10th World Congress on Structural and Multidisciplinary Optimization
May 19 - 24, 2013, Orlando, Florida, USA

A Matrix-Free Approach to Large-Scale Structural Optimization

Andrew B. Lambe[1] and Joaquim R. R. A. Martins[2]

[1]University of Toronto, Toronto, Canada, lambe@utias.utoronto.ca
[2]University of Michigan, Ann Arbor, Michigan, USA, jrram@umich.edu

1. Abstract

In many problems within structural and multidisciplinary optimization, the computational cost is dominated by computing gradient information for the objective and all constraints. If the problem contains both a large number of design variables and a large number of constraints, analytic gradient computation methods become inefficient. Constraint aggregation may be used together with the adjoint method to reduce the cost of the gradient computation at the expense of problem conditioning and the quality of the final solution. An alternative approach is proposed in which a specialized optimizer is employed that only requires products of the constraint Jacobian with appropriate vectors rather than the full Jacobian itself. These matrix-vector products can be formed for a fraction of the cost of forming the full matrix, allowing the original set of constraints to be used without aggregation. We regard the resulting optimizer as "matrix-free" in that it does not require the Hessian or Jacobian matrices of the optimization problem to be explicitly formed. Results on two simple structural optimization problems are presented.

2. Keywords

structural optimization, large-scale optimization, matrix-free optimization, augmented Lagrangian methods, quasi-Newton methods

3. Introduction

We state the constrained optimization problem of interest as follows:

$$\begin{aligned} \text{minimize} \quad & f(x, y(x)) \\ \text{with respect to} \quad & x \\ \text{subject to} \quad & c(x, y(x)) \leq 0 \\ & x_L \leq x \leq x_U \\ \text{where} \quad & y \text{ solves } \mathcal{R}(x, y) = 0 \end{aligned} \tag{1}$$

In this problem statement, x is the set of design variables with bounds x_L and x_U, y is the set of state variables, and $\mathcal{R}$ is the set of governing equations of the structural analysis. Note that Problem (1) can be extended to describe multidisciplinary systems by defining multiple groups of design variables, state variables, design constraints, and analysis equations. (Martins and Lambe [18] give a detailed discussion of the problem formulations in multidisciplinary design optimization.) In the literature, Problem (1) is often referred to as the Nested Analysis and Design (NAND) problem [1]. In this form, the state variables y are treated as functions of the design variables x by solving $\mathcal{R}(x, y) = 0$ every time x is updated. Simultaneous Analysis and Design [13] problems, in which $\mathcal{R}(x, y) = 0$ are treated as optimization problem constraints and y become independent variables, will not be considered here.

When gradient-based methods are used to solve Problem (1), most of the computational effort is spent evaluating the objective and constraint functions and their derivatives. The derivatives constitute the majority of the cost if the problem has a large number of constraints. Therefore, optimization algorithms that require few function and gradient evaluations to solve the problem are desirable. In the wider optimization community, sequential quadratic programming (SQP) methods of various types are the tool of choice. In structural optimization, convex approximation methods that exploit specific nonlinear problem structures are also very popular. These methods are implemented in software packages like CONLIN [8], MMA [22], and SAOi [12]. Each of these optimization codes use intervening variables to form high-quality nonlinear approximations to the objective and constraints of the original design problem. (See also [14] for a general introduction to structural optimization.)

While these optimization methods can all be used to solve Problem (1), their performance will necessarily suffer if Problem (1) contains both a large number of design variables and a large number of constraints. In SQP, for example, the full constraint Jacobian dc/dx must be computed and returned to the optimizer to solve the quadratic subproblem. In the convex approximation methods, even if a dual method can be employed to solve the convex subproblem, the full Jacobian is needed to compute coefficients that define the approximation functions. In a problem with many variables and many constraints, where the cost is dominated by the gradient computation, this requirement for the full Jacobian becomes the computational bottleneck for the whole problem.

Problems with both many design variables and many constraints are common in structural optimization. In particular, minimum-mass structural design problems subject to failure constraints will naturally require a large number of constraints — the stress on each finite element in the structural model will need to be constrained to prevent failure everywhere in the structure. A common remedy for the large number of constraints is to aggregate them to reduce the size of the Jacobian and therefore reduce the computational cost. However, constraint aggregation can cause the optimization problems to become nonsmooth or poorly conditioned. Even if the problem can be solved by gradient-based optimization, the extra effort required offsets the reduced cost of computing the Jacobian. Furthermore, if the aggregation is conservative, the final designs obtained when using constraint aggregation can have a higher mass than those obtained without aggregation.

In this work, we present an alternative approach to solving structural optimization problems with both many variables and many constraints that avoids the problems associated with constraint aggregation. The key idea is to use an optimizer that requires only matrix-vector products between the constraint Jacobian and some vectors of interest. Because the optimizer does not require problem Hessian or Jacobian information directly, we refer to such an optimizer as "matrix-free." [7, 10] Section 4. reviews how to compute the Jacobian analytically and how matrix-vector products can be computed efficiently. Section 5. briefly discusses our matrix-free optimization algorithm. Particular emphasis is placed on how we use quasi-Newton methods to keep the number of matrix-vector products low. Section 6. discusses the optimization results on two structural test problems and compares the cost with SQP. Section 7. offers concluding remarks and speculates about the future prospects of this optimization approach.

4. Gradient Computation Methods

The most efficient way to compute the gradients for Problem (1) is analytically. To do so requires knowledge of specific sets of partial derivatives of the governing equations with respect to design and state variables. While it is not a trivial task to obtain these derivatives, we will assume that they are available for the purpose of our discussion.

Under Problem (1), we can compute the derivative of constraint c_i with respect to variable x_j as

$$\begin{aligned} \frac{dc_i}{dx_j} &= \frac{\partial c_i}{\partial x_j} + \sum_{k=1}^{M} \frac{\partial c_i}{\partial y_k} \frac{dy_k}{dx_j} \\ &= \frac{\partial c_i}{\partial x_j} + \frac{\partial c_i}{\partial y} \frac{dy}{dx_j}, \end{aligned} \tag{2}$$

where M is the total number of state variables. We also use 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 (2) 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 $\mathcal{R}(x, y) = 0$ has been solved for x_j and no change in x_j alters this fact. Therefore, we can state that

$$\frac{d\mathcal{R}}{dx_j} = 0 = \frac{\partial \mathcal{R}}{\partial x_j} + \frac{\partial \mathcal{R}}{\partial y} \frac{dy}{dx_j}. \tag{3}$$

Rearranging the terms in Equation (3) yields

$$\frac{dy}{dx_j} = - \left[\frac{\partial \mathcal{R}}{\partial y} \right]^{-1} \frac{\partial \mathcal{R}}{\partial x_j}. \tag{4}$$

By direct substitution of Equation (4) into Equation (2), we obtain

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

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

$$\frac{dc}{dx} = \frac{\partial c}{\partial x} - \left[\frac{\partial c}{\partial y}\right]\left[\frac{\partial \mathcal{R}}{\partial y}\right]^{-1}\frac{\partial \mathcal{R}}{\partial x} \tag{6}$$

Note that Equation (6) describes an $m \times n$ matrix, where m is the number of constraints and n is the number of design variables.

The direct and adjoint methods [17] for computing gradient information follow immediately from Equation (6). In the direct method, a sequence of linear systems of the form

$$\left[\frac{\partial \mathcal{R}}{\partial y}\right]\frac{dy}{dx_j} = -\frac{\partial \mathcal{R}}{\partial x_j} \tag{7}$$

is solved to form a dy/dx matrix that is then used to compute dc/dx. In the adjoint method, an alternative sequence of linear systems of the form

$$\left[\frac{\partial \mathcal{R}}{\partial y}\right]^T \psi_{c_i} = -\left[\frac{\partial c_i}{\partial y}\right]^T \tag{8}$$

is solved to compute a matrix, ψ_c^T, that is then multiplied by $\partial \mathcal{R}/\partial x$ to compute dc/dx. Note that the number of linear systems to be solved in the direct method is n, while the number of linear systems to be solved in the adjoint method is m. Another way to think about these methods is that the direct method assembles dc/dx one column at a time while the adjoint method assembles it one row at a time. Because the number of state variables is often much larger than the number of design variables and constraints, the most expensive part of forming the Jacobian is solving the linear systems (7) or (8). Therefore, either the direct or adjoint method may be selected based on which method requires the fewest linear systems to be solved. For example, if $n > 100$ but $m < 10$ for the problem of interest, we would expect the adjoint method to be an order of magnitude more efficient.

We now turn to the subject discussed in the introduction: what if the optimization problem were to contain many (potentially thousands) of design variables *and* many constraints? While we may choose to select either the direct or adjoint method, the similar magnitudes of m and n suggests that the relative gain in efficiency will be small. Even if m and n are widely separated in magnitude but $\min(m, n)$ is large, the cost of computing a single Jacobian can be very high. Constraint aggregation approaches reduce this cost by lumping groups of constraints together to reduce the size of m. However, a major side effect of this approach is that it creates optimization problems that are nonsmooth or poorly conditioned. For example, constraining the maximum constraint violation leads to a nonsmooth problem because the gradient of the maximum function is not defined everywhere in the domain. The Kreisselmeier–Steinhauser (KS) aggregation technique [16]

$$KS[c(x)] = c_{max} + \frac{1}{\rho}\ln\left[\sum_{i=1}^{m}\exp\left(\rho(c_i(x, y(x)) - c_{max})\right)\right] \tag{9}$$

enjoys the advantage of smoothness, but can cause the resulting optimization problem to become ill-conditioned if the controlling parameter ρ is chosen to be too large. The ill-conditioning comes from the sharp curvature that results when the KS function tries to model the intersection of two or more constraints. [20] Effectively, the KS function behaves more like the maximum function as ρ increases. Furthermore, the conservative property of the KS function means that the optimal solution of the KS-constrained problem will not be the same as that of the original problem. While the error in the final constraint feasibility is bounded, [20] it is especially pronounced in regions where multiple constraints are active. Therefore, using a constraint aggregation approach will necessarily lead to a compromise between the computational effort and the quality of the final solution.

We propose to avoid the problems associated with aggregation by avoiding computation of the full Jacobian at each iteration. Instead, we rely on products of the Jacobian with appropriate vectors to

extract the gradient information necessary to solve the problem. Consider Equation (6) again, but multiply the Jacobian by an n-length vector v.

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

Similarly, consider multiplying the transpose of the Jacobian by an m-length vector w.

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

Note that to compute the final matrix-vector product, we never need to form dc/dx. We simply perform a set of matrix-vector multiplications from right to left in Equations (10) and (11). Furthermore, we do not even need explicit representations of the partial derivative matrices in order to compute the corresponding matrix-vector products. Most importantly, the cost of computing a single matrix-vector product becomes virtually independent of m and n. Because of the right-to-left multiplication scheme, the number of large linear systems to solve to compute a single product is *exactly one.* Thus, unlike the direct or adjoint methods, the optimization problem size no longer dictates the number of large linear systems to solve.
This approach does present other issues. First, few algorithms and, to our knowledge, no optimization software packages solve general nonlinear problems and accept only matrix-vector product information. We have therefore developed our own software to test out this idea. Second, by relying on matrix-vector products, the resulting algorithm is necessarily working with less information about the underlying problem. Therefore, we expect to trade-off the number of iterations of the optimization algorithm against the amount of information that the algorithm requires at each iteration. This is a favourable trade-off to make in structural optimization given the large imbalance in the computational work between the analysis and the optimization. Finally, to ensure that the trade-off works in our favour, we need to focus on keeping the total number of matrix-vector products per iteration small.

5. A Matrix-Free Optimization Algorithm

For our optimization algorithm, we have chosen to implement a matrix-free version of an augmented Lagrangian penalty method. Penalty methods are easy to make matrix-free due to the fact that the gradient of the penalty function includes a matrix-vector product involving the Jacobian. The augmented Lagrangian was chosen in particular because it is a differentiable penalty function and the optimal solution can be computed with a finite penalty weight [2]. Given a nonlinear optimization problem with variable bounds

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

the problem is reformulated as

$$\begin{aligned} \text{minimize} \quad & \phi(x, r; \lambda, \rho) = f(x) - \lambda^T (c(x) + r) + \frac{\rho}{2} (c(x) + r)^T (c(x) + r) \\ \text{with respect to} \quad & x, r \\ \text{subject to} \quad & x_L \leq x \leq x_U \\ & r \geq 0, \end{aligned} \tag{13}$$

where λ is a vector of Lagrange multipliers, r is a set of slack variables, and ρ is a penalty parameter. Each iteration of the augmented Lagrangian method (also known as the method of multipliers) consists of solving problem (13) and updating ρ and λ based on the infeasibility of the final solution x and r. We use the updating scheme proposed by Conn, Gould, and Toint [6], but choosing λ based on the solution to the least-squares problem

$$\text{minimize} \quad ||\nabla f - \tilde{J}^T \lambda||_2, \tag{14}$$

where $\tilde{J}$ is the Jacobian of equality constraints with respect to both r and x. This scheme allows problem (13) to be solved inexactly, yet still converge rapidly near a local minimum. Problem (13) is solved by a nonmonotone, l_∞ trust-region method in which the quadratic subproblems were solved by the

algorithm of Moré and Toraldo [19]. Complete details of our algorithm will be presented in a forthcoming paper. For the rest of this section, we focus on how the optimizer keeps the required number of matrix-vector products low while maintaining fast convergence.

Because we do not have analytic second derivative information available, we approximate the Hessian of the augmented Lagrangian using a limited-memory quasi-Newton method. However, we have found through experimentation that approximating $\nabla^2\phi$ directly generally results in poor performance, even on test problems of ten variables or less. Much better results can be obtained by exploiting the structure of $\nabla_x^2\phi$ while still maintaining the matrix-free nature of the algorithm. Observe that

$$\nabla_x^2\phi = \nabla^2 f - \sum_{i=1}^{m} \lambda_i \nabla^2 c_i + \sum_{i=1}^{m} \rho(c_i(x) + r_i)\nabla^2 c_i + \rho J^T J, \tag{15}$$

where J is used to denote the Jacobian of c with respect to x only. The first two terms are just the Hessian of the Lagrangian, while the last term is a product of Jacobians. Because the algorithm we use to solve the quadratic trust-region subproblems is based on the conjugate gradient (CG) method, we require Hessian-vector products rather than the Hessian itself. Therefore, we can account for the $J^T J$ term directly using Jacobian-vector products. To approximate the second-order terms in (15), we use a limited-memory symmetric rank-one [5] approximation to the Hessian of the Lagrangian and simply truncate the additional constraint Hessian term. (It is difficult to approximate this term well with only Jacobian-vector product information. We have observed no ill effects from truncating this term in our test problems.) The Hessian terms that incorporate the slack variables can be similarly decomposed and approximated. Our tests have shown this approximation scheme yields a large reduction in the number of iterations to solve (13) compared to a naïve application of a quasi-Newton approximation to $\nabla^2\phi$.

One issue remains with the above approximation scheme: while the number of trust-region iterations to solve (13) was greatly reduced, the total number of Jacobian-vector products was still very high. Because we use Jacobian-vector product information to help compute approximate Hessian-vector products in the CG iteration, each CG iteration requires two Jacobian-vector products. Many CG iterations are needed within each trust-region iteration so the total number of Jacobian-vector products remains high. Preconditioning the CG method to reduce the number of iterations is a possibility but we do not know any preconditioner that would be suitable for the Moré and Toraldo algorithm. Instead, we work around the problem by avoiding matrix-vector products with the true Jacobian within the CG algorithm. We do so by forming a quasi-Newton approximation to the Jacobian and using this approximation to solve the trust-region subproblem. While we pay a small penalty by increasing the number of trust-region iterations, (due to the inaccuracy of our Jacobian estimate,) we decouple the number of Jacobian-vector products from the number of CG iterations. This decoupling yields the vast reduction in matrix-vector products that we desire.

Quasi-Newton approximations to nonsquare matrices have been known in the literature for many years [4, 3] but have not frequently been applied to optimization problems. Our algorithm tests two quasi-Newton approximations proposed by Schlenkrich et al. [21] The first method is referred to as the adjoint Broyden method. For a given approximate Jacobian A_k the adjoint Broyden update is given by

$$A_{k+1} = A_k + \frac{\sigma_k \sigma_k^T}{\sigma_k^T \sigma_k} (J_{k+1} - A_k) \tag{16}$$

where J_{k+1} is the current Jacobian matrix and σ_k is an adjoint search direction. Note that this update requires only a single matrix-vector product: $J_{k+1}^T \sigma_k$. While several choices of σ_k are possible, we use

$$\sigma_k = c(x_{k+1}) + r_{k+1} - c(x_k) - r_k - A_k(x_{k+1} - x_k) \tag{17}$$

in this study. This choice is the same as option (B) in Schlenkrich et al. [21] The second method is the two-sided rank-one (TR1) method. While this method was first proposed by Griewank and Walther [11], our tests use the version discussed by Schlenkrich et al. [21] The TR1 update formula is given by

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

where $s_k = x_{k+1} - x_k$. We choose (17) to be the adjoint search direction in the TR1 update. Unlike the adjoint Broyden update (16), the TR1 update requires two matrix-vector products: $J_{k+1}^T \sigma_k$ and $J_{k+1} s_k$. Nevertheless, we have found both updates to be effective in reducing the number of Jacobian-vector products required by the augmented Lagrangian algorithm.

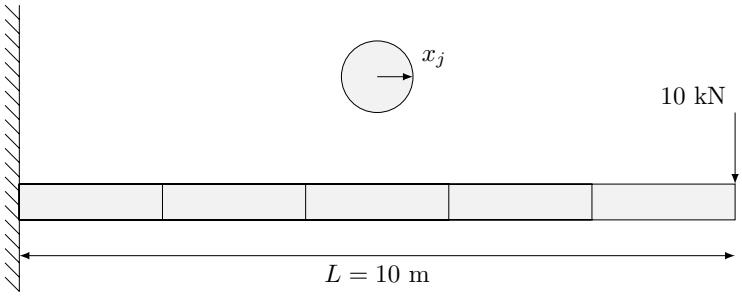

Figure 1: Layout of cantilever beam design problem

6. Structural Test Results

We now present results of our matrix-free augmented Lagrangian algorithm on two structural test problems. We also compare these results with those from the SQP optimizer SNOPT [9]. The basis for our comparison will be the total number of times we solve a linear system involving the matrix $\partial\mathcal{R}/\partial y$. This metric allows us to directly compare matrix-free optimizers with more traditional optimizers that require computation of the full Jacobian. Furthermore, a common technique to solve the system $\mathcal{R}(x, y) = 0$ is Newton's method, which relies on the linear system

$$\left[\frac{\partial\mathcal{R}}{\partial y}\right]\Delta y = -\mathcal{R}(\bar{x}, \bar{y}) \tag{19}$$

where $\bar{x}$ and $\bar{y}$ are particular choices of the design and state variables. Both of our test problems use the linear finite element method, which is equivalent to a single solution of linear system (19). (In the structural context, y is the set of displacements and $\partial\mathcal{R}/\partial y$ is just the stiffness matrix.) Therefore, by totalling the number of linear system solves, we create a cost metric that accounts for both the number of function evaluations and the number of gradient evaluations at the same time. As the structural model and optimization problem sizes increase, solving these linear systems becomes the dominant computational cost.

The first problem is to minimize the mass of a cantilever beam. The beam has a solid, circular cross-section and is loaded only at the free end. Figure 1 gives the initial geometry of the beam. The design variables for the problem are the radii of each beam element. Initially, all design variables are set to 3 mm. The problem constraints are that the maximum stress in each element must be less than the yield stress of the material. The material properties for the beam are as follows: $E = 70$ GPa, $G = 26$ GPa, $\sigma_{yield} = 324$ MPa, $\rho = 2780$ kg/m^3. In each version of the problem that we test, there is one design variable and one constraint per beam element.

Figure 2 plots the computational cost of solving each optimization problem against the problem size. The cantilever beam problem is solved for up to 1000 elements, corresponding to a problem size of 1000 variables and 1000 constraints. In all cases, SNOPT and the augmented Lagrangian method converge to the same solution. Again, the cost comparison is in terms of the number of expensive linear systems which must be solved. Figure 2 shows that SNOPT is the superior optimizer for small problems but becomes very costly to use on larger problems due to the need to compute the whole Jacobian for each iteration. In contrast, the growth rate in the cost is much lower using any version of the augmented Lagrangian solver. Furthermore, maintaining quasi-Newton approximations to the Jacobian within the trust-region solver yields a very large reduction in the computational cost — generally 60-80% for both types of approximation — when compared to using true Jacobian-vector products everywhere in the algorithm. It is interesting to note that SNOPT never requires more than 20 iterations (20 Jacobian computations) to solve the problem while the augmented Lagrangian method requires up to 200 iterations on the larger versions of the problem. Therefore, most of the cost reduction is attributed to avoiding the frequent recomputation of the full Jacobian.

The second structural test problem is to minimize the mass of a square plate clamped on all sides and subject to a pressure load. Figure 3 shows the initial geometry of the plate and the load. The material properties for the plate are as follows: $E = 70\ 000$, $\nu = 0.3$, $\sigma_{yield} = 300.0$, $\rho = 1.0$, $k_{corr} = 5/6$. The plate thickness variables are bounded between 3 mm and 7 mm. By symmetry, we only need to analyze one quarter of the plate to obtain results for the whole. As with the beam problem, every finite element in

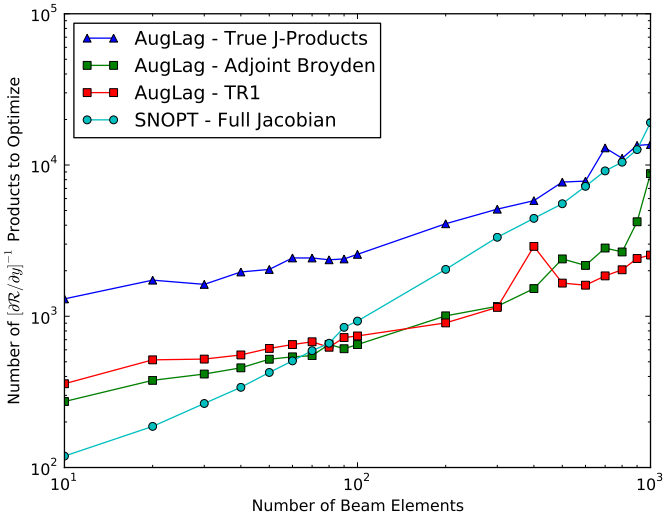

Figure 2: Computational cost of solving cantilever beam design problem

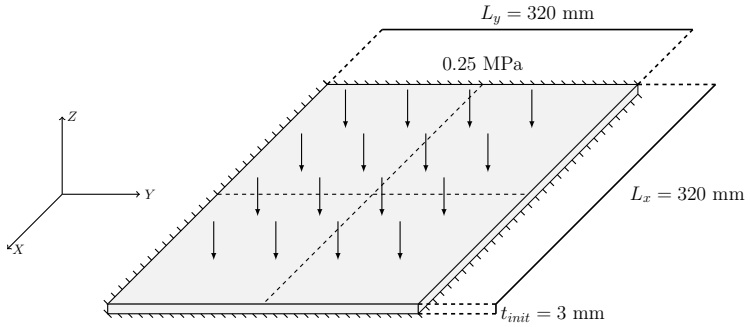

Figure 3: Layout of pressurized plate design problem

the discretization is associated with a single design variable, thickness, and a single constraint, maximum stress less than a yield stress. Five discretization schemes are tested: 12×12, 18×18, 24×24, 30×30, and 36×36 plate elements. The finite element analysis was carried out using the TACS code (Toolkit for the Analysis of Composite Structures [15]) using third-order MITC shell elements. In contrast to the cantilever beam problem, the plate problem is statically indeterminate. We therefore expect this test to be a more representative of the matrix-free algorithm's performance on general structural optimization problems.

Figure 4 plots the computational cost of solving different versions of the plate problem using SNOPT and the augmented Lagrangian algorithm with quasi-Newton Jacobian approximations. The final plate designs generated by each optimizer are not identical but the difference in optimum mass is less than 0.3% on the coarsest mesh. (This difference decreases as the mesh is refined.) Just like with the cantilevered beam problem, Figure 4 shows that the cost of the augmented Lagrangian algorithm has more favourable scaling than that of SNOPT. Results for the augmented Lagrangian algorithm with true Jacobian-vector products are not plotted because that algorithm is far more expensive than the versions using the approximate Jacobians. As with the beam problem, the combination of approximate Jacobians and a matrix-free optimization algorithm is effective at reducing computational cost, even compared to a robust SQP algorithm like SNOPT. Once again, SNOPT requires recalculation of the Jacobian less than 20 times to solve the problem. However, the cost of computing the whole Jacobian is so large that maintaining an approximation to the Jacobian, as we do in our matrix-free augmented Lagrangian method, is much more cost effective. For the largest case plotted in Figure 4, (1296 variables and constraints,) using the matrix-free

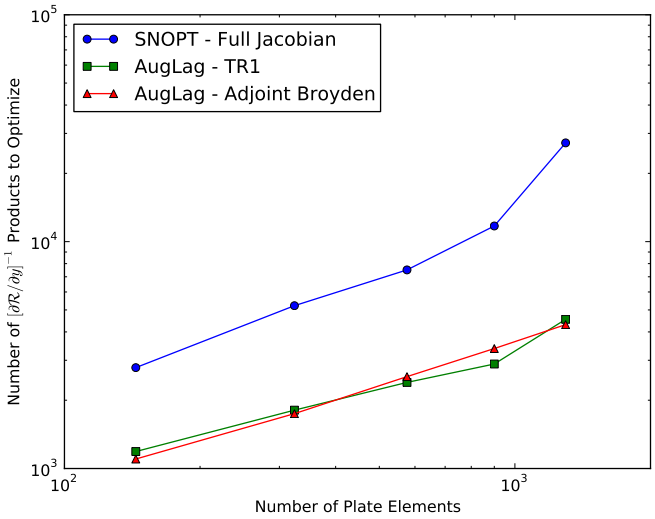

Figure 4: Computational cost of solving pressurized plate design problem

augmented Lagrangian algorithm reduces the number of large linear systems to solve by nearly 85%. We expect that an order-of-magnitude improvement over traditional SQP should be possible for larger test cases.

7. Conclusions

We have presented an alternative approach to structural optimization using a matrix-free algorithm. The matrix-free approach was motivated by taking another look at how gradient information was computed for an optimization problem with both many variables and many constraints. Because computing products of the Jacobian with a few vectors of interest is far cheaper than trying to compute the full Jacobian for large problems, we use this information to approximate and update the Jacobian through a quasi-Newton method. Our test results show great promise. Even though our augmented Lagrangian algorithm required many more iterations than an SQP algorithm to solve the test problems, the lower cost of the augmented Lagrangian iterations meant that we could reduce the total computational work by as much as 85% compared to SQP. We emphasize that because we did not change the original set of constraints on the problem at all, the reductions in computational cost do not come with any of the caveats associated with constraint aggregation.

Future work on this project consists of testing the algorithm on more complicated structures to verify the trend of reduced computational load as the model and problem sizes increase. We also forsee an application of this algorithm to multidisciplinary problems such as the combined aerodynamic and structural optimization of aircraft wings. In the multidisciplinary case, the imbalance in computational cost between the analysis and the optimization is even more pronounced than for the single-discipline case. Furthermore, multidisciplinary problems are more likely to rely on iterative methods to solve the linear systems because of the difficulty in computing partial derivative information within the multidisciplinary system. In these cases, the matrix-free optimization approach is especially attractive.

8. Acknowledgements

The authors would like to thank Sylvain Arreckx and Dominique Orban from École Polytechnique de Montréal for their assistance in developing the augmented Lagrangian algorithm within the NLPy computing environment. The first author was partially funded by a scholarship from the Natural Science and Engineering Research Council of Canada.

9. References

[1] R. J. Balling and J. Sobieszczanski-Sobieski. Optimization of Coupled Systems: A Critical Overview of Approaches. *AIAA Journal*, 34(1):6–17, 1996.

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

[3] S. K. Bourji and H. F. Walker. Least-Change Secant Updates of Nonsquare Matrices. *SIAM Journal on Numerical Analysis*, 27(5):1263–1294, 1990.

[4] C. G. Broyden. A Class of Methods for Solving Nonlinear Simultaneous Equations. *Mathematics of Computation*, 19(92):577–593, 1965.

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

[6] A. R. Conn, N. I. M. Gould, and P. L. Toint. A Globally Convergent Augmented Lagrangian Algorithm for Optimization with General Constraints and Simple Bounds. *SIAM Journal on Numerical Analysis*, 28(2):545–572, 1991.

[7] F. E. Curtis, J. Nocedal, and A. Wächter. A Matrix-Free Algorithm for Equality Constrained Optimization Problems with Rank-Deficient Jacobians. *SIAM Journal on Optimization*, 20(3):1224–1249, 2009.

[8] C. Fleury and V. Braibant. Structural Optimization: a New Dual Method Using Mixed Variables. *International Journal for Numerical Methods in Engineering*, 23:409–428, 1986.

[9] 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.

[10] J. Gondzio. Matrix-free interior point method. *Computational Optimization and Applications*, 51:457–480, 2012.

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

[12] A. A. Groenwold, L. F. P. Etman, and D. W. Wood. Approximated approximations for SAO. *Structural and Multidisciplinary Optimization*, 41:39–56, 2010.

[13] R. T. Haftka. Simultaneous Analysis and Design. *AIAA Journal*, 23(7):1099–1103, July 1985.

[14] R. T. Haftka, Z. Gürdal, and M. P. Kamat. *Elements of Structural Optimization*. Kluwer Academic Publishers, Dordrecht, NL, 1992.

[15] 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.

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

[17] J. R. R. A. Martins and J. T. Hwang. Review and Unification of Discrete Methods for Computing Derivatives of Single- and Multi-disciplinary Computational Models. *AIAA Journal*, 2013. (In press).

[18] J. R. R. A. Martins and A. B. Lambe. Multidisciplinary Design Optimization: Survey of Architectures. *AIAA Journal*, 2013. (In press).

[19] 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.

[20] 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.

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

[22] K. Svanberg. The Method of Moving Asymptotes - a New Method for Structural Optimization. *International Journal for Numerical Methods in Engineering*, 24:359–373, 1987.