckx[a]

Andrew Lambe[b]

Joaquim R.R.A. Martins[c]

Dominique Orban[a]

[a] *GERAD & Department of Mathematics and Industrial Engineering, Polytechnique Montréal, Montréal (Québec) Canada H3C 3A7*

[b] *University of Toronto Institute for Aerospace Studies, Toronto (Ontario) Canada M3H 5T6*

[c] *Department of Aerospace Engineering, University of Michigan, Ann Arbor, MI, U.S.A.*

sylvain.arreckx@gerad.ca
andrew.lambe@mail.utoronto.ca
jrram@umich.edu
dominique.orban@gerad.ca

September 2014

Les Cahiers du GERAD

G–2014–71

Abstract: In many large engineering design problems, it is not computationally feasible or realistic to store Jacobians or Hessians explicitly. Matrix-free implementations of standard optimization methods—that do not explicitly form Jacobians and Hessians, and possibly use quasi-Newton derivative approximations—circumvent those restrictions but such implementations are virtually non-existent. We develop a matrix-free augmented-Lagrangian algorithm for nonconvex problems with both equality and inequality constraints. Our implementation is developed in the Python language, is available as an open-source package, and allows for approximating Hessian and Jacobian information. We show that it is competitive with existing state-of-the-art solvers on the CUTEr (Gould et al., 2003) and COPS (Bondarenko et al.) collections. We report numerical results on a structural design problem inspired by aircraft wing design. The matrix-free approach makes solving problems with thousands of design variables and constraints tractable, even if the function and gradient evaluations are costly.

Résumé : Dans de nombreuses applications réelles d'ingénierie, il est impossible de stocker les Jacobiens ou les Hessiens de manière explicite. L'implémentation de méthodes sans matrice pour les algorithmes d'optimisation standard, c'est-à-dire qui ne forment pas explicitement les Jacobiens et les Hessiens et qui peuvent utiliser des approximations quasi-Newton des dérivées, contournent ces restrictions. Cependant, de telles implémentations sont quasi inexistantes. Dans cet article, nous présentons un algorithme sans matrice basé sur le Lagrangien augmenté pour des problèmes d'optimisation non convexes possédant des contraintes générales d'égalité et d'inégalité. Celui-ci permet l'approximation du Jacobien et du Hessien. Notre implémentation est écrite en Python et est disponible gratuitement à tous. Nous montrons que notre algorithme est compétitif avec des solveurs réputés sur les collections CUTEr (Gould et al., 2003) et COPS (Bondarenko et al.). Nous présentons des résultats numériques sur un problème de conception structurelle inspiré de la conception d'une aile d'avion. L'approche sans matrice permet alors de résoudre des problèmes avec des milliers de variables et de contraintes, même lorsque les évaluations de fonctions et gradients sont coûteuses.

Acknowledgments: We would like to thank Graeme Kennedy for his assistance in setting up the structural design problem shown in this work. The computations for that problem were performed on the General Purpose Cluster supercomputer at the SciNet HPC Consortium.

1 Introduction

Aerospace engineered systems have been a prime target for the application of numerical optimization due to the large impact that weight reduction has on system performance. This is evident in the fuel mass required to launch a satellite into orbit and in the operating cost of modern transport aircraft, where the primary cost driver is the price of fuel.

One of the first such applications by aerospace engineers was structural design optimization, first proposed by Schmit (1960). The field was made possible by the advent of the finite element method for structural analysis (Argyris, 1954; Turner et al., 1956), which enabled engineers to analyze much more complex geometries than was possible with analytic methods.

This work is motivated by aircraft wing design optimization using coupled, high-fidelity physics-based models of aerodynamics and structures (Kenway et al., 2014; Kenway and Martins, 2014). In such problems, the objective, constraints, and derivative evaluations are expensive because of the expense of the aerodynamic and structural analyses. We seek optimization solvers that can rapidly converge with few function and derivative evaluations.

The design optimization problem of interest can be stated in the general form

$$\underset{x\in\mathbb{R}^n}{\text{minimize}} \;\; f(x) \quad \text{subject to} \;\; c(x) = 0, \;\; \ell \leq x \leq u, \tag{NLP}$$

where $f : \mathbb{R}^n \to \mathbb{R}$ and $c : \mathbb{R}^n \to \mathbb{R}^m$ are twice continuously differentiable. For the time being, it is sufficient to note that any nonlinear program may be reformulated as (NLP).

Kennedy and Martins (2010) and Kenway and Martins (2014) solve aircraft design problems based on (NLP) using general-purpose Sequential Quadratic Programming (SQP) software, such as SNOPT (Gill et al., 2002). This approach is particularly effective when used in conjunction with the efficient adjoint method to compute first derivatives (Kenway and Martins, 2014; Hwang et al., 2014; Lyu and Martins, 2014).

Structural design optimization problems often include both a large number of constraints (e.g., a failure criterion for each structural element), and a large number of variables (e.g., the thickness of each structural element). In addition the constraint Jacobian is typically dense because the structures being optimized are statically indeterminate—the static equilibrium equations alone are not sufficient to compute the stress in each element of the structure. As a result, the stress placed on a given element not only depends on the properties of that element but also on how the load is transmitted throughout the structure. Thus, each failure constraint depends on many design variables.

To make a factorization-based SQP approach such as SNOPT feasible, Poon and Martins (2007) aggregate constraints. The technique is effective for solving problems with hundreds of structural failure constraints (Kenway and Martins, 2014; Kennedy and Martins, 2013) but causes the final structural mass to be overestimated because it underestimates the size of the feasible region (Poon and Martins, 2007).

There is a need for an optimization approach that does not require aggregation, yet is still computationally efficient in the presence of dense Jacobians, and a matrix-free approach is the natural choice. A matrix-free SQP method would compute steps as inexact minimizers of quadratic programs with both linear equality and bound constraints. A more intuitive and simpler candidate is the augmented-Lagrangian method, as it may be implemented as the approximate solution of a sequence of bound-constrained subproblems. The subproblem solutions are used to update estimates of the Lagrange multipliers for the constraints of (NLP). Direction-finding subproblems involve solving linear systems with a coefficient matrix of the form $H = B + \rho J^T J$, where $\rho > 0$ is a penalty parameter. Efficient iterative methods, typically variants on the conjugate-gradient method, are available for this type of system. Indeed if B is positive semi-definite and J has full row rank, H is symmetric and positive definite. Note that operator-vector products with H require operator-vector products with the constraint Jacobian and its adjoint, an operation that is often available in practical large-scale applications. The main disadvantage is that augmented-Lagrangian methods typically do not exhibit the favorable local convergence properties of SQP methods. However, work is under way to combine the advantages of both paradigms.

Augmented Lagrangian methods are a staple of the optimization library of numerical methods. It would be impossible to give a complete list of references here. We refer the reader to general textbooks such as (Bertsekas, 1982), (Conn et al., 2000) and (Nocedal and Wright, 2006) for a thorough literature review and a complete convergence analysis.

Our implementation is released as part of the open-source package NLPy (Orban, 2014), a programming environment for designing numerical optimization methods written in the Python programming language.

Few other implementations of the augmented Lagrangian method exist. Amongst them MINOS (Murtagh and Saunders, 1978, 2003), LANCELOT (Conn et al., 1992), and ALGENCAN (Andreani et al., 2005) are the most widely used. MINOS takes advantage of linear constraints in the problem and, like SNOPT, is a commercial product. LANCELOT is designed to exploit the group-partially separable structure of the objective and constraints in order to gain efficiency when dealing with large sparse problems, which makes the code arduous to modify. Although LANCELOT does not strictly require forming Jacobians, its implementation is not matrix free. The main algorithmic difference between LANCELOT and ALGENCAN resides in the way they handle inequalities. Contrary to LANCELOT, which replaces inequalities by equality constraints by way of slack variables, ALGENCAN keeps inequalities intact, and uses the Powell-Hestenes-Rockafellar (Rockafellar, 1973) augmented Lagrangian function, which leads to discontinuous second derivatives in the objective of the subproblems. ALGENCAN is a matrix-free implementation, and is a good candidate to compare with our own implementation.

Recently, Gawlik et al. (2012) developed a linearly-constrained augmented Lagrangian method for solving partial differential equation (PDE) constrained optimization problems as part as the Toolkit for Advanced Optimization (TAO) (Munson et al., 2012). Their matrix-free method only handles equality constraints—bound constraints are not considered—so it cannot handle our problem of interest.

We now introduce the notation used in the remainder of this paper. The i-th component of the vector x is x_i, whereas x^k or $x^{k,j}$ stands for the vector x at outer iteration k or inner iteration (k, j). Unless explicitly specified, the norm used in this paper is the infinity norm $\|x\|_\infty$. Define the Lagrangian

$$\mathcal{L}(x, \lambda) := f(x) + \lambda^T c(x), \tag{1}$$

where $\lambda \in \mathbb{R}^m$ is the current approximation to the vector of Lagrange multipliers associated to the equality constraints of (NLP). The augmented Lagrangian function is

$$\Phi(x; \lambda, \rho) := \mathcal{L}(x, \lambda) + \frac{1}{2}\rho \sum_{j=1}^{m} c_j^2(x). \tag{2}$$

We separate λ and ρ from x by a semicolon in the arguments of Φ to indicate that they are treated as parameters, and that Φ is really a function of the primal variables x. For future reference, note that

$$\nabla_{xx}\Phi(x; \lambda, \rho) = \nabla_{xx}\mathcal{L}(x, \lambda + \rho c(x)) + \rho J(x)^T J(x). \tag{3}$$

Finally, $P_\Omega(g)$ is the projection of the vector $g \in \mathbb{R}^n$ into the set of simple bounds

$$\Omega := \{x \in \mathbb{R}^n \mid \ell \leq x \leq u\},$$

that is, the closest point to x in Ω, and is defined componentwise as $P_\Omega(g)_i = \text{median}(\ell_i, g_i, u_i)$ for $i = 1, \ldots, n$.

The rest of this paper is organized as follows. Section 2 is devoted to a detailed description of our matrix-free algorithm and its implementation in the Python language. We provide numerical results on standard test problems in order to validate our implementation and to compare it to existing software. In Section 3 we explore in further detail the structural design optimization problem, and show the benefits of the matrix-free approach over SNOPT. Conclusions and future work are discussed in Section 4.

2 A matrix-free augmented Lagrangian

2.1 Algorithmic details

In this section, we briefly cover the algorithmic details of our augmented Lagrangian framework. Although the framework itself is standard and well known, the description allows us to highlight certain algorithmic choices and relate them to implementation specifics to be described in Section 2.2.

The k-th *outer iteration* of the augmented-Lagrangian algorithm consists in approximately solving the subproblem

$$\underset{x\in\mathbb{R}^n}{\text{minimize}}\ \Phi(x;\lambda^k,\rho^k) \quad \text{subject to}\ \ell \le x \le u, \tag{4}$$

for fixed values of λ^k and ρ^k. We enforce satisfaction of the bound constraints explicitly in the subproblem. Each subproblem solution is followed by updates to λ^k, ρ^k, and subproblem stopping tolerances. Those updates are typically based on the improvement in constraint violation achieved in the most recent subproblem. Algorithm 1 summarizes this process, and follows (Nocedal and Wright, 2006, Algorithm 17.4) and Conn et al. (1992). The parameter updates in Step 4 are classic and follow updates implemented in LANCELOT (Conn et al., 1992) and ALGENCAN (Andreani et al., 2005).

Algorithm 1 Outer Iteration

1: Initialize $x_0 \in \Omega$, $\lambda^0 \in \mathbb{R}^m$, $\rho^0 \ge 1$, $\omega^0 > 0$ and $\eta^0 > 0$. Choose stopping tolerances $\epsilon_g > 0$ and $\epsilon_c > 0$. Set $k = 0$.

2: If the stopping conditions

$$\|x^k - P_\Omega(x^k - \nabla_x \mathcal{L}(x^k,\lambda^k))\|_\infty \le \epsilon_g \quad \text{and} \quad \|c(x^k)\|_\infty \le \epsilon_c$$

are satisfied, terminate with (x^k,λ^k) as final solution. Otherwise continue to step 3.

3: Compute x^{k+1} by approximately solving (4) and stopping as soon as

$$\|x^{k+1} - P_\Omega(x^{k+1} - \nabla_x \Phi(x^{k+1};\lambda^k,\rho^k))\|_\infty \le \omega^k.$$

4: If $\|c(x^{k+1})\|_\infty \le \eta^k$, set

$$\lambda^{k+1} = \lambda^k + \rho^k c(x^{k+1}), \qquad \rho^{k+1} = \rho^k \tag{5}$$
$$\eta^{k+1} = \eta^k/(\rho^{k+1})^{0.9}, \qquad \omega^{k+1} = \omega^k/\rho^{k+1}. \tag{6}$$

Otherwise, set

$$\lambda^{k+1} = \lambda^k, \qquad \rho^{k+1} = 10\rho^k, \tag{7}$$
$$\eta^{k+1} = 0.1/(\rho^{k+1})^{0.1}, \qquad \omega^{k+1} = 1/\rho^{k+1}. \tag{8}$$

Increase k by 1 and return to step 2.

At every outer iteration, (4) must be solved efficiently. In our implementation, two options are available. The first option follows LANCELOT and uses the method of Moré and Toraldo (1989). The iterate at the j-th inner iteration corresponding to the k-th outer iteration will be denoted $x^{k,j}$. We begin by building a quadratic model q_j of Φ about $x^{k,j}$:

$$q_j(p) := \nabla_x \Phi(x^{k,j};\lambda^k,\rho^k)^T p + \frac{1}{2} p^T B^{k,j} p,$$

where $B^{k,j}$ is a symmetric approximation of $\nabla_{xx}\Phi(x^{k,j};\lambda^k,\rho^k)$ that need not be definite. Our implementation allows $B^{k,j}$ to be defined as limited-memory BFGS or SR1 approximations (Nocedal and Wright, 2006). The step p^j is then obtained as an approximate solution of the bound-constrained quadratic program

$$\underset{p\in\mathbb{R}^n}{\text{minimize}}\ q_j(p) \quad \text{subject to}\ \|p\|_\infty \le \Delta^j,\ \ell \le x^{k,j} + p \le u, \tag{9}$$

where $\Delta^j > 0$ is the current trust-region radius. The step p^j is accepted or rejected and the radius Δ^j is updated following the standard trust-region criterion (Conn et al., 2000). Algorithm 2 summarizes the main steps involved in the inner iteration.

Algorithm 2 Inner Iteration

1: Choose $\Delta^0 > 0$, $0 < \epsilon_1 \leq \epsilon_2 < 1$, and $0 < \gamma_1 < 1 < \gamma_2$. Set $j = 0$. Choose an initial guess $x^{k,0} \in \Omega$.
2: If
$$\|x^{k,j} - P_\Omega(x^{k,j} - \nabla\Phi(x^{k,j}; \lambda^k, \rho^k))\|_\infty \leq \omega^k,$$
terminate with $x^{k+1} := x^{k,j}$. Otherwise continue to Step 3.
3: Choose $B^{k,j}$ symmetric and compute p^j as an approximate solution of (9).
4: Compute $\Phi(x^{k,j} + p^j; \lambda^k, \rho^k)$ and define
$$r^j := \frac{\Phi(x^{k,j} + p^j; \lambda^k, \rho^k) - \Phi(x^{k,j}; \lambda^k, \rho^k)}{q^j(p^j)}.$$
If $r^j \geq \epsilon_1$, then set $x^{k,j+1} = x^{k,j} + p^j$, otherwise set $x^{k,j+1} = x^{k,j}$.
5: Update the trust-region radius
$$\Delta^{j+1} = \begin{cases} \gamma_1\|p^j\|_\infty & \text{if } r^j < \epsilon_1 \\ \Delta^j & \text{if } r^j \in [\epsilon_1, \epsilon_2) \\ \max\{\Delta^j, \gamma_2\|p^j\|_\infty\} & \text{otherwise.} \end{cases}$$
Increment j by one and return to step 2.

In Algorithm 2, the initial guess $x^{k,0}$ may be simply set to the current outer iterate x^k or to a better approximation if one is available. In Step 3, p^j is computed using an extension of the method of Moré and Toraldo (1991) to nonconvex quadratic programs. In contrast with the trust-region subproblem solver used in LANCELOT, the method of Moré and Toraldo (1991) allows the additions of many constraints at a time to the active-set estimate. This active-set method is divided into two stages. In the first stage, a projected gradient search is used to select a face of the feasible set of (9), i.e., the intersection of Ω with the trust region. This face acts as a prediction of the optimal active set of (9). In the second stage, a reduced quadratic program is formed by fixing the components of p that are at their bounds on the selected face. This reduced quadratic program is then solved approximately using the conjugate gradient method to yield a search direction. A projected line search is then performed along the latter direction to ensure satisfaction of the bound and trust-region constraints. Both the projected gradient search and the conjugate gradient algorithm are designed to terminate early and promote fast progress. If the binding set at the iterate resulting from the projected search along the conjugate gradient direction coincides with the face identified in the first stage, the conjugate gradient iterations are resumed to enforce further descent. We refer the reader to (Moré and Toraldo, 1991) for complete algorithmic details. In the case where negative curvature is detected during the conjugate gradient iterations, we follow this direction to the boundary of the feasible set and subsequently perform a projected line search.

In practice, several improvements related to the management of the trust region can increase the efficiency of Algorithm 2. Two such improvements turned out to be effective in our implementation. The first is a non-monotone descent strategy (Toint, 1997). As described, Algorithm 2 enforces a monotone descent in $\Phi(\cdot; \lambda^k, \rho^k)$. In a non-monotone trust-region algorithm, a trial point may be accepted even if it results in an increase in Φ. However, a sufficient decrease is required after a prescribed number of iterations.

The second improvement to Algorithm 2 is the simplified version of the backtracking strategy of Nocedal and Yuan (1998) described by Conn et al. (2000). If p^j is rejected at Step 4 of Algorithm 2, we perform an Armijo line search along p^j instead of recomputing a new trust-region step. We impose a maximum of five backtracking iterations. If the line search is unsuccessful, we accept the last trial point nevertheless in hopes of promoting progress at the next inner iteration.

The second option to solve (4) is to use an existing method for bound-constrained problems, and our method of choice for this task is TRON (Lin and Moré, 1998). TRON is an active-set method for bound-constrained problems that iteratively determines a current working set by way of a projected gradient method, and explores faces of the feasible set using a Newton / trust-region method. In its default implementation, TRON has the significant disadvantage that it requires the explicit Hessian in order to compute an incomplete Cholesky preconditioner used to speed up the conjugate gradient iterations. We modified TRON so that only Hessian-vector products are required. In this case, the incomplete Cholesky factorization is made impossible, and no preconditioner is used in the conjugate gradient iterations. However, this modification to the source code also allows us to use quasi-Newton approximations of the Hessian. NLPy now features both a Python interface to this modified version of the Fortran TRON code, as well as a pure Python version.

2.2 Implementation

We choose to implement our algorithm in the Python language as part of the NLPy programming environment for linear and nonlinear optimization optimization (Orban, 2014). Optimization problems are only accessed to evaluate the objective and its gradient, and to compute operator-vector products with the Hessian of $\mathcal{L}(x,\lambda)$ and the constraints Jacobian. Our implementation, named AUGLAG, is open source, written in Python and available at `https://github.com/dpo/nlpy`.

Our implementation accepts problems with a mixture of general equality and inequality constraints and transforms the latter into non-negativity cosntraints, i.e., $c_{\mathcal{E}}(x) = 0$ and $c_{\mathcal{I}}(x) \geq 0$. We subsequently add slack variables to (NLP) to obtain constraints of the form

$$c_{\mathcal{E}}(x) = 0,\ c_{\mathcal{I}}(x) - t = 0,\ t \geq 0,\ \ell \leq x \leq u.$$

The augmented Lagrangian (2) becomes $\Phi(x,t;\lambda,\rho) := f(x) + \lambda^T(c(x) - \tilde{t}) + \frac{1}{2}\rho\sum_{i=1}^{m}(c_i(x) - \tilde{t}_i)^2$, where $\tilde{t} := (0,t) \in \mathbb{R}^{|\mathcal{E}|} \times \mathbb{R}^{|\mathcal{I}|}$, $m = |\mathcal{E}| + |\mathcal{I}|$. The latter augmented Lagrangian is iteratively minimized subject to the combination of the bounds $t \geq 0$, $\ell \leq x \leq u$, and the trust-region constraint $\|(x,t)\|_\infty \leq \Delta$. In the presence of inequalities, $\Phi(x,\cdot;\lambda,\rho)$ is a convex quadratic function of t. Every time Algorithm 2 identifies a new inner iterate $(x^{k,j}, t^{k,j})$, we may further minimize Φ in t subject to $t \geq 0$. This yields the *magical step* (Conn et al., 1999, 2000)

$$t_i := \max\left(0, \frac{\lambda_i}{\rho} + c_i(x^{k,j})\right), \quad i \in \mathcal{I}.$$

The user must provide first derivatives. Second derivatives may be provided if they are available. However, in some applications, such as the one described in Section 3 the Hessian of the augmented Lagrangian cannot be computed even in the form of Hessian-vector products, and we must be content with quasi-Newton approximations. Following the notation of Martínez (1988), the Broyden class of secant updates can be written as

$$B^{k+1} = B^k + \Delta_2(s, y, B^k, v) \tag{10}$$

where B^k and B^{k+1} are the current and updated approximations, respectively,

$$\Delta_2(s,y,B,v) = \frac{(y - Bs)v^T + v(y - Bs)^T}{v^T s} - \frac{(y - Bs)^T s}{(v^T s)^2} vv^T, \tag{11}$$

and

$$\begin{aligned} s &:= (x^{k,j+1}, t^{k,j+1}) - (x^{k,j}, t^{k,j}), \\ y &:= \nabla\mathcal{L}(x^{k,j+1}, t^{k,j+1}; \lambda^k) - \nabla\mathcal{L}(x^{k,j}, t^{k,j}; \lambda^k), \end{aligned}$$

for some choice of the vector $v \in \mathbb{R}^n$, called the *scale* of the update.

Approximating (3) as a monolithic Hessian without exploiting its structure leads to poor numerical behavior. Because we assume that exact first derivatives are available, products with $J(x)$ and $J(x)^T$ may be evaluated, and it remains to approximate the Hessian of (1), as first suggested by Dennis and Walker

(1981) in the context of nonlinear least-squares problems using the DFP secant method. Martínez (1988) generalized this DFP Hessian approximation to the Broyden class of secant methods, in particular for BFGS and SR1. The structured quasi-Newton approximation takes the form

$$B^k := S_k + \rho_k J(x^k)^T J(x^k), \qquad k = 0, 1, \ldots,$$

where we seek an update of $S_k \approx \nabla_{xx}\mathcal{L}(x^k, t^k, \lambda^k)$, such that B^{k+1} satisfies the secant equation

$$B^{k+1}s_k = (S_{k+1} + \rho_{k+1}J(x^{k+1})^T J(x^{k+1}))s_k = y_k. \tag{12}$$

Consider the intermediate secant equation $S^{k+1}s_k = y^\sharp$, where $y^\sharp$ is to be determined. From (12), we have

$$y_k = y^\sharp + \rho_{k+1}J(x^{k+1})^T J(x^{k+1}))s_k.$$

The secant update of B^{k+1} is then defined as

$$B^{k+1} = S_k + \Delta_2(s, y^\sharp, S_k, v(s, y_k, B^k)) + \rho_{k+1}J(x^{k+1})^T J(x^{k+1}).$$

It now seems reasonable to define

$$S_{k+1} = S_k + \Delta_2(s, y^\sharp, S_k, v(s, y_k, B^k)).$$

There remains to specify our choice of $y^\sharp$. It is not difficult to verify that the choice

$$\begin{aligned} y^\sharp &= S^{k+1}s_k \\ &= \nabla_{xx}f(x^{k+1})s_k + \sum_{i=1}^{m} \nabla_{xx}c_i(x^{k+1})[\lambda_i^{k+1} + \rho_{k+1}c_i(x^{k+1})]s_k \\ &= \nabla f(x^{k+1}) - \nabla f(x^k) + \left[J(x^{k+1}) - J(x^k)\right]^T [\lambda_i^{k+1} + \rho_{k+1}c_i(x^{k+1})] \end{aligned}$$

satisfies the intermediate secant equation.

2.3 Benchmarks

The numerical results have been obtained on a MacBook Pro with a 2.4 GHz microprocessor and 4 GB of memory running Mac OSX Lion. We report our results using performance profiles (Dolan and Moré, 2002).

We first present a comparison of our inner solver, SBMIN, versus the bound-constrained optimization code TRON (Lin and Moré, 1998) on all the problems from the COPS 3.0 collection (Dolan et al., 2004) and from the CUTEr collection (Gould et al., 2003) with at least one bound constraint. This results in 255 problems, all of which were used in their default dimension. Each problem is given a limit of 3000 iterations and 1 hour of CPU time.

By default TRON terminates the iterations as soon as

$$\|x^k - P_\Omega(x^k - \nabla f(x^k))\|_2 \leq 10^{-7}\,\|x^0 - P_\Omega(x^0 - \nabla f(x^0))\|_2.$$

In order to make a fair comparison between the two solvers, we adjusted TRON's stopping criterion such that SBMIN and TRON stop as soon as the relative *infinity* norm of the projected gradient is below 10^{-7}. For both algorithm, the initial trust region radius is set to

$$\Delta_0 = \tfrac{1}{10}\|x^0 - P_\Omega(x^0 - \nabla f(x^0))\|_\infty.$$

Figure 1 shows performance profiles in terms of number of iterations and of Hessian-vector products. The results indicate that TRON is slightly more robust than SBMIN, but requires substantially less iterations and Hessian-vector products to converge. In this regard, it appears that treating the bound constraints at a high level, as in TRON, instead of at the lowest level, as does SBMIN, pays off in terms of efficiency.

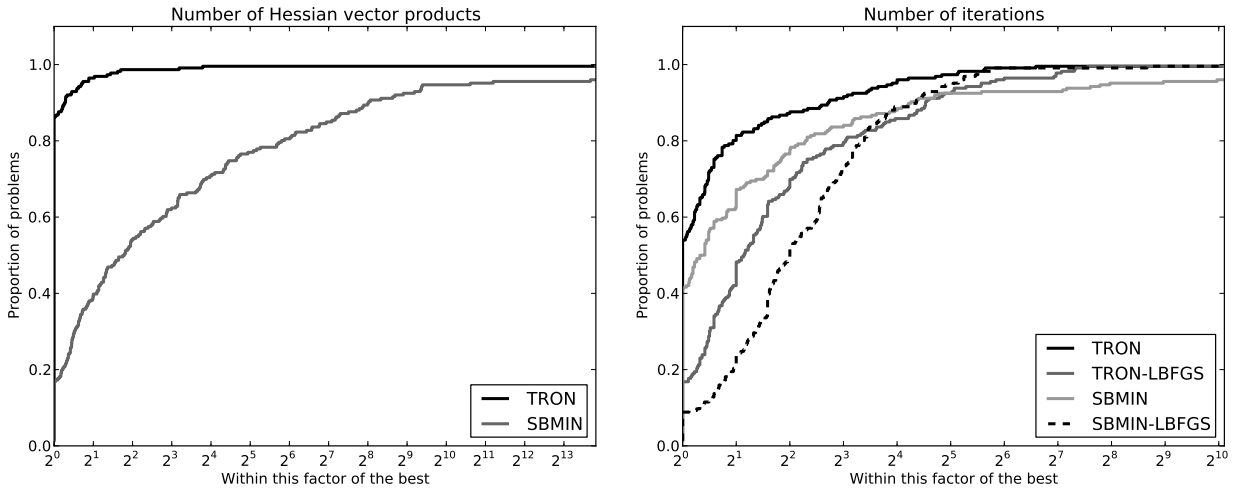

Figure 1: Comparison between SBMIN and TRON in terms of number of iterations and Hessian-vector products with exact second derivatives.

We now compare the two variants of our matrix-free augmented-Lagrangian implementation AUGLAG, one using SBMIN as inner solver (AUGLAG-SBMIN) and the other one using TRON (AUGLAG-TRON), to LANCELOT A (Conn et al., 1992). For simplicity of notation, x denotes all the variables, including slack variables, if any.

Because of the matrix-free nature of our algorithm and in order to do fair comparisons, partial group separability is disabled in LANCELOT, we use a box trust region, and disable the preconditioner in the conjugate gradient method. Furthermore, the Cauchy point calculation option was set to "approximate". All other options are set to their default values. Finally, for both solvers, the relative stopping tolerances, on the infinity norm of the projected gradient and constraint violation, are set to 10^{-7}. The initial trust region radius is set to

$$\Delta_0 = \tfrac{1}{10}\|x^0 - P_\Omega(x^0 - \nabla\Phi(x^0; \lambda^0, \rho^0))\|_\infty,$$

where λ^0 is a least-square estimate of the Lagrange multipliers. We set $\epsilon_1 = 10^{-4}$, $\epsilon_2 = 0.9$, $\gamma_1 = 0.25$ and $\gamma_2 = 2.5$. These values have been chosen because they seem to produce overall good performance compared to other values we have explored.

When limited-memory quasi-Newton approximations of the Hessian are employed, all three optimization codes are run with the same number of pairs in the history: 3 for L-BFGS, and 5 for L-SR1. The automatic problem scaling procedure available in NLPy is disabled for the AUGLAG results because the other codes do not perform any scaling of the problem.

Finally, we compare the algorithms on all problems from the COPS 3.0 collection (Dolan et al., 2004) and from the CUTEr collection (Gould et al., 2003) which possess at least one equality constraint or at least one bound constraint. This amounts to 675 problems. Again, a CPU time limit of 1 hour and an iteration count limit of 3000 is imposed. Figure 2 summarizes the performance of LANCELOT A and AUGLAG. This figure only shows the number of iterations because the LANCELOT A interface doesn't provide the number of Hessian-vector products made during the optimization. The results indicate that AUGLAG-TRON is more robust than the two other codes when using either exact Hessian or quasi-Newton approximations. Both versions of AUGLAG perform slightly better than LANCELOT A when using exact derivatives but LANCELOT A with LSR1 seems to perform equilavently to the two other codes.

3 Structural design optimization application

We now turn to a particular area of application for our matrix-free algorithm: aircraft structural design. Reducing the structural weight improves the fuel efficiency of the aircraft and therefore influences both the

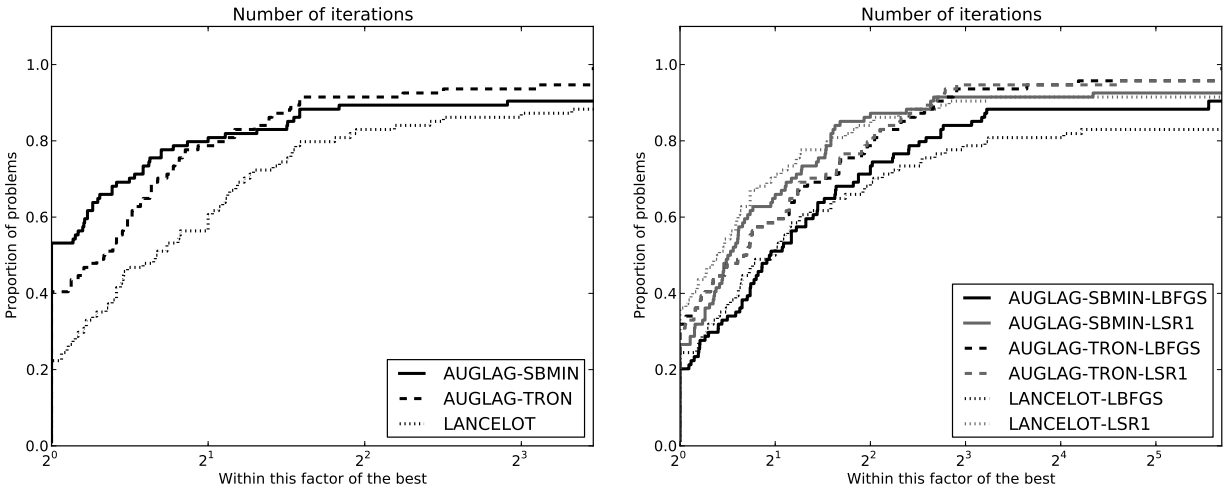

Figure 2: Comparison between AUGLAG-SBMIN, AUGLAG-TRON and LANCELOT A in terms of number of iterations with exact second derivatives (left) and with Quasi-Newton approximations (right).

operating cost to the airline and the environmental impact of air transportation. Our goal is to minimize the mass of the structure subject to failure constraints. While many structural optimization problems are formulated with compliance (strain energy) constraints, the resulting solutions often show stress concentrations that would result in failure if the real structure were designed in that way. Therefore, optimization subject to failure constraints is more practical from an engineering design perspective. We start by describing the optimization problem formulation and how a matrix-free optimizer is helpful in this case before discussing the structural design optimization results.

3.1 Problem formulation and derivative evaluations

Structural analysis involves the solution of static equilibrium equations in the form of a discretized PDE so this problem may be interpreted as a special case of PDE-constrained optimization. However, the stress constraints place further restrictions on the optimal set of state variables, and eliminating the discretized PDEs does not eliminate all of the constraints involving state variables. The full-space (Biros and Ghattas, 2005) or *simultaneous analysis and design* (SAND) problem (Haftka and Kamat, 1989) is stated as

$$\underset{x,y}{\text{minimize}} \;\; F(x,y) \quad \text{subject to} \;\; C(x,y) \leq 0, \;\; R(x,y) = 0, \;\; \ell \leq x \leq u, \tag{SAND}$$

where $x \in \mathbb{R}^N$ are the design variables, $y \in \mathbb{R}^M$ are the state variables, and $R : \mathbb{R}^N \times \mathbb{R}^M \to \mathbb{R}^M$ are the discretized PDEs. Because we often use specialized software to solve the governing PDEs, and because N is usually much smaller than M, an alternative is to solve the *nested analysis and design* (NAND) problem (Biros and Ghattas, 2005)

$$\underset{x}{\text{minimize}} \;\; f(x) \quad \text{subject to} \;\; c(x) \leq 0, \;\; \ell \leq x \leq u, \tag{NAND}$$

where $y(x)$ is defined implicitly via $R(x, y(x)) = 0$, $f(x) := F(x, y(x))$, and $c(x) := C(x, y(x))$. Despite its smaller size, even (NAND) can have thousands of variables and constraints. Furthermore, the governing equations $R(x, y(x)) = 0$ must be re-solved for each new point computed by the optimizer, making function and gradient evaluation expensive. The chain rule and the implicit function theorem yield

$$\nabla c(x) = \nabla_x C(x, y(x)) + \nabla_x y(x) \nabla_y C(x, y(x)) \tag{13}$$

$$= \nabla_x C(x, y(x)) + \nabla_x R(x, y(x)) \nabla_y R(x, y(x))^{-1} \nabla_y C(x, y(x)) \tag{14}$$

where $\nabla_x C(x, y(x))$ denotes the transpose Jacobian of C with respect to x, i.e., the matrix whose columns are the gradients with respect to x of the component functions of C. We use a similar notation for the derivatives

of R, and use "-1" for the inverse. Each matrix-vector product with $\nabla c(x)$ and $\nabla c(x)^T$ involves solving a linear system with coefficient matrix $\nabla_y R(x, y(x))$ and $\nabla_y R(x, y(x))^T$, respectively. Because both operations involve the solution of a large system of linear equations, the computational cost of a single matrix-vector product is similar to the cost of evaluating all the objective and constraint functions. Therefore, the success of the matrix-free approach for solving problem (NAND) hinges on keeping the sum of function evaluations and matrix-vector products small.

3.2 Approximating Jacobian information

As mentioned in Section 2.2, exploiting the structure of the Hessian of the augmented Lagrangian leads to better convergence results on a wide range of problems. In particular, computing exact Jacobian-vector products within the trust-region solver and using a structured Hessian approximation to estimate the remaining terms is an effective strategy. However, this strategy can be very expensive when applied to structural design problems. Every time a Hessian-vector product is computed in the trust-region solver, two products with the Jacobian (one forward and one transpose) are required. We have observed many instances in which the number of Jacobian-vector products needed to solve a given trust-region subproblem exceeds the number of constraints of the problem. Under these circumstances, if sufficient memory were available, it would be more efficient to form and store the entire Jacobian for computing these products than to compute the products from scratch. Therefore, we need to further refine the basic algorithm to reduce the number of expensive matrix-vector products.

We propose two different approaches for reducing the number of Jacobian-vector products in our matrix-free algorithm. Both approaches rely on using the Jacobian-vector products to create more accurate trust-region subproblem models using additional quasi-Newton matrix approximations. By using approximate Jacobian information in the trust-region subproblem, we prevent the number of Jacobian-vector products in any given iteration from becoming too large and keep the cost of solving the subproblem low. Note that exact Jacobian-vector products are still used to compute gradients of the Lagrangian and augmented Lagrangian function. Approximate Jacobian information is only used in the trust-region subproblem.

The first approach estimates the Hessian of quadratic penalty term of the augmented Lagrangian function separately from the Hessian of the Lagrangian. We refer to this approach as the "split" quasi-Newton method. We define $B_{\mathcal{L}} \approx \nabla^2_{xx}\mathcal{L}$ and $B_{\mathcal{I}} \approx \nabla^2_{xx}\frac{\rho}{2}c(x)^T c(x)$. The gradient of the infeasibility function is simply $\rho J(x)^T c(x)$ so constructing the Hessian approximation is straightforward using a single Jacobian-vector product. The limited-memory SR1 approximation is used for $B_{\mathcal{L}}$ while limited-memory BFGS is preferred for $B_{\mathcal{I}}$. To further improve the approximation provided by $B_{\mathcal{I}}$, we use a starting diagonal that is an approximation of the true diagonal of $J(x)^T J(x)$. The approximation is computed in the same way as the preconditioner proposed by De Simone and di Serafino (2013). Because both quasi-Newton approximations are limited-memory approximations, this approach is very memory-efficient for large optimization problems.

The second approach estimates the Jacobian matrix directly. In other words, we replace the true Jacobian-vector products for the algorithm outlined in Section 2.2 with the products of the same vectors with an approximate Jacobian matrix. In general, the Jacobian is not a square matrix, so alternative quasi-Newton approximations need to be used. Two such approximations are the two-sided rank-one (TR1) method, proposed by Griewank and Walther (2002), and the adjoint Broyden method, proposed by Schlenkrich et al. (2010). Because the TR1 method requires more frequent updates to the Lagrange multipliers than we have available in our algorithm, we have selected the adjoint Broyden method for implementation. Unfortunately, no convergence theory exists for limited-memory quasi-Newton Jacobian estimates and it is not obvious how to initiate a robust limited-memory approximation. Therefore, we have chosen to implement a full-memory version of this approximation.

The basic adjoint Broyden update is given by the formula

$$A_{k+1} = A_k + \frac{\sigma_k \sigma_k^T}{\sigma_k^T \sigma_k} \left(J(x_{k+1}) - A_k\right) \tag{15}$$

where A is the approximate Jacobian and σ is an "adjoint search direction." Note that this update requires at least one (adjoint) Jacobian-vector product. Unlike traditional quasi-Newton methods, the choice of the

search direction is not obvious and Schlenkrich et al. suggest several alternatives. We choose option (A) from (Schlenkrich et al., 2010), given by

$$\sigma_k = (J_{k+1} - A_k)s_k \tag{16}$$

where $s_k = x_{k+1} - x_k$, as the method to use with our algorithm. This particular choice of σ yields an update that is similar to the original TR1 update. Compared to the split quasi-Newton strategy, this strategy requires an additional Jacobian-vector product to compute σ_k. Despite the increase in required memory and higher cost of the update, this method has a distinct advantage over the split quasi-Newton approach in that the sparsity structure of any slack variables in the Hessian is preserved. That is, the block of $\nabla^2_{xx}\frac{\rho}{2}c(x)^T c(x)$ associated with the slack variables is known exactly (an identity matrix) so it may be treated exactly in the Hessian-vector product. This approach leads to a much more accurate Hessian approximation than the split quasi-Newton method if the problem contains many slack variables.

We close this section with a few implementation details of the adjoint Broyden method. Similar to other quasi-Newton schemes, we reject the update if the denominator of the update term in (15) is sufficiently small, i.e., if $\sigma^T\sigma \leq 10^{-20}$. Our initial approximation A_0 is set to be the exact Jacobian J_0. While this strategy has a very high up-front cost, we found that it paid off on our test problem in terms of many fewer major iterations required by the optimization. We recognize that our strategy may not be sound for all problems, especially those in which the constraints are highly nonlinear. However, we expect the approach to be sound given that a full-memory quasi-Newton approximation is able to converge to the true matrix after many iterations.

3.3 Optimization results

We use the following test problem to compare our matrix-free algorithm against an optimizer that requires the full Jacobian. The problem is to minimize the mass of a square, metallic plate that is clamped on all sides and subject to a uniform pressure load. The structural analysis of the plate is conducted using the finite-element program TACS (Kennedy and Martins, 2014) with third-order shell elements. The optimization problem is constrained so that the maximum von Mises stress on any of the plate elements does not exceed the material yield stress. The design variables of the problem are the thicknesses of each plate element. Minimum and maximum thicknesses are imposed on each element. The geometry of the plate is shown in Figure 3. To simplify the problem, we analyze only one quarter of the plate and apply symmetry boundary conditions on the unclamped edges. By construction, the problem has the same number of structural elements, design variables, and constraints for each test case. Except for the design variable bounds, all constraints are nonlinear.

While this test problem does not represent a complete aircraft structure, it shares two challenging features of such structures. First, the structure is a shell structure subject to a distributed load. This type of

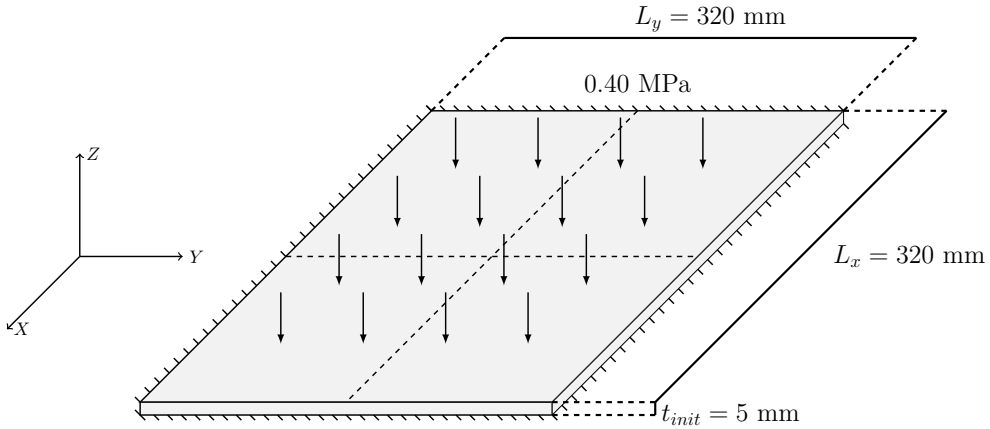

Figure 3: Geometry and load condition of plate mass minimization problem.

structure requires higher-order two- or three-dimensional finite elements to be used for accurate analysis of the structural behavior. The resulting analysis is therefore much more expensive than analyses using one-dimensional elements due to the larger number of degrees of freedom. Second, and more importantly, the structure is not statically determinate and has many degrees of indeterminacy. This means that the full finite-element analysis must be completed in order to compute stresses and strains; no shortcuts can be taken in evaluating the failure constraints. In practice, this finite-element analysis can be ill-conditioned so the NAND problem formulation (NAND) is used to hide the ill-conditioning from the optimizer.

Our benchmark optimizer for this test is the general purpose optimizer SNOPT (Gill et al., 2002) which is accessed in Python through the pyOpt interface (Perez et al., 2012). SNOPT is an active-set SQP optimizer capable of solving nonlinear and nonconvex problems. While the full version of SNOPT has no limits on the number of variables or constraints in the problem, it is especially suited to problems with a large number of sparse constraints and few degrees of freedom. Like our optimizer, SNOPT does not require second derivatives because it approximates them using a limited-memory quasi-Newton method. Unlike our optimizer, SNOPT requires first derivative information from the objective and all constraint functions. Our optimizer just requires the gradient of the objective function and forward and transpose products with the constraint Jacobian.

Due to the design of the TACS software, we are able to accommodate both traditional optimizers like SNOPT and matrix-free optimizers. For our expression for the Jacobian of the reduced-space problem in (14), TACS provides modules for computing the action of $[\nabla_x R(x, y(x))]$ and $[\nabla_x R(x, y(x))]^T$ on vectors of appropriate length. The different partial derivatives of the constraints themselves are computed with respect to individual constraints, effectively providing column-wise evaluation of $[\nabla_x C(x, y(x))]$ and $[\nabla_y C(x, y(x))]$. The term $[\nabla_y R(x, y(x))]^{-1}$ is computed implicitly by a specialized, sparse, parallel, direct factorization method. Every time we multiply this inverse or its transpose by a vector, we solve the appropriate upper- and lower-triangular systems by substitution. When computing the full Jacobian for SNOPT, TACS exploits parallel structure in the adjoint method to compute multiple adjoint vectors at the same time. This feature is not needed by the matrix-free optimizer since only individual matrix-vector products are ever called for. However, this added awareness of parallel computing does tend to skew the run-time results in favour of SNOPT.

We use the following settings in our matrix-free optimizer. The LSR1 Hessian approximation with five pairs of vectors is used to estimate the Hessian of the Lagrangian. The adjoint Broyden approximation is used to estimate the constraint Jacobian, where the initial Jacobian is computed exactly. In the split quasi-Newton strategy, the LBFGS approximation with five pairs of vectors is used to estimate the feasibility Hessian. Both magical steps and Nocedal–Yuan backtracking are turned on in the nonlinear, bound-constrained solver. In SBMIN, a limit of 50 iterations is imposed to avoid computing an exact solution to an inexact problem statement. (On this specific problem, we found that SBMIN was superior to TRON when the additional quasi-Newton matrices were used.) A damping factor is added to the basic Lagrange multiplier update to ensure that the multipliers do not become unrealistically large or small when far from the solution. This factor is calculated by minimizing the final value of $||\nabla \mathcal{L}||_2$ in the direction of the update. Finally, parallel computations are used in the adjoint Broyden approximation to allow the approximate Jacobian to be stored in a distributed fashion and reduce the memory requirements of the solver.

Example design solutions to the benchmark problem are shown in Figure 4 for three different mesh sizes. Corresponding stress distributions are presented in Figure 5. In these figures, the x- and y-axes of the plots represent the clamped edges of the plate. The built-up regions of the plate along the clamped edges and in the center of the plate are clearly visible. For every case in which both solvers found an optimal solution, both SNOPT and our solver converged on similar final designs. The feasibility and optimality tolerances of both solvers were set to 10^{-5}, and both solvers achieved these tolerances at the final designs.

Figure 6 compares the number of finite element linear systems—those involving $[\nabla_y R(x, y(x))]$—that are solved using each algorithm for a range of problem sizes. The finest mesh solved using either optimizer was 70 elements by 70 elements. The corresponding optimization problem had 4900 thickness variables and 4900 failure constraints. Figure 6 shows our primary metric for comparing the optimizers since solving the linear system associated with the finite element method is the most costly operation in the optimization process.

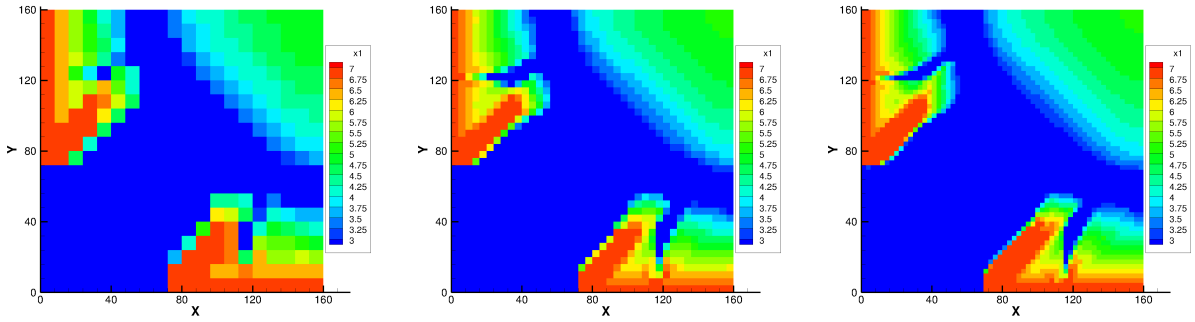

Figure 4: Final thickness distributions for the 400-, 1600-, and 3600-element plate problems. These solutions were all obtained by the matrix-free optimizer. The solutions from SNOPT for the 400- and 1600-element problems are nearly identical.

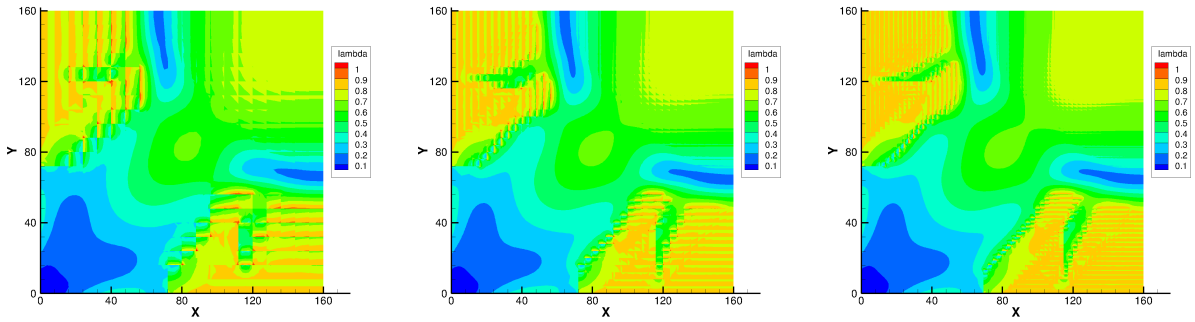

Figure 5: Stress distributions as a fraction of the local yield stress for the 400-, 1600-, and 3600-element plate problems.

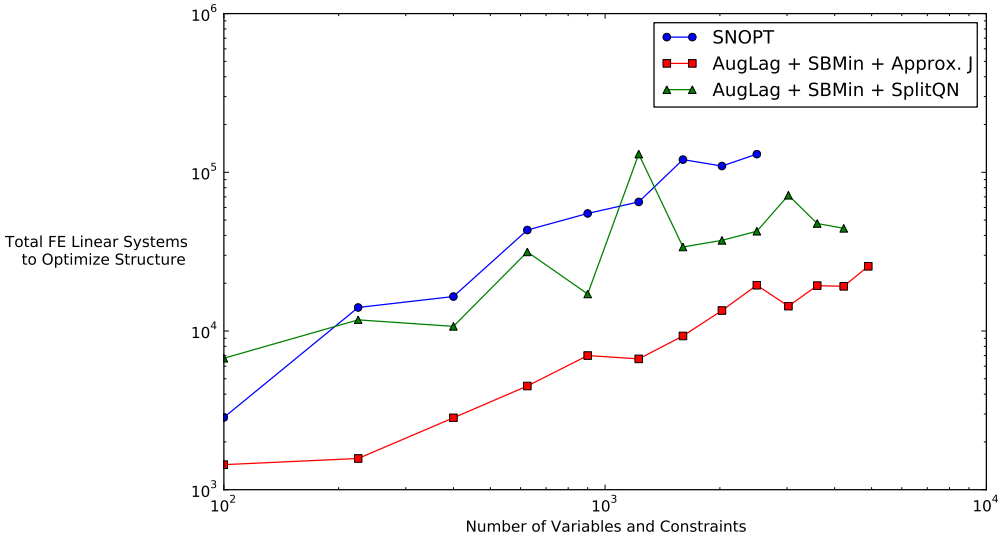

Figure 6: Number of finite-element linear solutions required to solve the plate optimization problem.

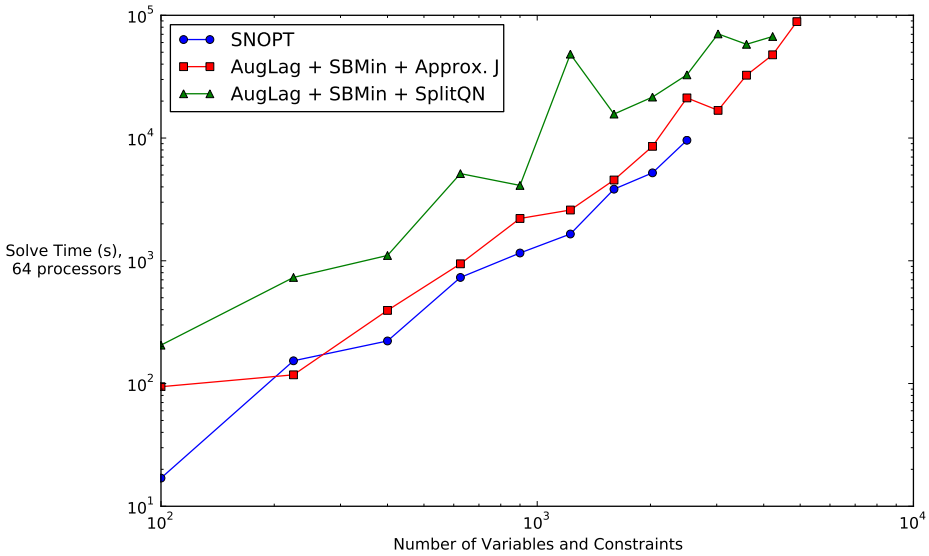

Figure 7: Run time to solve the plate optimization problem using 64 processors.

This operation occurs once to evaluate the failure constraints and once for every Jacobian-vector product. To form the entire Jacobian for SNOPT, a linear system is solved to obtain one column of the matrix so the matrix size determines the total work. Figure 6 demonstrates that, by not forming the Jacobian at each iteration, both matrix-free algorithms successfully reduce the number of expensive linear solve operations as the problem size increases. In fact, for problems with more than 1000 variables and constraints, the reduction produced by the approximate Jacobian approach is nearly an order of magnitude over SNOPT.

Figure 6 also shows that the matrix-free optimizer was able to solve larger optimization problems than SNOPT. SNOPT was unable to solve any problems for meshes larger than 50 elements by 50 elements due to a lack of memory. Each instance of the benchmark problem was solved in a distributed-memory computing environment. Because SNOPT was not designed to exploit this environment, it could only access the memory available to a single computing node, limiting its potential to solve large problems. In contrast, both matrix-free strategies were designed with this environment in mind. In the split quasi-Newton approach, only limited-memory Hessian approximations are used so the optimizer can solve very large problems given sufficient time. In the approximate Jacobian approach, Message Passing Interface (MPI) standard commands are added to the code, via the mpi4py library, to distribute the stored matrix across multiple nodes and compute matrix-vector products in parallel. Reducing the memory requirements of the optimizer is critical to solving large instances of the optimization problem.

Figure 7 presents a wall-time comparison for solving the optimization problems using 64 processors. Comparing Figures 6 and 7, the large reduction in linear system solve operations does not translate into reduced run time. In fact, SNOPT is still the fastest optimizer for those problem sizes which it is able to solve. We attribute this behaviour to two causes. First, as mentioned above, the TACS solver is able to parallelize the (implicit) multiplication of $[\nabla_y R(x, y(x))]^{-1}$ by multiple right hand sides, reducing the time needed to form a large Jacobian. Second, SNOPT requires many fewer iterations than our augmented Lagrangian solver to find the solution in each case. Fewer iterations means fewer points for which the partial derivative matrices must be recomputed. While this cost is small in comparison to the cost of a linear solve operation, the increase in the number of iterations seems to outweigh the reduction in linear solves for this choice of algorithm.

One implementation decision that does not exert too much influence on the run time is the choice of implementation language of the optimizer. Figure 8 shows a breakdown of the run time spent on each case evaluating the problem functions, evaluating the gradients, (including Jacobian-vector products,) and computing the next point in the optimizer. When using SNOPT, only a small fraction of the run time is spent

in the optimizer unless the problem is large. This increase in run time is probably due to the extra work needed to factorize the Jacobian in the active-set SQP algorithm. For the split quasi-Newton approach, the fraction of the run time spent in the optimizer decreases with increasing problem size. This result suggests that replacing the Python implementation of the algorithm with a compiled-language implementation would not result in large reductions in wall time. For the approximate Jacobian approach, the optimizer appears to take up the majority of the run time of the optimization process. However, nearly all of this time is spent forming matrix-vector products with the approximate Jacobian. Python makes use of both distributed-memory parallel processing and compiled-language libraries to complete this operation, so it is unlikely that moving to a compiled language implementation would result in a large reduction in run time.

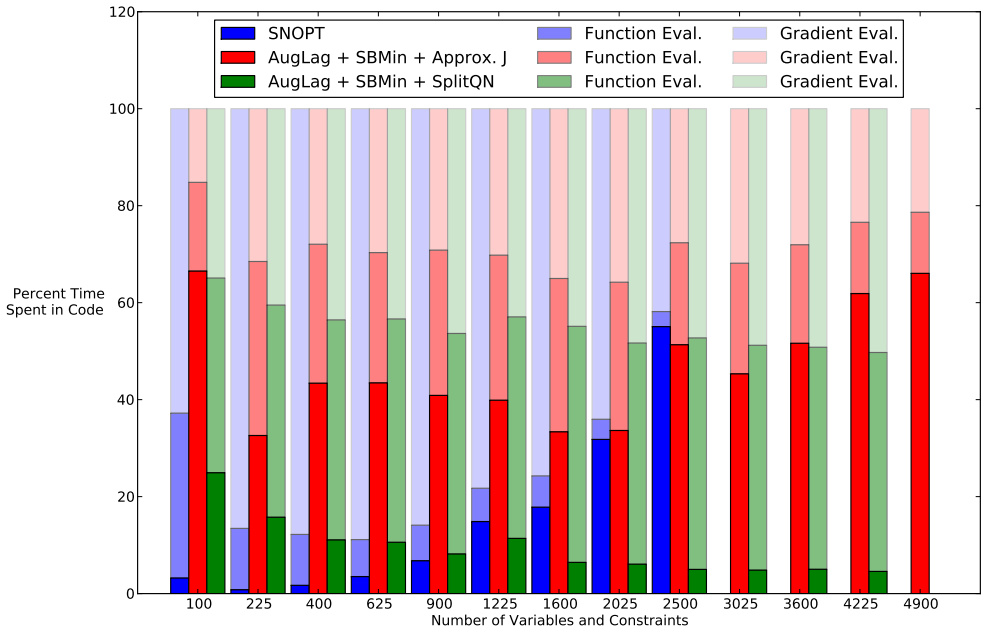

Figure 8: Breakdown of wall time spent in optimizer and computing function and gradient information.

These results effectively show the intrinsic trade-off of matrix-free optimization in engineering design applications. As demonstrated in Figure 6, if the engineering design problem has many constraints, using a matrix-free optimizer can lead to a massive reduction in the computational effort spent calculating gradient information. However, this reduction is offset by the overhead incurred by recomputing the design constraints and relevant partial derivative matrices at more points in the design space. We suspect that changing the basic optimization algorithm from an augmented Lagrangian to an SQP method should allow the matrix-free optimizer to be more competitive in terms of run time. If the problem of interest contains thousands of variables and constraints, using limited-memory approximations or exploiting distributed storage is essential to solving problems of this size.

4 Conclusion and future work

This work details the implementation of a matrix-free optimizer based on the augmented Lagrangian algorithm. Benchmarking results indicate that it is competitive with LANCELOT on standard test sets. We then extend the algorithm to store approximate Jacobian information to reduce the required number of matrix-vector products. The extended algorithm is then applied to a test problem motivated by aircraft structural design. Our results indicate that the matrix-free optimizer successfully reduces the computational work of the structural analysis when the structural design problem contains a large number of design variables and a large number of contraints. In the near future, we hope to extend these encouraging results to the design of aircraft wings, including coupled aerodynamic and structural optimization.

Given their robustness and convergence rate, it appears important to provide a matrix-free implementation of an SQP-type method. How to achieve this is not obvious for the following reasons. Let us assume for simplicity that the problem is stated in standard form and that the only inequality constraints are simple bounds. If the bounds are kept explicit as in SNOPT, each SQP subproblem is a quadratic program with both equality and inequality constraints, and it is not immediately apparent how to solve such problems efficiently using a matrix-free method, even if they are convex. If the bounds are enforced by way of a logarithmic barrier, as in, e.g., IPOPT (Wächter and Biegler, 2006) or KNITRO (Byrd et al., 2006), the subproblems are equality-constrained quadratic programs. KNITRO adds a trust-region constraint to those subproblems in order to guarantee global convergence and solves them by way of the projected conjugate-gradient method Gould et al. (2001). Unfortunately, the latter requires accurate projections into the nullspace of the linear equality constraints and this is best achieved if the Jacobian is explicitly available.

Arreckx and Orban (2014) describe a matrix-free implementation of a fully-regularized SQP-type method for equality-constrained problems related to that of Armand et al. (2012), and highlight its connections with the standard augmented-Lagrangian method. If a logarithmic barrier enforces satisfaction of the bound constraints, the method still appears to lend itself to a matrix-free implementation thanks to the regularization terms. Greif et al. (2014) propose various system formulations for the solution of regularized convex quadratic programs by way of an interior-point method and Orban (2013) illustrates the efficiency of limited-memory preconditioners combined with a full-space iterative solver such as MINRES. Arioli and Orban (2013) propose other families of iterative methods that are exploited by Arreckx and Orban (2014) and appear to be suitable in either linesearch or trust-region contexts.

References

R. Andreani, E.G. Birgin, J.M. Martínez, and M.L. Schuverdt. On augmented lagrangian methods with general lower-level constraints. Technical report, University of Campinas, Campinas SP, Brazil, 2005.

J.H. Argyris. Energy theorems and structural analysis: A generalized discourse with applications on energy principles of structural analysis including the effects of temperature and non-linear stress-strain relations. Aircraft Engineering and Aerospace Technology, 26(10):347–356,383–387,394, 1954. doi: 10.1108/eb032482.

M. Arioli and D. Orban. Iterative methods for symmetric quasi-definite linear systems—Part I: Theory. Les Cahiers du GERAD G–2013–32, HEC Montréal, Canada, 2013.

P. Armand, J. Benoist, and D. Orban. From global to local convergence of interior methods for nonlinear optimization. Optimization, Methods & Software, 28(5):1051–1080, 2012. doi: 10.1080/10556788.2012.668905.

S. Arreckx and D. Orban. A regularized SQP method for degenerate equality-constrained optimization. Technical report, GERAD, Montréal, Canada, 2014. In preparation.

D.P. Bertsekas. Constrained Optimization and Lagrange Multiplier Methods. Academic Press, 1982.

G. Biros and O. Ghattas. Parallel Lagrange-Newton-Krylov-Schur Methods for PDE-Constrained Optimization. Part I: the Krylov-Schur Solver. SIAM Journal on Scientific Computing, 27(2):687–713, 2005. doi: 10.1137/S106482750241565X.

A.S. Bondarenko, D.M. Bortz, and J.J. Moré. COPS: Large-scale nonlinearly constrained optimization problems. URL `http://www.mcs.anl.gov/~more/cops`.

R.H. Byrd, J. Nocedal, and R.A. Waltz. KNITRO: An integrated package for nonlinear optimization. In G. di Pillo and M. Roma, editors, Large-Scale Nonlinear Optimization, pages 35–59. Springer Verlag, 2006.

A.R. Conn, N.I.M. Gould, and P.L. Toint. Lancelot: A Fortran Package for Large-Scale Nonlinear Optimization (Release A). Springer Publishing Company, Incorporated, 1st edition, 1992.

A.R. Conn, L.N. Vicente, and C. Visweswariah. Two-step algorithms for nonlinear optimization with structured applications. SIAM Journal on Optimization, 9(4):924–947, 1999. doi: 10.1137/S1052623498334396.

A.R. Conn, N.I.M. Gould, and P.L. Toint. Trust-region methods. Society for Industrial and Applied Mathematics, Philadelphia, PA, USA, 2000.

V. De Simone and D. di Serafino. A matrix-free approach to build band preconditioners for large-scale bound-constrained optimization. Journal of Computational and Applied Mathematics, 268:82–92, 2014.

J.J. Dennis and H. Walker. Convergence theorems for least-change secant update methods. SIAM Journal on Numerical Analysis, 18(6):949–987, 1981. doi: 10.1137/0718067.

E.D. Dolan and J.J. Moré. Benchmarking optimization software with performance profiles. Mathematical Programming, 91(2):201–213, 2002. doi: 10.1007/s101070100263.

E.D. Dolan, J.J. Moré, and T.S. Munson. Benchmarking Optimization Software with COPS 3.0. Technical Report ANL/MCS-TM-273, Argonne National Laboratory, Mathematics and Computer Science Division, 2004.

E. Gawlik, T. Munson, J. Sarich, and S.M. Wild. The TAO linearly constrained augmented Lagrangian method for PDE-constrained optimization. Preprint ANL/MCS-P2003-0112, Mathematics and Computer Science Division, January 2012. URL `http://www.mcs.anl.gov/uploads/cels/papers/P2003-0112.pdf`.

P.E. Gill, W. Murray, and M.A. Saunders. SNOPT: An SQP Algorithm for Large-Scale Constrained Optimization. SIAM Journal on Optimization, 12(4):979–1006, 2002. doi: 10.1137/S1052623499350013.

N.I.M. Gould, M.E. Hribar, and J. Nocedal. On the solution of equality constrained quadratic problems arising in optimization. SIAM Journal on Scientific Computing, 23(4):1375–1394, 2001. doi: 10.1137/S1064827598345667.

N.I.M. Gould, D. Orban, and P.L. Toint. CUTEr and SifDec, a Constrained and Unconstrained Testing Environment, revisited. ACM Trans. Math. Softw., 29(4):373–394, 2003. doi: 10.1145/962437.962439.

C. Greif, E. Moulding, and D. Orban. Bounds on the eigenvalues of block matrices arising from interior-point methods. SIAM Journal on Optimization, 24(1):49–83, 2014. doi: 10.1137/120890600.

A. Griewank and A. Walther. On Constrained Optimization by Adjoint based Quasi-Newton Methods. Optimization Methods and Software, 17:869–889, 2002. doi: 10.1080/1055678021000060829.

R.T. Haftka and M.P. Kamat. Simultaneous Nonlinear Structural Analysis and Design. Computational Mechanics, 4:409–416, 1989. doi: 10.1007/BF00293046.

J.T. Hwang, D.Y. Lee, J.W. Cutler, and J.R.R.A. Martins. Large-scale multidisciplinary optimization of a small satellite's design and operation. Journal of Spacecraft and Rockets, 2014. doi: 10.2514/1.A32751. (In press).

G.J. Kennedy and J.R.R.A. Martins. Parallel Solution Methods for Aerostructural Analysis and Design Optimization. In 13th AIAA/ISSMO Multidisciplinary Analysis and Optimization Conference, Fort Worth, TX, Sept. 2010.

G.J. Kennedy and J.R.R.A. Martins. A laminate parametrization technique for discrete ply angle problems with manufacturing constraints. Structural and Multidisciplinary Optimization, 48(2):379–393, 2013. doi: 10.1007/s00158-013-0906-9.

G.J. Kennedy and J.R.R.A. Martins. A parallel finite-element framework for large-scale gradient-based design optimization of high-performance structures. Finite Elements in Analysis and Design, 87:56–73, 2014. doi: 10.1016/j.finel.2014.04.011. URL `http://linkinghub.elsevier.com/retrieve/pii/S0168874X14000730`.

G.K.W. Kenway and J.R.R.A. Martins. Multi-point high-fidelity aerostructural optimization of a transport aircraft configuration. Journal of Aircraft, 2014. doi: 10.2514/1.C032150. (In press).

G.K.W. Kenway, G.J. Kennedy, and J.R.R.A. Martins. Scalable parallel approach for high-fidelity steady-state aeroelastic analysis and derivative computations. AIAA Journal, 2014. doi: 10.2514/1.J052255. (In press).

C.-J. Lin and J.J. Moré. Newton's method for large bound-constrained optimization problems. SIAM Journal on Optimization, 9:1100–1127, 1998. doi: 10.1137/S1052623498345075.

Z. Lyu and J.R.R.A. Martins. Aerodynamic design optimization studies of a blended-wing-body aircraft. Journal of Aircraft, 2014. doi: 10.2514/1.C032491. (In press).

H.J. Martínez. Local and superlinear convergence of structured secant methods for the convex class. Technical report, Rice University, Houston, USA, 1988.

J.J. Moré and G. Toraldo. Algorithms for bound constrained quadratic programming problems. Numerische Mathematik, 55:377–400, 1989. doi: 10.1007/BF01396045.

J.J. Moré and G. Toraldo. On the solution of large quadratic programming problems with bound constraints. SIAM Journal on Optimization, 1(1):93–113, 1991. doi: 10.1137/0801008.

T. Munson, J. Sarich, S. Wild, S. Benson, and L.C. McInnes. TAO 2.0 Users Manual. Technical Report ANL/MCS-TM-322, Mathematics and Computer Science Division, Argonne National Laboratory, 2012. `http://www.mcs.anl.gov/tao`.

B.A. Murtagh and M.A. Saunders. Large-scale linearly constrained optimization. Mathematical Programming, 14:41–72, 1978.

B.A. Murtagh and M.A. Saunders. MINOS 5.51 user's guide. Technical Report SOL 83-20R, Stanford University, Stanford, CA, USA, 2003.

J. Nocedal and S.J. Wright. Numerical Optimization. Springer, New York, 2nd edition, 2006.

J. Nocedal and Y. Yuan. Combining trust region and line search techniques. In Advances in Nonlinear Programming, Proceedings of the 96 International Conference on Nonlinear Programming, 153–175, Springer-Verlag, 1998.

D. Orban. Limited-memory ldl^t factorization of symmetric quasi-definite matrices. Les Cahiers du GERAD, G–2013–87, HEC Montréal, Canada, 2013.

D. Orban. NLPy — a large-scale optimization toolkit in Python. Les Cahiers du GERAD G–2014–xx, HEC Montréal, Canada, 2014. In preparation.

R.E. Perez, P.W. Jansen, and J.R.R.A. Martins. pyOpt: a Python-based object-oriented framework for nonlinear constrained optimization. Structural and Multidisciplinary Optimization, 45(1):101–118, January 2012. doi: 10.1007/s00158-011-0666-3.

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

R.T. Rockafellar. The multiplier method of hestenes and powell applied to convex programming. Journal of Optimization Theory and Applications, 12(6):555–562, 1973. doi: 10.1007/BF00934777.

S. Schlenkrich, A. Griewank, and A. Walther. On the local convergence of adjoint Broyden methods. Mathematical Programming, 121:221–247, 2010. doi: 10.1007/s10107-008-0232-y.

L.A. Schmit. Structural Design by Systematic Synthesis. In 2nd Conference on Electronic Computation, pages 105–132, New York, NY, 1960. ASCE.

P.L. Toint. Non-monotone trust-region algorithms for nonlinear optimization subject to convex constraints. Mathematical Programming, 77(3):69–94, 1997. doi: 10.1007/BF02614518.

M.J. Turner, R.W. Clough, H.C. Martin, and L.J. Topp. Stiffness and deflection analysis of complex structures. Journal of Aeronautical Science, 23(9):805–823, 1956.

A. Wächter and L.T. Biegler. On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming. Mathematical Programming, 106:25–57, 2006. doi: 10.1007/s10107-004-0559-y.