INTERNATIONAL JOURNAL FOR NUMERICAL METHODS IN ENGINEERING
Int. J. Numer. Meth. Engng 2016; **00**:1–26
Published online in Wiley InterScience (www.interscience.wiley.com). DOI: 10.1002/nme

Level-set topology optimization with many linear buckling constraints using an efficient and robust eigensolver

Peter D. Dunning[1*], Evgueni Ovtchinnikov[2], Jennifer Scott[2], H. Alicia Kim[3]

[1]*School of Engineering, University of Aberdeen, Aberdeen AB24 3UE, UK*
[2]*Scientific Computing Department, STFC Rutherford Appleton Laboratory, Harwell Oxford, Didcot OX11 0QX, Oxfordshire, UK*
[3]*Structural Engineering Department, University of California, San Diego, USA*

SUMMARY

Linear buckling constraints are important in structural topology optimization for obtaining designs that can support the required loads without failure. During the optimization process the critical buckling eigenmode can change; this poses a challenge to gradient based optimization and can require the computation of a large number of linear buckling eigenmodes. This is potentially both computationally difficult to achieve as well as prohibitively expensive. In this paper, we motivate the need for a large number of linear buckling modes and show how several features of the Block Jacobi Conjugate Gradient (BJCG) eigenvalue method, including optimal shift estimates, the reuse of eigenvectors, adaptive eigenvector tolerances and multiple shifts, can be used to efficiently and robustly compute a large number of buckling eigenmodes.
This paper also introduces linear buckling constraints for level-set topology optimization. In our approach the velocity function is defined as a weighted sum of the shape sensitivities for the objective and constraint functions. The weights are found by solving an optimization sub-problem to reduce the mass, whilst maintaining feasibility of the buckling constraints. The effectiveness of this approach in combination with the BJCG method is demonstrated using a 3D optimization problem. Copyright © 2016 John Wiley & Sons, Ltd.

Received . . .

KEY WORDS: Topology optimization; buckling constraints; level-set method; block conjugate gradient eigensolver; sparse direct linear solver

1. INTRODUCTION

The objective of optimum structural design is often to find the lightest structure that can support the loading conditions without failure. There has been extensive research into developing and applying optimization methods to structural design problems [1, 2]. However, structures that are optimized for minimum weight often consist of beam-like members or thin panels. If these types of structure are subject to compressive loads, then they may be prone to buckling, which ultimately limits the capacity of the structure to carry load. Therefore, it is important to include buckling as a constraint in structural optimization problems.

Topology optimization is one approach to design efficient structures that has received significant attention [3, 4]. The idea is to open up the design space such that sizing, shape and connectivity of the structure are considered simultaneously, leading to the greatest design freedom.

Buckling has been considered in both truss and continuum type topology optimization. In truss type topology optimization, the design space typically consists of a fixed set of nodes connected

*Correspondence to: School of Engineering, University of Aberdeen, Aberdeen AB24 3UE, UK. E-mail: peter.dunning@abdn.ac.uk

by a large number of pin-jointed truss members, a so-called ground structure. The design variables are usually member cross-sections, that can go to zero to allow a change in topology. Local Euler buckling constraints can be specified on each member, although a modification of the design space is often required to handle discontinuities when member areas go to zero [5]. However, the local constraints may not prevent global buckling of the structure, because of the presence of hinges [6]. Thus, a global buckling constraint is also included [7]. For continuum structures, it is difficult to identify discrete structural members. Therefore, specific local buckling constraints are not used [8]. Instead, a linear buckling analysis [8, 9, 10, 11] or a geometrically nonlinear analysis [8, 12] is used to compute the critical buckling load. Buckling constraints have been used with several continuum topology optimization methods including Simple Isotropic Material with Penalization (SIMP) [12], Evolutionary Structural Optimization (ESO) [10] and a nodal design variable method [8]. The level-set method is another approach to continuum topology optimization that has received significant attention in that last 10 years [3, 4, 13, 14]. However, to the authors knowledge, buckling constraints have not been included in level-set based topology optimization.

This paper is concerned with the design of continuum structures, where the critical load factor is computed using a linear buckling analysis. The analysis is performed in two steps [15]. The first solves a linear elastic problem using standard Finite Element Analysis (FEA) to obtain displacement and stress values for the given loading. A generalized eigenvalue analysis is then performed, where the eigenvalues are the buckling load factors and the eigenvectors correspond to the buckling mode shapes. This is sometimes known as the buckling eigenvalue problem.

A number of key issues have been identified that can affect the convergence of optimization problems involving linear buckling. One issue relates to the computation of buckling load factor derivatives. The geometric or stress stiffness matrix used in the buckling eigenvalue problem is dependent on the displacements from the linear elastic analysis, which are in turn dependent on the design variables. However, this contribution to the gradient calculation is sometimes ignored for simplicity, resulting in some error in the computed gradients [16]. Another issue is that repeated eigenvalues are only directionally differentiable [17]. Ascent directions in the presence of repeated eigenvalues can be computed by introducing additional requirements that the directional derivative of all repeated eigenvalues must be positive [17, 18]. Non-differentiability can be avoided if the problem is solved using semi-definite programming methods [7], although the addition of matrix constraints significantly increases the computational cost. Bruyneel et al [16] also discuss how the choice of optimizer can affect the convergence of buckling constrained problems and show that the use of information from two consecutive iterations to approximate the curvature produces an envelope of the responses and helps find a feasible solution. Moreover, methods that use an internal move-limit strategy can improve convergence.

Another key issue is the switching of the critical buckling mode during optimization. A constraint is usually posed that the lowest buckling load factor must be greater than one. However, the buckling mode with the lowest load factor may change during optimization. This introduces a discontinuity, as the mode shape and gradient information of the lowest mode will change, leading to slow convergence, or even a failure to converge. Using a greater number of modes provides the optimizer with more comprehensive information about the design space and can improve the convergence. Bruyneel et al [16] showed for one example that increasing the number of buckling mode constraints from 12 to 100 reduced the number of iterations to convergence from 44 to 6. For realistic structures with complex topological configurations, such as 3D aircraft wings [19], even more buckling modes may be required.

Topology optimization methods often use a fictitious weak material to model void regions in the design space. The weak material has stiffness and density values several orders of magnitude than the real material. This leads to large differences in the diagonal entries of the system matrices (such as the stiffness matrix) large condition numbers and ill-conditioning [20]. Furthermore, topology optimization methods that use the extended finite element method to model a material-void interface can produce ill-conditioned matrices if element integration areas are small [21].

The efficient computation of a large number of buckling load factors and mode shapes for large-scale topology optimization problems presents a significant computational challenge, as the

DOI: 10.1002/nme

solution of a sequence of generalized eigenvalue problems with ill-conditioned matrices is required. Currently, the most widely used solvers for such problems, such as ARPACK [22] (which is used by the MATLAB [23] function `eigs`) and the F12 package from the NAG library [24], employ Krylov subspace methods with a shift-and-invert technique. When used to solve a buckling problem, the shift-invert technique requires the user to solve a sequence of large sparse linear systems of the form $(A - \sigma B)u = f$, where A and B are real symmetric ill-conditioned matrices. At present, such systems can only be reliably solved using a sparse direct solver, such as the `HSL_MA97` package [25] from the HSL mathematical software library [26] or the `SPRAL_SSIDS` package from the `SPRAL` library [27]. The solution involves two steps: the factorization of $A - \sigma B$, which must be done once for each shift σ, and the repeated solution of linear systems using the matrix factorization. Although the second step is normally less computationally expensive than the first, it may account for a substantial proportion of the eigenvalue computation time if it is performed a large number of times, for example if a large number of eigenvalues are required.

For a fixed shift σ, Krylov solvers are theoretically optimal, as they, in a sense, require the minimal possible number of shifted solves. For this reason, ARPACK remains a highly competitive eigensolver more than two decades after its introduction. However, there are certain limitations that hinder its efficient use. Firstly, even though a Krylov solver can compute several eigenvalues simultaneously, the shifted systems are solved one after another, limiting the scope for parallel computation. Further, Krylov solvers generally cannot reliably compute repeated eigenvalues, and may miss eigenvalues that are close to each other. This is of particular concern to a linear buckling constrained problem, where the minimum eigenvalue must be greater than one and the optimal design may have several eigenvalues close to one. If a critical eigenvalue close to one is missed then the optimizer does not receive gradient information for the eigenvalue and may change the structure such that the eigenvalue becomes infeasible. This can slow down convergence, as the missed mode oscillates between feasible and infeasible. Also, if a critical eigenvalue is missed throughout the optimization, then the final design may not be feasible if the missed eigenvalue is less than one. Both slow convergence and infeasible final designs are undesirable in optimization and thus a solver that can guarantee that eigenvalues are not missed is preferred. Part of the reason for ARPACK's problems with nearly repeated eigenvalues is the fact that ARPACK counts eigenvalues as they converge. The speed of convergence is determined by the relative separation of the computed eigenvalue from the rest of the spectrum. As a result, well separated eigenvalues further down the spectrum that are not wanted often converge before wanted eigenvalues. In practice, this implies that the user must force the convergence to all wanted eigenvalues by using machine accuracy error tolerance, even if such accuracy is not required, and compute more eigenvalues than are needed.

In this paper, we opt for an alternative approach that exploits two time-tested techniques: subspace, or block, iterations and the conjugate gradient (CG) method. The full details will be given in Section 3: here we briefly summarize the advantages of such an approach. The block iterative nature of the proposed eigensolver ensures that all wanted eigenvalues are computed and allows the efficient exploitation of parallel capabilities of modern computer architectures, both via the use of highly optimized Level 3 BLAS subroutines and the use of OpenMP for simultaneous shifted solves. We need to solve several increasingly close eigenvalue problems and our eigensolver enables us to exploit the previously computed eigenvalues and eigenvectors, significantly improving performance. The reliability of block iterations allows the use of a relaxed error tolerance at early outer iterations and a reduced tolerance as the iterations proceed. Finally, the use of CG delivers convergence similar to that of a block Krylov method (the two are mathematically equivalent when applied to a linear system with a matrix right-hand side [28]) at a considerable reduction in the computational cost. We note that the block CG approach has been successfully used by several authors in combination with preconditioning techniques [29, 30]. However, their experiments have been performed using research software that is not be of library quality in terms of robustness, efficiency and reliability. Our new eigensolver package `SPRAL_SSMFE`, that implements the algorithm described in Section 3, is a fuly supported and maintained high-quality implementation that is included within the `SPRAL` software library [27].

DOI: 10.1002/nme

The novel contributions of this paper are first, the demonstration, using a clear example, that a large number of eigenmodes in linear buckling constrained optimization may be required for stable convergence, second, the efficient and robust computation of such modes using the block Jacobi CG (BJCG) eigenvalue solver `SPRAL_SSMFE` is investigated, and third, a method for including linear buckling constraints in level-set topology optimization is introduced. The paper is organized as follows. Section 2 introduces the buckling constrained minimization of mass problem and the linear buckling eigenvalue problem. The need for computing a large number of buckling modes is highlighted using a simple example. The BJCG eigenvalue solver is introduced and discussed in Section 3 and its performance is investigated in Section 4. The level-set method with buckling constraints is detailed in Section 5. A 3D topology optimization example is presented in Section 6, followed by conclusions in Section 7.

2. LINEAR BUCKLING OPTIMIZATION

2.1. Buckling constrained problem

The optimization problem studied in this paper is to minimize the mass of a structure, subject to a constraint that the lowest positive buckling load factor is greater than or equal to one:

$$\begin{aligned} \min_{x} \quad & M(x) \\ \text{s.t.} \quad & K(x)u = f \\ & [K(x) + \alpha K_s(x,u)]v = 0 \\ & \alpha_1 \geq 1, \end{aligned} \tag{1}$$

where x is a vector of design variables, K the structural stiffness matrix, u the displacement vector, f the force vector, K_s the stress stiffness matrix, α a buckling load factor, with α_1 the lowest positive factor, and v a buckling mode shape eigenvector. The first two constraints comprise a linear static equation, followed by a buckling eigenvalue problem. Additional constraints on the design variables, x, may also be included. Note that K is symmetric positive definite and K_s is symmetric and, in general, indefinite. Both matrices are also usually sparse.

Following the discussion in the introduction, convergence can be improved by including multiple buckling load factors and associated modes in the problem formulation. Thus, the optimization problem becomes:

$$\begin{aligned} \min_{x} \quad & M(x) \\ \text{s.t.} \quad & K(x)u = f \\ & [K(x) + \alpha K_s(x,u)]v = 0 \\ & \alpha_m \geq \alpha_{m-1} \geq \cdots \geq \alpha_1 \geq 1, \end{aligned} \tag{2}$$

where m is the number of load factors considered. Note that a buckling eigenvalue problem must be solved each time the design is updated.

2.2. Illustrative example

A simple example is used to illustrate the advantage of including more modes in a buckling constrained optimization problem. A 2D simply supported beam, subject to compressive load, is reinforced with vertical springs, as shown in Figure 1a. The beam is discretized using 60 beam elements, that have two nodes and three degrees of freedom per node (two translations and a rotation). The axial stiffness is modelled as a simple bar and the bending stiffness is modelled as an Euler-Bernoulli beam [15]. The beam has a circular cross section with a diameter of 0.1m and Young's modulus of 10^3. A spring is attached to each free node of the discretized beam.

The optimization problem is of the form (2); the stiffness values of the reinforcing springs are the design variables x and their sum is the objective function $M(x)$. Note that for this simple example, u and K_s do not depend on the design variables, as the beam is loaded in pure compression and the springs do not affect axial stiffness. The compressive force, f, is set to six times the Euler buckling

DOI: 10.1002/nme

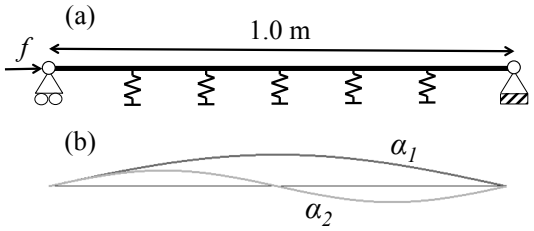

Figure 1. Beam buckling example: a) overview, b) first two buckling modes.

load of the unreinforced beam. The optimization problem is solved using the Sequential Quadratic Programming (SQP) method [31, 32] implemented in the package NLopt [33]

The beam reinforcement problem is first solved using a single mode ($m = 1$) and derivatives of the lowest positive eigenvalue with respect to the design variables are passed to the optimizer. The load factors associated with the lowest two buckling modes are tracked during optimization using an eigenvector orthogonality correlation method [34]. This procedure uses dot products between the current and previous eigenvectors to match eigenvalues with the correct modes. The first two buckling modes of the unreinforced beam are shown in Figure 1b. The convergence history of the objective function and load factors are shown in Figure 2 and the solution is shown in Figure 3a. During the first 20 iterations there is a switching of the critical mode every 2 or 3 iterations, which severely slows down convergence. This is because only the derivatives of the current lowest load factor are computed at each iteration and design changes that increase the first load factor often decrease the second, although the change in the second load factor is two orders of magnitude smaller. Despite the switching, the optimizer eventually increases both load factors and a converged feasible design is obtained in 66 iterations.

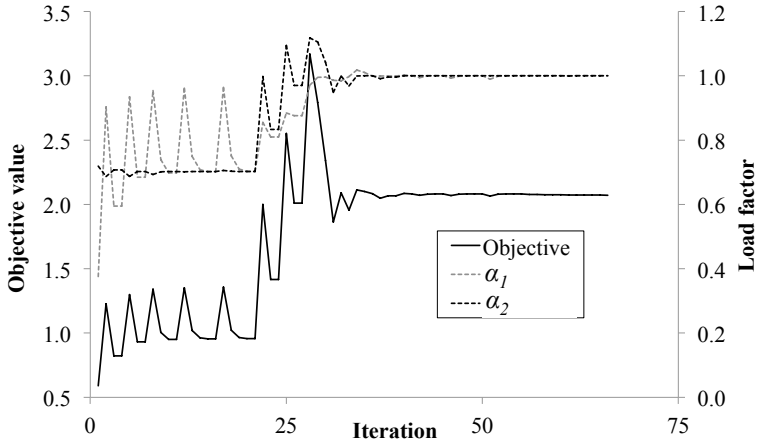

Figure 2. Convergence history using only the first buckling mode.

Figure 3. Optimal spring stiffness values for the beam reinforcement problem using a single mode with a) no tracking, b) tracking.

DOI: 10.1002/nme

One method for handling mode switching during optimization is to use mode tracking and compute derivatives for the same mode at every iteration. Using this approach, the beam reinforcement problem is again solved using a single mode (see Figure 3b). The convergence history is shown in Figure 4; 30 iterations are required. However, the lowest buckling load factor of the final design corresponds to the second buckling mode and is less than one and the design is infeasible. This occurs because we did not consider the second mode, which becomes critical.

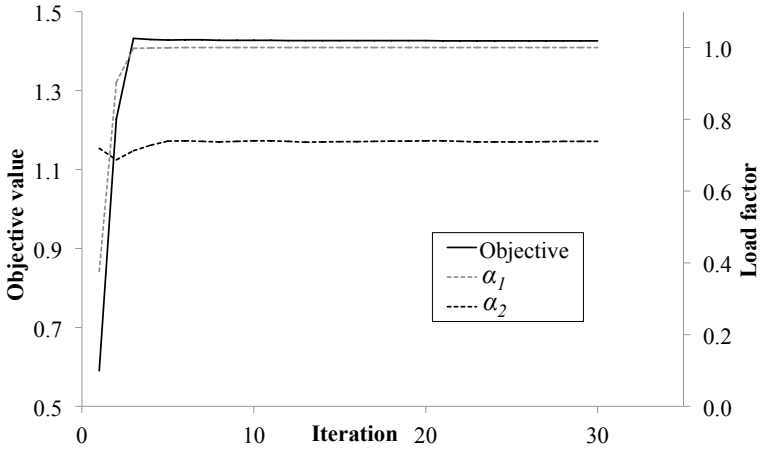

Figure 4. Convergence history using mode tracking and the first buckling mode.

This example illustrates that if we only use derivative information from the current lowest load factor, convergence may be slow while if we use mode tracking and derivative information from a single mode, an infeasible solution may be obtained. This suggests that more modes must be included. The convergence history using two modes ($m = 2$) and no mode tracking is shown in Figure 5. Convergence is achieved in 15 iterations and both the first two load factors are equal to one. The same solution is obtained when using mode tracking with the first two modes, as mode switching does not occur in this simple example.

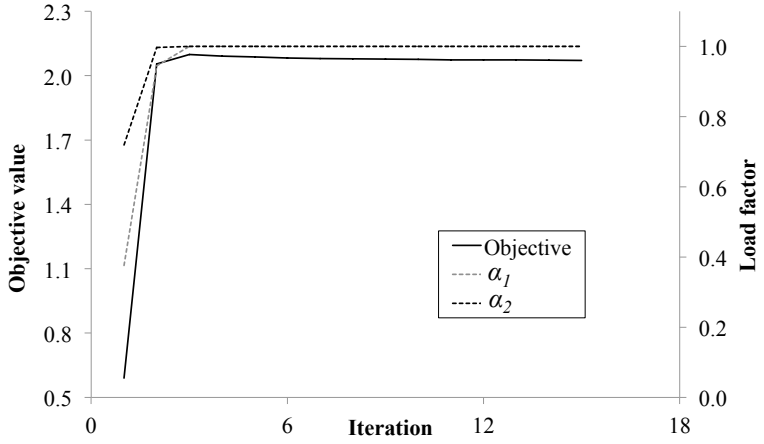

Figure 5. Convergence history using the lowest two buckling modes.

This illustrates that by using more buckling modes during optimization the number of iterations to obtain a feasible solution can be significantly reduced. For more complicated structures it may be necessary to use tens or even hundreds of modes to find a feasible design or to improve convergence. Theoretically, we need to include enough modes such that when switching occurs the most critical

DOI: 10.1002/nme

(lowest load factor) mode was included in the previous iteration. This ensures that the derivative information for the most critical mode is always included in the optimization. This is important because an optimizer can only make decisions on how to update the design from the information supplied. If critical derivative information is missing, then a gradient based optimizer may update the design such that it is further from satisfying the optimality conditions, leading to slow, or even failed convergence.

To determine *a priori* the required number of modes to ensure derivative information for the most critical mode is always included in the previous iteration throughout the optimization is not straightforward and beyond the scope of this paper. For now, we simply choose a number of buckling constraints, perform the optimization, then determine *a posteriori* if mode switching caused slow convergence. Ideally this would be done by computing a very large number of modes each iteration and then determining the order of the current eigenvalues with respect to their order in the previous iteration using a mode tracking method, as was done for the simple example in this section. For larger, more complex 3D examples this approach is not computationally feasible. Instead, a simpler approach is used whereby the same problem is solved using an increasing number of modes and the convergence histories compared. If significantly increasing the number of modes (such as doubling) does not reduce the number and magnitude of observed oscillations in the convergence of the lowest buckling load factor, then it is reasonable to assume that enough modes are included to avoid convergence issues resulting from mode switching.

3. SOLVING THE GENERALIZED EIGENVALUE PROBLEM

As we have already observed, at the heart of the buckling constrained optimization problem lies a generalized eigenvalue problem that must be solved each time the design is updated. Moreover a large number of buckling modes may be required to reduce oscillations during convergence and obtain feasible designs. In Section 3.1, we introduce the block Jacobi conjugate gradient algorithm (BJCG) for solving a generalized eigenvalue problem and then, in Section 3.2, we discuss how this algorithm may be combined with shift-invert to efficiently and robustly solve the buckling eigenvalue problem.

3.1. BJCG algorithm

We start with the problem of computing m extreme (leftmost and/or rightmost) eigenvalues and the corresponding eigenvectors of the generalized eigenvalue problem

$$ABv = \lambda v, \tag{3}$$

where A and B are real sparse symmetric $n \times n$ matrices with B positive definite and $m \ll n$. The main features of the BJCG algorithm are its high computational intensity and inherent parallelism: the bulk of the computation is taken by dense matrix-matrix products and the simultaneous computation of sparse matrix-vector products Av_i and Bv_i for a set of vectors v_i. These features facilitate efficient exploitation of highly optimized matrix-matrix multiplication subroutines from the BLAS library and modern multicore computer architectures. The BJCG algorithm is based on the block conjugate gradient method of [35, 36, 37] and is implemented as the software package `SPRAL_SSMFE`, which is available within the `SPRAL` mathematical software library [27].

In what follows, $[u_1 \ \cdots \ u_k]$ denotes the matrix with columns $u_1, \ldots, u_k$, and for $U = [u_1 \ \cdots \ u_k]$ and $W = [w_1 \ \cdots \ w_l]$ we denote $[U \ W] = [u_1 \ \cdots \ u_k \ w_1 \ \cdots \ w_l]$. Alongside the standard notation (u, w) for the Euclidean scalar product of two vectors u and w, we also use the following 'energy scalar product' and 'energy norm' notation: $(u, w)_H = (Hu, w)$ and $\|u\|_H^2 = (Hu, u)$, where H is positive definite or semi-definite. If $(u, w)_H = 0$, then we say that u and w are H-orthogonal, and if $\|u\|_H = 1$, then we say that u is H-normalized.

The BJCG algorithm for (3) essentially performs simultaneous optimization (minimization or maximization or both depending on which eigenvalues are of interest) of the Rayleigh quotient

Prepared using nmeauth.cls
DOI: 10.1002/nme

functional

$$\rho(u) = \frac{(BABv, v)}{(Bv, v)} = \frac{(ABv, v)_B}{\|v\|_B} \tag{4}$$

on sets of B-orthogonal vectors by a block version of the conjugate gradient (CG) method. Focusing for simplicity on the minimization case, we observe that the CG method minimizes a functional $\psi(u)$ by iterating two vectors, the current approximation v^i to the minimum point $v_* = \arg\min \psi(v)$ and the current search direction u^i along which the next approximation v^{i+1} is sought in the following manner:

$$v^{i+1} = v^i - \tau_i u^i, \tag{5}$$

$$u^{i+1} = \nabla\psi(v^{i+1}) + \sigma_i u^i. \tag{6}$$

Step (5) finds the minimum of $\psi(v)$ in the direction set by u^i, i.e. $\tau_i = \arg\min \psi_i(\tau)$, where $\psi_i(\tau) = \psi(v^i - \tau u^i)$. Step (6), referred to as the conjugation of search directions, computes the new search direction u^{i+1} based on the gradient of $\psi(v)$ at $v = v^{i+1}$ and the previous search direction u^i (not present at the first iteration). The role of the scalar parameter σ_i is to make the search direction closer to the ideal one, the error direction $v^i - v_*$, which would result in an immediate convergence on the next iteration. In the case of a quadratic functional $\psi(v) = (Av, v) - 2(f, v)$ with a symmetric positive definite A, the minimum of which solves the linear system $Av = f$, it is possible to choose σ_i in such a way that all v^i are optimal, i.e. each $\psi(v^i)$ is the smallest possible value achievable by iterations (5)–(6). In the non-quadratic case, such optimality is generally not achievable. In [35] various suggestions for σ_i have been studied thoroughly to arrive at the conclusion that most of them are asymptotically equivalent and make the value $\psi(v^{i+2})$ 'nearly' the smallest possible for a given v^{i+1} and u^i. It should be noted that the value of σ_i that makes $\psi(v^{i+2})$ the smallest possible can be computed by finding the minimum of $\psi(v)$ among all linear combinations of v^{i+1}, $\nabla\psi(v^{i+1})$ and u^i [30, 38], however, this *locally optimal* version of CG is numerically unstable [37].

When computing several eigenpairs by simultaneous optimization of the Rayleigh quotient (4), it is necessary to maintain the B-orthogonality of approximate eigenvectors v^i_j lest they converge to the same eigenvector corresponding to an extreme eigenvalue. One way to achieve this, which has an additional benefit of improved convergence, is to employ the Rayleigh–Ritz procedure. Given a set of vectors $z_1, \ldots, z_k$, $k < n$, this procedure solves the generalized eigenvalue problem

$$Z^T BABZ\hat{v} = \hat{\lambda} Z^T BZ\hat{v}, \tag{7}$$

where $Z = [z_1 \ \cdots \ z_k]$. The vectors $\tilde{v}_j = Z\hat{v}_j$, where $\hat{v}_j$ are the eigenvectors of (7), are called Ritz vectors, and the corresponding eigenvalues $\hat{\lambda}_j = \rho(\tilde{v}_j)$ are called Ritz values. In what follows, we refer to z_i as the basis vectors of the Rayleigh–Ritz procedure, and we use the following 'function call' notation for this procedure: for a given $Z = [z_1 \ \cdots \ z_k]$, $Y = \texttt{RayleighRitz}(Z)$ is the matrix whose columns are B-normalized Ritz vectors. The Ritz vectors are B-orthogonal by construction, and if there exists a linear combination of the basis vectors that approximates an eigenvector of (3), then there exists a Ritz vector that approximates this eigenvector to a comparable accuracy [39].

The above two ingredients, CG and the Rayleigh–Ritz procedure, are blended in the BJCG algorithm in the following way. The line search step (5) is replaced by the Rayleigh–Ritz procedure that uses all approximate eigenvectors v^i_j and all search directions u^i_j as the basis vectors, and a matrix analogue of (6) is used for conjugation, which after simple manipulations yields the iterative scheme

$$[V^{i+1} \ W^{i+1}] = \texttt{RayleighRitz}(V^i, U^i), \tag{8}$$

$$U^{i+1} = R^{i+1} + W^{i+1} S_i. \tag{9}$$

In (8), the columns of V^{i+1} approximate the eigenvectors of interest, i.e. if we are computing $m_l \geq 0$ leftmost and $m_r \geq 0$ rightmost eigenvalues and corresponding eigenvectors, then the $m = m_l + m_r$ columns of V^{i+1} correspond to the m_l leftmost and m_r rightmost Ritz values. The rest of the Ritz vectors are the columns of W^{i+1}, and are used as the previous search directions

DOI: 10.1002/nme

for conjugation (this is shown by [36] to be mathematically equivalent to using U^i in (9) but simplifies the calculation of the optimal conjugation matrix S_i). In (9), $R^{i+1} = [r_1^{i+1} \cdots r_m^{i+1}]$, where $r_j^{i+1} = ABv_j^{i+1} - \rho(v_j^{i+1})v_j^{i+1}$ are the residual vectors for v_j^{i+1}, which are collinear to the gradients of ρ in B-scalar product. For the elements σ_{pq} of the matrix S_i the following formulae were derived in [36] that ensure the asymptotic local optimality of the new search directions:

$$\sigma_{pq} = -\frac{(r_q^{i+1}, BABw_p^{i+1} - \rho(v_q^{i+1})Bw_p^{i+1})}{\rho(w_p^{i+1}) - \rho(v_q^{i+1})}. \tag{10}$$

While the columns of V^i are B-orthonormal, some of the columns of U^i may be nearly linearly dependent on the rest of the basis vectors. Such vectors are removed from the set of basis vectors to avoid numerical instability caused by the ill-conditioned matrix Z^TBZ in (7). The number of Ritz vectors then reduces accordingly, and S_i has less rows than columns.

We observe that the most costly computational steps of the described algorithm in terms of the number of arithmetic operations are:

1. The computation of sparse-by-dense matrix products $V_B^i = BV^i$, AV_B^i, $U_B^i = BU^i$ and AU_B^i.
2. The computation of dense matrix-matrix products $Z_B^T Z_A$ and $Z^T Z_B$ with $Z = [V^i \ U^i]$, $Z_A = AZ_B$ and $Z_B = BZ$.
3. The computation of V^{i+1} and W^{i+1} from the eigenvectors of (7) by dense matrix-matrix products.
4. The computation of sparse-by-dense matrix products $W_B^{i+1} = BW^{i+1}$ and AW_B^{i+1}.
5. The computation of a dense matrix-matrix product $W^{i+1}S_i$.

(The A- and B-images of V^{i+1} and W^{i+1} can be alternatively computed from those of V^i and U^i, which reduces the number of multiplications by A and B threefold at the cost of extra storage [30].) The computational cost of the rest of the algorithm is negligible if $m \ll n$. We observe that steps 2, 3 and 5 require dense matrix-matrix multiplications that can be efficiently performed by highly optimized BLAS subroutines, e.g. those provided by Intel MKL library, which also has highly optimized subroutines for sparse-by-dense matrix multiplications performed on steps 1 and 4.

3.2. Application of BJCG to the buckling eigenvalue problem

In this section, we show how we use the BJCG algorithm to compute buckling modes, i.e. eigenvalues α_i and eigenvectors v_i of the generalised eigenvalue problem

$$(K + \alpha K_s)v = 0, \tag{11}$$

where K and K_s are $n \times n$ real sparse symmetric matrices, K is positive definite and K_s is indefinite. The eigenvalues of interest lie in the vicinity of 1, notably: any eigenvalues α_i that lie in the interval $(0, 1]$ are required as well as several eigenvalues $\alpha_i > 1$ that are closest to 1, together with the corresponding eigenvectors. As in the previous section, m denotes the number of wanted eigenvalues; all eigenvalues are assumed to be enumerated in ascending order.

We observe immediately that (11) can be rewritten as the generalized eigenvalue problem

$$K_s v = \beta K v, \tag{12}$$

where $\beta = -\alpha^{-1}$. The BJCG algorithm can be applied to this problem but convergence may be very slow, as can be seen from the following simplified illustration that extends the informal discussion of the convergence of block CG given in [30].† Suppose that we start the BJCG iterations with the initial vectors $v_j^0 = v_j$, $j = 2, \ldots, m$ and with v_1^0 close to v_1 and K-orthogonal to $v_2, \ldots, v_m$. It can be shown that the BJCG iterations then reduce to single-vector CG iterations

†Rigorous convergence analysis of CG for eigenvalue computation is a very difficult, and virtually all available results merely state that the convergence of CG is at least as fast as that of steepest descent, whereas in reality the convergence of CG is much faster and is more adequately portrayed by a conjectured estimate of [30] and those of this section.

DOI: 10.1002/nme

(5)–(6) for computing the zero eigenvalue of $A_1 = K_s - \beta_1 K$, performed in the subspace that is A_1-orthogonal to $v_2, \ldots, v_m$. Using the comparative convergence analysis techniques developed in [35], it can be shown that these iterations are asymptotically close to the CG iterations for the system $A_1 v_1 = 0$ in that subspace. For the error $v_1^i - v_1$, we have $\|v_1^i - v_1\|_{A_1}^2 = (A_1(v_1^i - v_1), v_1^i - v_1) = (A_1 v_1^i, v_1^i) = (\beta_1^i - \beta_1)(K v_1^i, v_1^i)$, where $\beta_1^i = (K_s v_1^i, v_1^i)/(K v_1^i, v_1^i)$. Therefore, from the standard CG convergence estimate for $\|v_1^i - v_1\|_{A_1}$ we obtain, after dropping asymptotically small terms, an estimate similar to that conjectured in [30]:

$$\beta_1^i - \beta_1 \lesssim \left(\frac{\sqrt{\kappa_1} - 1}{\sqrt{\kappa_1} + 1}\right)^{2i} (\beta_1^0 - \beta_1), \tag{13}$$

where κ_1 is the ratio of the n-th to $(m+1)$-th eigenvalue of the matrix $K_s - \beta_1 K$ (for BJCG, a similar estimate without square roots is proved in [36]). Since $\beta_1 \approx -1$ and K is the discretization of a partial differential operator, the n-th (i.e. the largest) eigenvalue of $K_s - \beta_1 K$ is very large, and hence the ratio in question is very small, leading to unacceptably slow convergence.

To achieve good convergence, we resort to the shift-invert technique. We pick a value α_0 in the eigenvalue region of interest and rewrite (11) as

$$(K + \alpha_0 K_s)^{-1} K v = \gamma v, \tag{14}$$

where

$$\gamma = \frac{\alpha}{\alpha - \alpha_0}. \tag{15}$$

We observe that (14) is of the form (3) with:

$$A = (K + \alpha_0 K_s)^{-1}, B = K \tag{16}$$

and hence can be solved by the BJCG algorithm.

To illustrate how the transformation from (11) to (14) improves the convergence, let us assume for simplicity that $\alpha_0 = (\alpha_m + \alpha_{m+1})/2$. Then the leftmost eigenvalue of (14) is $\gamma_1 = \alpha_m/(\alpha_m - \alpha_0) < 0$, the rightmost is $\gamma_n = \alpha_{m+1}/(\alpha_{m+1} - \alpha_0) > 0$, and the number of negative γ_i is m, i.e. $\gamma_{m+1} \geq 0$. Just as with (12), the convergence rate for γ_1 is determined by the ratio of the n-th to $m+1$-th eigenvalue of the matrix $(K + \alpha_0 K_s)^{-1} K - \gamma_1 I$, where I is the identity matrix, i.e. by the ratio

$$\kappa_1 = \frac{\gamma_n - \gamma_1}{\gamma_{m+1} - \gamma_1} \leq -\frac{\gamma_n - \gamma_1}{\gamma_1} = 1 - \frac{\gamma_n}{\gamma_1} = 1 - \frac{\alpha_{m+1}}{\alpha_m}\frac{\alpha_m - \alpha_0}{\alpha_{m+1} - \alpha_0} = 1 + \frac{\alpha_{m+1}}{\alpha_m} \approx 2.$$

The substitution of this κ_1 into (13) with γ_1 and its approximations γ_1^i in place of β_1 and β_1^i yields

$$\gamma_1^i - \gamma_1 \lesssim \left(\frac{\sqrt{2} - 1}{\sqrt{2} + 1}\right)^{2i} (\gamma_1^0 - \gamma_1) \approx 0.03^i (\gamma_1^0 - \gamma_1).$$

Thus, just two iterations reduce the eigenvalue error by three orders of magnitude.

When dealing with eigenvalues on both ends of the spectrum simultaneously, it is convenient to use positive indices for the eigenvalues on the left margin: $\gamma_1 \leq \gamma_2 \leq \cdots$, and negative for those on the right margin: $\gamma_{-1} \geq \gamma_{-2} \geq \cdots$ (i.e. $\gamma_{-j} = \gamma_{n-j+1}$). Let m_l and m_r be the number of wanted eigenvalues of (11) to the left and right of α_0 respectively. The same kind of reasoning as we used for deriving (13) suggests the following estimate for further eigenvalues of (14):

$$\gamma_j^i - \gamma_j \lesssim \left(\frac{\sqrt{\kappa_j} - 1}{\sqrt{\kappa_j} + 1}\right)^{2i} (\gamma_j^0 - \gamma_j), \tag{17}$$

where

$$\kappa_j = \frac{\gamma_{-m_r - 1} - \gamma_j}{\gamma_{m_l + 1} - \gamma_j}, \quad \kappa_{-j} = \frac{\gamma_{-j} - \gamma_{m_l + 1}}{\gamma_{-j} - \gamma_{-m_r - 1}}, \quad j > 0. \tag{18}$$

The estimate (17)–(18) has practical implications, in particular the following one.

DOI: 10.1002/nme

Remark 1
Since the density of the spectrum increases towards $\gamma = 1$, for a large m_r, the value $\gamma_{-m_r} - \gamma_{-m_r-1}$ may be very small. Therefore, to avoid very slow convergence to γ_{-j} that are close to γ_{-m_r-1}, the block size m of BJCG should be set to a value that is larger than the actual number of wanted eigenvalues. The effect of the increased block size can be seen from (18) where, if the block size is increased by $k > 0$, one should replace γ_{-m_r-1} with γ_{-m_r-k-1}, which reduces both κ_j and κ_{-j}. Note that the additional eigenvalues and eigenvectors need not be accurately computed. i.e. the BJCG iterations should be stopped as soon as $\gamma_1, \ldots, \gamma_{m_l}$ and $\gamma_{-1}, \ldots, \gamma_{-m_r}$ and corresponding eigenvectors converge to required accuracy.

We note that the spectral transformation (15) maps the eigenvalues as follows. The eigenvalues in the interval $(0, \alpha_0)$ correspond to all negative γ_i. The eigenvalues of interest to the right of α_0 correspond to the rightmost γ_i. The rest of the spectrum of (11) corresponds to a huge cluster of γ_i around $\gamma = 1$. When using BJCG for solving (14), it is very important to set the number of wanted leftmost eigenvalues m_l exactly to the number of negative γ_i. Setting m_l to a larger value will result in a large number of iterations, as BJCG will try to compute an eigenvalue in the cluster at $\gamma = 1$. For this reason, using an LDLT factorization for solving the shifted system $(K + \alpha_0 K_s)x = y$ (this is needed for computing the product Ay (16) in BJCG) is strongly recommended (e.g. the direct solvers `HSL_MA97` [25] or `SPRAL_SSIDS` [27] may be used). Recall that an LDLT factorization of the matrix $K + \alpha_0 K_s$ consists in finding a unit lower triangular matrix L, a block-diagonal matrix D with 1×1 and 2×2 blocks on the main diagonal and a permutation matrix P such that $P^T(K + \alpha_0 K_s)P = LDL^T$. By the inertia theorem, the number of eigenvalues to the left and right of the shift α_0 is equal to the number of negative and positive eigenvalues of D, which allows quick computation of the number of eigenvalues to each side of the shift after. Knowing the number of eigenvalues to the left of α_0 also helps avoid re-computation of the already computed buckling modes after a new shift.

Remark 2
If the number of wanted buckling modes is very large, or the number of eigenvalues on each side of the shift α_0 are disproportional, multiple shifts should be considered. Note that for each choice of shift α_0, a factorization of $K + \alpha_0 K_s$ is required. Once the factorization has been performed, the factors may be used repeatedly for solving any number of systems.

The decision on whether to use more than one shift can be made on the basis of a 'trial run', whereby a portion (e.g. a quarter) of the wanted buckling modes is computed, and the time taken by BJCG is extrapolated and compared with the factorization time. This can also be used to select a better shift, based on the number of eigenvalues each side of the initial shift. If no new shift is deemed to be necessary, the remaining buckling modes can be computed by BJCG in the subspace K-orthogonal to the computed eigenvectors.

In this paper, the shift-invert solves are performed using a sparse direct linear solver. Alternatively, an iterative method could be used. Such methods have the potential advantage of requiring much less memory and so can be used to solve very large linear systems of equations. However, in general, to achieve an acceptable rate of convergence, a preconditioner is required; unfortunately, designing an efficient preconditioner is highly problem dependent and we are not aware of such preconditioners for linear buckling problems. Note that if we were able to obtain such a preconditioner, it would be unnecessary to solve the shifted system using a preconditioned iterative method since `SPRAL_SSMFE` can employ preconditioning directly, a feature not shared by Krylov solvers, by essentially applying BJCG iterations to the problem $T(K + \alpha K_s)v = 0$, where T is the preconditioner, without first converting to the shifted system (14). The design of an efficient preconditioner is beyond the scope of this paper and is a topic for future research.

3.3. Application to topology optimization

As we have seen in §2, topology optimization algorithms require the solution of a sequence of buckling eigenvalue problems. As the outer (optimization) iterations progress, these problems

DOI: 10.1002/nme

become ever closer to each other, which suggests using data from the previous outer iteration for the acceleration of BJCG convergence on the next outer iteration

An obvious approach is to use the buckling modes from the previous outer iteration as the initial approximations $v_1^0, \ldots, v_m^0$ for BJCG. Another option is to use the approximate eigenvalues computed on the previous outer iteration and the estimate (17)–(18) to select α_0. Note that κ_j is an increasing function of γ_j, and κ_{-j} is a decreasing function of γ_{-j}. Hence, their maxima are κ_{m_l} and κ_{-m_r}, respectively. A good choice for α_0 therefore should make the maximum of these two values the minimal possible. There is an obstacle however: γ_{m_l+1} may be difficult to compute. To bypass this, assume that any negative α_i is sufficiently far away from zero, so that $\gamma_{m_l+1} \approx 1$. Let $\tilde{\alpha}_j$, $j = 1, m$, be the eigenvalues of (11) computed on the previous outer iteration. Assuming $\gamma_{m_l+1} = 1$, we obtain from (15) and (18) via elementary calculation

$$\kappa_{m_l} = \frac{\alpha_{m+1} - \tilde{\alpha}_1}{\alpha_{m+1} - \alpha_0}, \quad \kappa_{-m_r} = \frac{\alpha_{m+1} - \alpha_0}{\alpha_{m+1} - \tilde{\alpha}_m}. \tag{19}$$

We observe that κ_{m_l} is an increasing function of α_0 and κ_{-m_r} is a decreasing one. Hence, the maximum of the two is minimal when $\kappa_{m_l} = \kappa_{-m_r}$, which is attained at

$$\alpha_0 = \alpha_{m+1} - \sqrt{(\alpha_{m+1} - \tilde{\alpha}_1)(\alpha_{m+1} - \tilde{\alpha}_m)}. \tag{20}$$

In practice, α_{m+1} is replaced by an approximation. We also note that if the block size of BJCG is increased by $k > 0$ to improve convergence (cf. Remark 1), then in (20) α_{m+1} should be replaced by an approximation to α_{m+k+1}, or simply by $q\tilde{\alpha}_m$, where $q > 1$ (e.g. $q = 1.1$, which roughly corresponds to $k = m/10$), which yields

$$\alpha_0 = q\tilde{\alpha}_m - \sqrt{(q\tilde{\alpha}_m - \tilde{\alpha}_1)(q-1)\tilde{\alpha}_m}. \tag{21}$$

This paper assumes a linear relationship between loads, deflections and stresses and that deflections are small. These assumptions allow the use of a linear buckling analysis, which is an efficient method for estimating the critical buckling load factor and is useful in engineering design. However, linear buckling analysis often overestimates the critical load when imperfections and nonlinearities are accounted for. When these factors are significant, nonlinear buckling analysis can be performed. For topology optimization of geometrically nonlinear structures, Lindgaard and Dahl [12] suggest using the arc length method for geometrically nonlinear analysis until an instability is detected. A buckling eigenvalue analysis is then performed for the structure in the configuration one step before the instability occurs. Therefore, when using this approach, it may be possible to use the BJCG method to reduce the computational cost of the eigenvalue analysis when performing optimization with nonlinear buckling constraints.

4. PERFORMANCE OF THE EIGENVALUE SOLVER

In this section, we use two linear buckling problems to benchmark the performance of our BJCG eigenvalue solver `SPRAL_SSMFE` and to highlight its potential for the efficient computation of a large number of buckling eigenmodes during optimization. The two examples are rectangular blade stiffened panels, simply supported and loaded in compression along their short edges, see Figure 6. Both panels are 1 x 0.5 m and have evenly spaced blade stiffeners. The depth of the stiffeners is 25 mm and 30 mm, respectively. The material properties are Young's modulus of 70GPa and Poisson's ratio of 0.32. The panels are discretized using 3-node shell elements, with six degrees of freedom (dof) per node. The thickness of the shells are 4 mm and 3 mm, respectively. Different levels of discretization are used to test the scalability of `SPRAL_SSMFE`.

The tests are performed on a 16-core Intel(R) Xeon(R) E5-2687W CPU. and the GNU `gfortran` compiler with flags `-O3 -fopenmp` are used. The factorization of the shifted matrices $K + \alpha_0 K_s$ and subsequent solves is performed by `HSL_MA97` [26] run in parallel mode. The shifted solves are performed by the solver subroutines from `HSL_MA97` called in an OpenMP parallel do loop each using one MKL thread.

DOI: 10.1002/nme

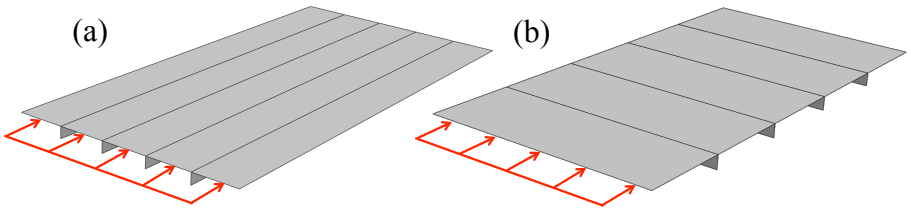

Figure 6. Buckling examples: a) Panel 1, b) Panel 2.

Unless specified explicitly, a shift of $\alpha_0 = 1$ and an eigenvector error tolerance of 10^{-4} is used in the reported tests. The BJCG block size for computing k modes is set to $k + max(10, k/10)$. All reported timings are measured using the function `clock_gettime` and are in seconds.

Table I displays `SPRAL_SSMFE` solve and `HSL_MA97` factorization timings for different levels of discretization for Panel 1, where 100 buckling eigenmodes are computed. The third column presents the solve times divided by the dof in thousands. We observe a nearly-linear scaling of the solve times with the problem size. Similar results are obtained for Panel 2. This shows the potential of `SPRAL_SSMFE` to efficiently compute buckling eigenvalues and eigenvectors for large-scale optimization problems.

Table I. Timings for Panel 1 using different discretizations.

problem size (dof)	solve time	time per 10^3 dof	factorization time
8902	0.7	0.083	0.1
17576	1.4	0.079	0.2
37625	3.1	0.081	0.3
74383	6.7	0.090	0.5
144823	12.1	0.084	1.0
224522	19.6	0.087	1.6
394962	36.3	0.092	2.9
577643	54.7	0.095	4.6
728142	61.0	0.084	5.1
908065	83.1	0.092	7.4

Table II shows performance data for Panel 1 (using the 577,643 dof discretization) for different numbers of buckling modes. The block size m is set to the number of modes plus 10. We observe that the number of BJCG iterations remains practically unchanged as the number of modes increases, thanks to the increase in the block size, which improves the convergence to leftmost modes (see [36] for the convergence analysis of BJCG). The last two columns show that the time per mode and the number of shifted solves per mode decreases for larger numbers of modes. This is a promising result when we consider that a large number of buckling eigenmodes may be required during optimization, as discussed in Section 2.2.

Table II. Performance data for different numbers of modes for Panel 1 with 577,643 dof.

modes	solve time	iterations	shifted solves	time per mode	shifted solves per mode
20	22.5	16	443	1.13	22
40	29.7	17	712	0.74	18
60	38.1	17	981	0.64	16
80	48.9	14	1145	0.61	14
100	54.7	16	1394	0.55	14

Table III shows the results of using different shifts α_0 for computing 100 modes for Panel 1 (using the 577,643 dof discretization). We observe that the minimal number of BJCG iterations and shifted

DOI: 10.1002/nme

solves is attained at $\alpha_0 = 20$, which is very close to the optimal shift value of 20.3 predicted by (21) with $q = 1.1$.

Table III. Performance data for different shifts for Panel 1 with 577,643 dof.

shift α_0	eigenvalues left of α_0	eigenvalues right of α_0	solve time	iterations	shifted solves
1	0	100	54.7	16	1394
5	4	96	52.5	14	1277
10	10	90	50.7	13	1172
15	30	70	44.3	11	1055
19	49	51	43.1	11	1040
20	55	45	41.3	11	1020
21	65	35	40.4	13	1027
21.3	66	34	40.0	13	1042
22	68	32	40.9	14	1062
25	88	12	49.2	17	1233

Table IV shows analogous results for Panel 2 (using a 539,291 dof discretization). In this case, the best shift (in terms of the iteration number) is $\alpha_0 = 11$, which is further from the theoretically optimal $\alpha_0 = 14.4$. This is due to the rapidly increasing density of the spectrum to the right from α_0, which makes the linear extrapolation $\alpha_{m+k+1} \approx (1 + (k+1)/m)\tilde{\alpha}_m$ used in (21) inaccurate.

Table IV. Performance data for different shifts for Panel 2 with 539,291 dof.

shift α_0	eigenvalues left of α_0	eigenvalues right of α_0	solve time	iterations	shifted solves
1	0	100	47.1	16	1275
2	3	97	46.0	16	1256
3	8	92	46.4	15	1211
4	15	85	43.0	15	1169
5	20	80	46.5	16	1168
6	23	77	43.5	14	1139
7	32	68	42.3	14	1125
8	37	63	39.4	12	1095
9	42	58	38.3	12	1080
10	49	51	38.5	11	1097
11	56	44	39.7	10	1092
12	63	37	40.3	11	1132
13	67	33	40.5	11	1157
14	73	27	43.7	14	1196
14.4	76	24	46.9	14	1233

Remark 3
To highlight potential problems with employing ARPACK in topology optimization, we used ARPACK to compute the 100 leftmost eigenvalues of Panel 2 (539,291 dof) with $\alpha_0 = 9$ and maximal accuracy, i.e. the ARPACK tolerance was set to zero. The leftmost eigenvalue $\alpha_1 = -1.1434995674$ is closer to α_0 than $\alpha_{100} = -18.472317385$, so we expected that ARPACK would compute all the wanted 42 eigenpairs that lie to the left of α_0. However, 22 were missing, i.e. ARPACK computed α_{23} to α_{122} instead of wanted α_1 to α_{100}

In Table V, we compare the computation of 200 modes using one shift α_0 for the computation of 100 modes and then the remaining 100 modes using a new shift $\alpha_{100} + 5 \times (\alpha_{100} - \alpha_{90}) \approx \alpha_{150}$. For this example, we observe that when computing 200 modes, two shifts are preferable to a single shift. However, the opposite is observed when a total of 100 modes are computed. Thus, the multiple shift strategy may provide additional efficiency gains during optimization when a large number of buckling modes is required.

Table VI compares the solve times for 100 modes for Panels 1 and 2 for various eigenvector error tolerances. It takes approximately twice as long to compute eigenvectors with 10^{-8} accuracy

DOI: 10.1002/nme

Table V. Times using a single shift and two shifts for 200 modes for Panels 1 and 2.

shifts	Panel 1			Panel 2		
	factorization	solve	total	factorization	solve	total
1	4.6	143.9	148.5	4.2	109.7	113.9
2	9.2	114.9	124.1	8.4	98.6	107.0

as with 10^{-2} accuracy. This suggests that using a relaxed tolerance at early outer iterations, when the computed modes are far from optimal, and then a progressively smaller tolerance as the outer iterations proceed is a potential strategy for reducing the overall optimization time.

Table VI. Times for different eigenvector error tolerances for 100 modes for Panels 1 and 2.

tolerance	Panel 1	Panel 2
10^{-2}	42.6	39.5
10^{-4}	54.7	47.1
10^{-6}	65.3	60.0
10^{-8}	80.9	79.3

Figure 7 provides a justification for the use of the eigenvector error tolerance of 10^{-4} in most tests by comparing the estimated discretization and solution errors in eigenvalues of Panel 1 using 728,142 dof. The discretization error is computed as the difference between the eigenvalues at two different discretizations (728,142 and 908,065 dof), both computed with the eigenvector tolerance set to 10^{-8}, which makes the error in the computed eigenvalues close to machine accuracy. The solution error is computed as the difference between eigenvalues with eigenvector tolerances of 10^{-4} and 10^{-8}. We clearly observe that this difference is several orders of magnitude below the discretization error, and hence using an eigenvector error tolerance of 10^{-4} provides useful results for this example.

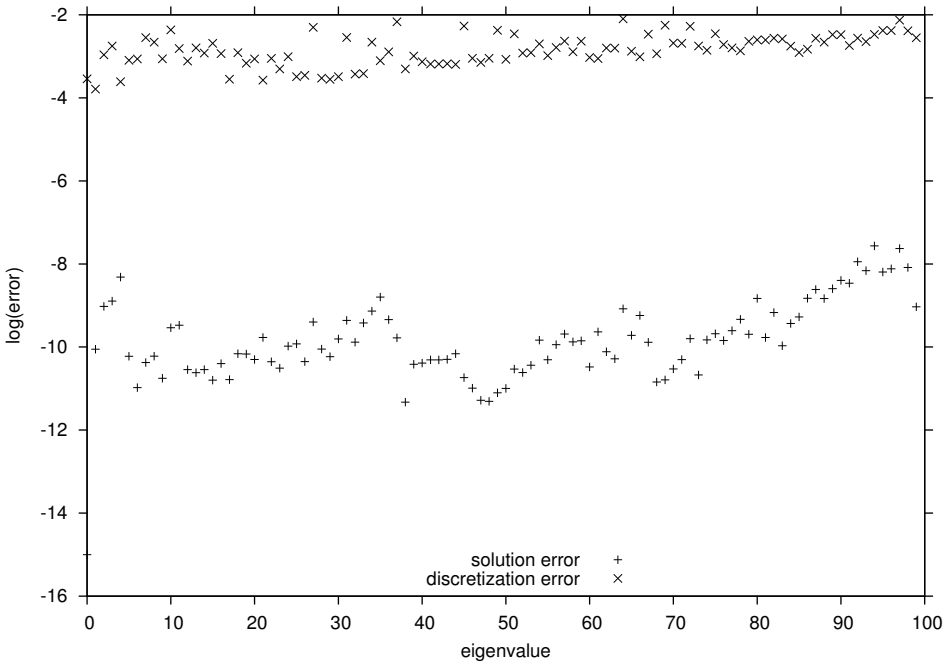

Figure 7. Estimated discretization and solution errors in eigenvalues of Panel 1

DOI: 10.1002/nme

In summary, we have demonstrated the potential for the BJCG eigensolver `SPRAL_SSMFE` to efficiently compute a large number of buckling modes during optimization. Particular features that are promising for efficient computation are: almost linear scaling with problem size, better than linear scaling with the number of wanted modes, the ability to estimate the optimal shift from previous eigenvalues, potential for efficiency gains using multiple shifts and the potential to use an adaptive eigenvector convergence tolerance to reduce the overall optimization time. These features will be explored and demonstrated further using a topology optimization example problem in Section 6.

5. LEVEL-SET TOPOLOGY OPTIMIZATION WITH BUCKLING CONSTRAINTS

In this section we introduce linear buckling constraints for the conventional level-set method.

5.1. The level-set method

In level-set topology optimization, the structure is defined using an implicit function

$$\begin{cases} \phi(\chi) \geq 0 & \chi \in \Omega, \\ \phi(\chi) = 0 & \chi \in \Gamma, \\ \phi(\chi) < 0 & \chi \notin \Omega, \end{cases} \tag{22}$$

where Ω is the structure domain, Γ the structure boundary, $\phi(\chi)$ the implicit function and $\chi \in \Omega_d$, where Ω_d is the design domain containing the structure, $\Omega \subset \Omega_d$, as shown in Figure 8. Thus, the structure boundary is defined as the zero level-set of the implicit function. The design domain is discretized such that the implicit function is defined at a finite number of points and interpolated between these points using shape functions in a similar manner to a finite element (FE) discretization. Several optimization methods use implicit functions to describe the structure. In this

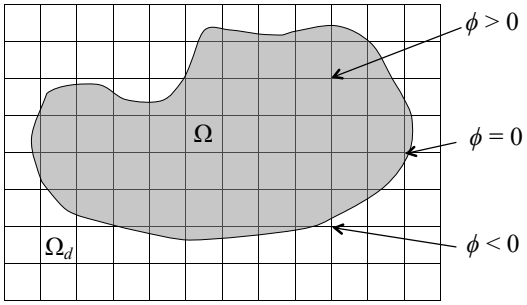

Figure 8. A structure described using an implicit function

paper, the conventional level-set method is used [3]. The implicit function is initialized as a signed distance function, where the magnitude is the shortest distance to the structure boundary and the sign is defined by (22). The implicit function is updated by solving the following advection equation:

$$\frac{\partial \phi}{\partial t} + \nabla \phi(\chi, t) \frac{d\chi}{dt} = 0, \tag{23}$$

where t is the structural optimization evolution time. Equation (23) can be discretized and rearranged to give an update formula for optimization:

$$\phi_i^{k+1} = \phi_i^k + \Delta t |\nabla \phi_i(\chi)| V_{n,i}, \tag{24}$$

where k is the outer iteration number, i a discrete grid point, Δt the time step and $V_{n,i}$ is a velocity function defined normal to the boundary and a positive value indicates inward boundary

DOI: 10.1002/nme

movement. The velocity function dictates the change in the implicit function and the movement of the zero level-set that defines the structure boundary. Maintaining the signed distance property of the implicit function is important for the stability of the conventional level-set method. In our implementation, velocity values are only defined at the boundary and then extended to all grid nodes. We use a velocity extension strategy that maintains the signed distance function by keeping velocity values constant along a line normal to the boundary. We also reinitialize the implicit function to a signed distance function every 30 iterations during optimization. For full details of our numerical implementation of the conventional level-set method in 3D see [40].

To perform optimization with the level-set method, the velocity function in (24) is usually defined using shape derivatives of the objective and constraint functions. Shape derivatives usually take the form of a boundary integral of a shape sensitivity function multiplied by the velocity function:

$$\frac{\partial f(\Omega)}{\partial \Omega} = \int_{\Gamma} (s_f V_n) d\Gamma, \tag{25}$$

where s_f is the shape sensitivity function for generic function f. In this work, the velocity function is defined as a weighted sum of the shape sensitivity functions for the objective and constraints

$$V_n(\lambda) = \lambda_f s_f + \sum_{i=1}^{m} \lambda_i s_i, \tag{26}$$

where s_f and s_i are the shape sensitivity functions for the objective and constraints, respectively, and λ are the weight values. The weights are obtained by solving the following sub-problem:

$$\begin{array}{ll} \min\limits_{\lambda} & \Delta f\left(V_n(\lambda)\right) \\ \text{s.t.} & \Delta g_i\left(V_n(\lambda)\right) \leq G_i \ , \ i = 1 \ldots m \\ & \lambda_{min} \leq \lambda \leq \lambda_{max} \end{array} \tag{27}$$

where G_i is the target change for constraint function g_i, set to maintain constraint feasibility, and Δf is an approximation for the change in f for a given velocity function, which is obtained by using numerical integration to evaluate (25). Full details on this approach for handling constraints in level-set based optimization are given in [41].

5.2. Buckling constraints

An important aspect of structural optimization using the level-set method is the formulation of the structural stiffness and mass properties from the implicit function. Here, an efficient fixed grid approach is used, where the background FE mesh remains fixed and the properties of elements cut by the boundary are approximated using a volume weighted approach:

$$\begin{aligned} \bar{E}_i &= \mu_i E, \\ \bar{\rho}_i &= \mu_i \rho, \end{aligned} \tag{28}$$

where E and ρ are the Young's modulus and density of the structural material, the over bar denotes the effective material properties for element i used to build the FE matrices and μ_i is the volume-fraction, defined as:

$$\mu_i = \frac{\overline{Vol}_i}{Vol_i}(1 - \mu_{min}) + \mu_{min}, \tag{29}$$

where $\overline{Vol}_i$ is the volume of element i that lies inside the structure, Vol_i is the total volume of the element and μ_{min} is a small positive value to avoid singular matrices, $\mu_{min} = 10^{-6}$.

A well-known issue in topology optimization involving linear buckling is the occurrence of spurious modes in the void region of the design space [11, 12]. One method for preventing such modes is to penalize stresses in elements with a low volume-fraction, which effectively increases the eigenvalues of the spurious modes making them non-critical. We use a continuous function that

DOI: 10.1002/nme

depends on the element volume-fraction to penalize stresses:

$$\overline{\mu}_i(\mu_i) = \frac{\mu_i}{1 - \mu_{min}(1 - \mu_i^{-p})}, \tag{30}$$

where $\overline{\mu}_i$ is a factor that multiplies the stresses in element i, which are used to compute the element stress stiffness matrix, and p is a penalization factor, chosen here as 3. The effect of the penalization is shown in Figure 9.

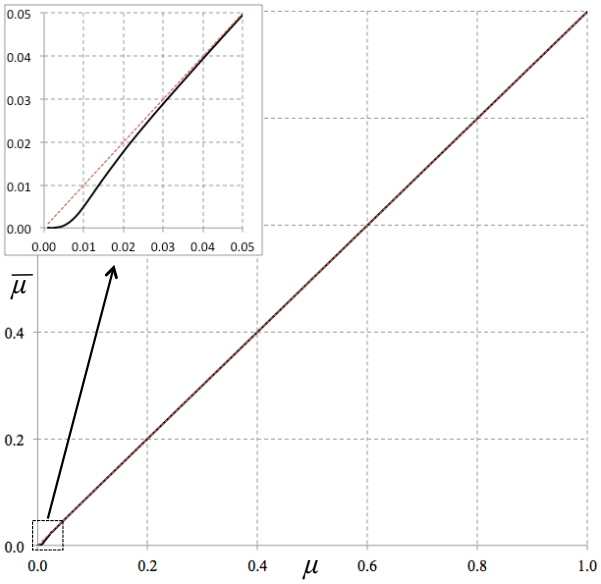

Figure 9. Stress penalization factor.

Shape sensitivities of buckling load factor eigenvalues must be computed to derive the velocity function that updates the implicit function and optimizes the structure. Our approach is to first compute element-wise derivatives of the buckling eigenvalues with respect to a change in the volume-fraction:

$$\frac{d\alpha_i}{d\mu_j} = -\frac{v_i^T \left(\partial K/\partial \mu_j + \alpha_i \, \partial K_s/\partial \mu_j\right) v_i}{v_i^T K_s v_i} - \tilde{u}_i^T \frac{\partial K}{\partial \mu_j} u, \tag{31}$$

where the sub-script j is the element number and $\tilde{u}_i$ is the solution to an adjoint problem:

$$K^T \tilde{u}_i = -\frac{\alpha_i \, v_i^T \left(\partial K_s/\partial u\right) v_i}{v_i^T K_s v_i}. \tag{32}$$

The element-wise derivatives are then interpolated at the boundary using a weighted least squares method to obtain the shape sensitivities [42]. Note that the derivative in (31) includes the part dependent on the change in the stress stiffness matrix with a change in displacement vector through the solution of the adjoint equation (32).

The level-set topology optimization method only uses first order gradient information when computing the shape sensitivities that are used to define the velocity function that updates the design (26). In general, buckling eigenvalues are nonlinear with respect to the design variables. Therefore, if too large a step is taken for an outer iteration the linear approximation of the buckling eigenvalues may result in the design becoming infeasible or less feasible. To partially alleviate this problem, we introduce a scaling factor, ζ, for the time step in the implicit function update equation (24):

$$\phi_i^{k+1} = \phi_i^k + \zeta \Delta t |\nabla \phi_i(\chi)| V_{n,i}. \tag{33}$$

DOI: 10.1002/nme

The scaling factor is dynamically chosen during optimization. Initially $\zeta = 1$, then if any buckling constraint becomes infeasible, or less feasible, it is reduced by half, to a minimum of 0.25. If three successive outer optimization iterations do not detect a constraint becoming infeasible, or less feasible, then ζ is doubled, to a maximum of 1. This strategy allows the time step to reduce when the nonlinearity of the buckling load factors disrupts convergence, but also to increase to improve convergence rate.

Our aim is to minimize structural mass, whilst ensuring all critical buckling eigenvalues are above 1.0. As the optimum solution is approached, many eigenvalues are close to 1.0. This increases the possibility of repeated eigenvalues, that are difficult to handle with gradient based optimization as they are only directionally differentiable. From a practical engineering perspective, it is often good design practice to avoid several failure modes occurring simultaneously, which is the case for a structure with several buckling eigenvalues near 1.0. These observations motivate our use of a loose definition of the linear bucking constraint. The idea is to set a range of values for which a buckling constraint is considered active by the optimizer. Here we consider any buckling eigenvalue between 1.0 and 1.2 to be active. This is implemented in the present algorithm by setting the constraint change target G_i in the velocity sub-problem (27) depending on the value of the load factor α_i:

$$G_i = \begin{cases} 1.0 - \alpha_i & \alpha_i \leq 1.0, \\ 10^{-3} & 1.0 < \alpha_i < 1.2, \\ 1.2 - \alpha_i & 1.2 \leq \alpha_i. \end{cases} \tag{34}$$

The small positive value of 10^{-3} is used to encourage the optimizer to choose a search direction that increases the load factor, which reduces the likelihood of the buckling constraint becoming violated from the nonlinearity of the buckling load factors.

6. EXAMPLE

In this section, a 3D topology optimization example for the linear buckling constrained mass minimization problem (2) is solved using the level-set method described in the previous section. The benefit of using multiple buckling modes is investigated. The efficiency of the BJCG eigenvalue solver `SPRAL_SSMFE` is also demonstrated by comparing the computation time with that of ARPACK. Other key features of the solver are also investigated, such as reusing eigenvector information.

All tests are performed on a 16-core Intel(R) Xeon(R) E5-2687W CPU. The shift value for `SPRAL_SSMFE` is computed using the eigenvalues from the previous outer iteration and (21) with $q = 1.1$. However, the number of eigenvalues to the left of the computed shift may exceed the required number. In this case, first we set the eigenvector tolerance to 1.0 to switch off error control and approximately solve the buckling eigenvalue problem for all eigenvalues to the left of the chosen shift. This provides a good estimate of the required eigenvalues and (20) is used to compute a new shift. The buckling eigenvalue problem is then solved to the desired tolerance using the new shift. ARPACK cannot guarantee that all required eigenvalues are found if a shift inside the spectrum is chosen. Therefore, a more robust strategy is adopted for ARPACK in which the shift is chosen as half the lowest eigenvalue from the previous optimization iteration. For both eigenvalue solvers the initial shift for the first iteration is chosen as 1.0.

The factorization of the stiffness matrix, used to solve the linear static (1) and adjoint (32) problems, and of the shifted matrix (14), required to solve the buckling eigenvalue problem, are again computed using `HSL_MA97` [25]. The highly optimized BLAS subroutines provided by the Intel MKL library are also utilized. The Intel compiler with flags `-O3 -openmp -fp-model precise -fp-model source` is employed.

The default convergence tolerance of machine precision is used for ARPACK while the tolerance is set to 10^{-12} for `SPRAL_SSMFE`. For the example studied here, this `SPRAL_SSMFE` tolerance provides a similar order of magnitude in the eigenvector energy norm error (see Section 3.2) as ARPACK with the default tolerance. For example, at iteration one, the errors for the first 10

DOI: 10.1002/nme

eigenvectors range between 5.2×10^{-13} to 1.1×10^{-12} for SPRAL_SSMFE and between 1.2×10^{-13} to 1.3×10^{-12} for ARPACK.

The optimization problem is shown in Figure 10. The initial structure contains eight circular holes that extend through the width of the domain. All degrees of freedom (dof) on one face are fixed and a uniformly distributed load totalling 36,000 units is applied to the top. The design domain is discretized using unit sized 8-node brick elements [15] giving a total of 144,000 elements and 153,842 nodes. Each node has three dof and, with the fixed dof removed from the system, the size of the stiffness and stress stiffness matrices is 456,768. The material properties are $E = 10^3$, Poisson's ratio of 0.3 and $\rho = 1$.

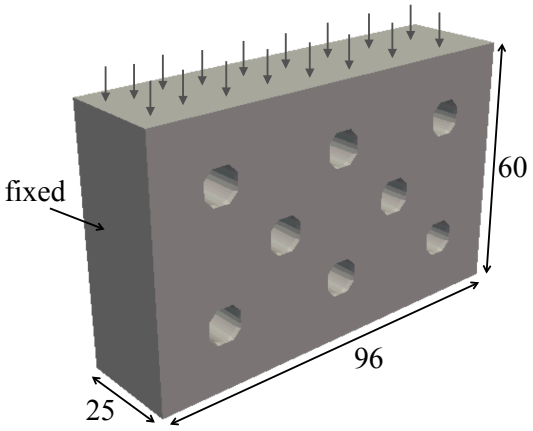

Figure 10. Topology optimization example: Initial design, loading and boundary conditions.

The buckling constrained mass minimization problem (2) is first solved using 5, 10 and 25 buckling modes. A comparison of timings for ARPACK and SPRAL_SSMFE is shown in Table VII, where the buckle solve time is defined to be the time spent solving the buckling eigenvalue problem (not including the shift-invert phase) and the outer iteration time is the total time for one complete optimization iteration. SPRAL_SSMFE shows a significant reduction of 60% to 75% in the buckle solve time compared with ARPACK. Table VII also shows that for ARPACK the buckle solve time accounts for between 68% and 77% of the total time (i.e. the most significant part), but is only around 45% of the total time when SPRAL_SSMFE is used. There are four other operations that contribute significantly to the average outer iteration time: the linear solve (1), computation and factorization of the shift-inverted matrix (16), adjoint solve (32) and velocity sub-problem solve (27). Note that the adjoint solve does not require a matrix factorization, as the stiffness matrix factorization is saved from the linear solve phase. Also, the code for computing the adjoint load vector (32) is not optimized for speed. The linear solve and shift-invert times are essentially constant, as these operations are independent of the number of modes. However, the adjoint solve and velocity sub-problem times do increase with the number of modes. The increase is more significant for the velocity sub-problem because, as the number of buckling constraints is increased, the sub-problem is more difficult to solve and thus requires more iterations and gradient computations.

Figure 11 shows the buckle solve time per outer iteration using 10 modes. It is observed that the SPRAL_SSMFE solve time is more consistent than ARPACK, for which the solve time generally increases as the optimization progresses. One contributing factor is that the separation of the buckling eigenvalues reduces as the optimization progresses. For example, the ratio of the 10^{th} to 1^{st} eigenvalue on the first iteration is 1.61, but reduces to 1.08 at the end of the optimization. We note that this does not affect the performance of SPRAL_SSMFE because the BJCG algorithm is *cluster robust*, i.e. its convergence is not adversely affected by the eigenvalue clustering [36].

Convergence for different numbers of modes using SPRAL_SSMFE is shown in Figure 12. For this example, using 5 bucking modes is insufficient to obtain a feasible solution because of mode switching throughout the optimization. Feasible solutions are obtained using 10 or more modes and

DOI: 10.1002/nme

Table VII. Average (Ave.) times for SPRAL_SSMFE and ARPACK.

SPRAL_SSMFE No. modes	Ave. time (s) linear solve	shift-invert	buckle solve	adjoint solve	sub-problem	outer iteration
5	17.4	19.3	44.8	12.4	0.5	96.8
10	17.3	19.2	49.5	12.2	4.0	108.8
25	17.4	19.8	82.7	28.3	26.3	183.8
ARPACK						
5	17.3	18.8	117.2	12.6	0.8	171.3
10	17.3	20.4	197.6	11.1	3.0	256.0
25	17.7	21.2	241.2	28.2	22.1	337.8

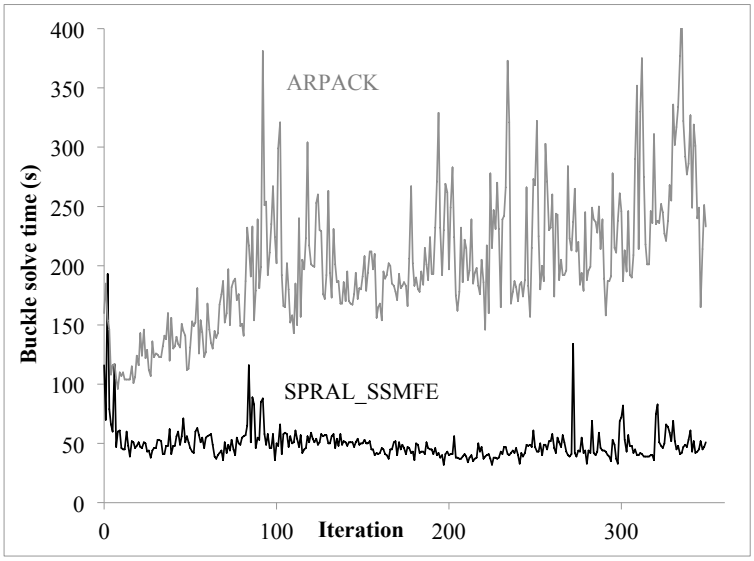

Figure 11. Buckle solve time using 10 modes.

these are all very similar (within 1% of the objective function). However, significant oscillations in the lowest buckling load factor are observed for fewer than 25 modes, especially near the optimum, which can increase the number of required iterations. This is highlighted by 20 modes taking 128 more iterations to converge than 25 modes. Oscillations still occur when using 25 modes, but are less frequent and of smaller magnitude, making convergence more reliable. Looking at the convergence of the objective function, using more modes slows the rate of convergence because the mass reduces more slowly. This is because a greater number of modes introduces more constraints into the problem, and this leads to the optimizer taking smaller steps to satisfy these constraints at each iteration. The problem was also solved using 50 modes and the oscillations in the lowest buckling load factor were comparable to using 25 modes and the convergence was similar. Thus, for this example, it appears that 25 modes is sufficient to prevent mode switching disrupting convergence, particularly near the optimum.

During optimization, the buckling eigenvalues become very close. However, the minimum difference between eigenvalues is 10^{-5} and therefore we did not observe any difficulties with repeated eigenvalues in the present study and no treatment was necessary. However, if a repeated buckling eigenvalue occurs, it may be possible to adapt the approach used by Xia et al. [18] to obtain an ascent direction for the repeated eigenvalue.

The solution obtained using SPRAL_SSMFE with 10 modes is shown in Figure 13; the solutions obtained using more modes are very similar. The structure is composed of two main diagonal struts that support the uniform loading. These struts have I-beam cross sections that the optimizer has

DOI: 10.1002/nme

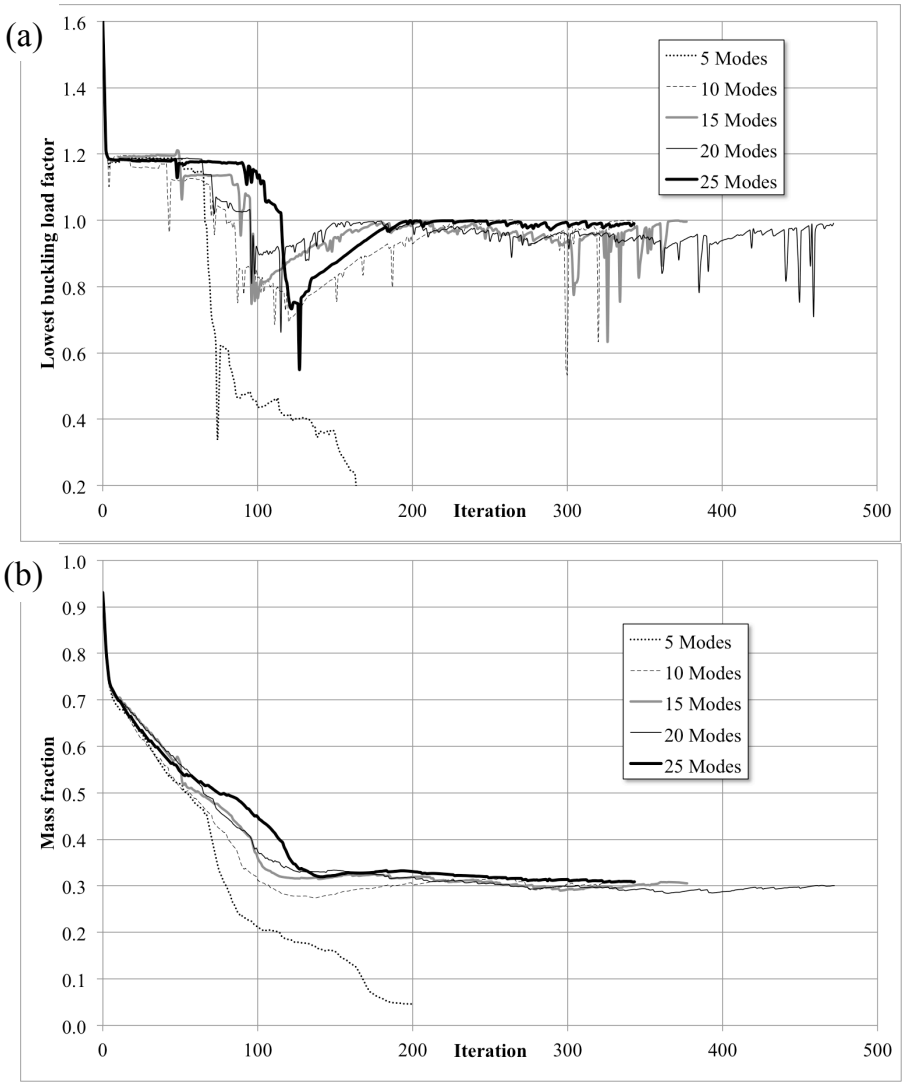

Figure 12. Topology optimization convergence: a) buckling load factor, b) mass fraction.

orientated to maximise bending stiffness in the direction of buckling. The first 10 buckling modes of the optimum solution are shown in Figure 14. The first 10 modes contain both global and local buckling modes. The modes 1, 2, 3 and 6 are global buckling modes, affecting large areas of the structure. The remaining modes buckle locally over a small area of the structure.

These local modes were investigated to ensure that they are not artificial modes caused by the numerical approximation of stiffness and stress in intermediate volume-fraction elements. The detailed results of the investigation are omitted for brevity. The results show that the buckling eigenvectors for the local modes significantly deform solid elements ($\mu = 1$) and that similar local modes were obtained when using a more conservative penalization scheme for stress. Therefore, we conclude that the local modes shown in Figure 14 are real physical modes that should be included in the optimization.

We now investigate two features of `SPRAL_SSMFE` that can be used to further reduce the buckle solve time: the reuse of eigenvectors and employing a relaxed eigenvector convergence tolerance. In the following investigations, 25 modes are used. When eigenvectors from the previous outer

DOI: 10.1002/nme

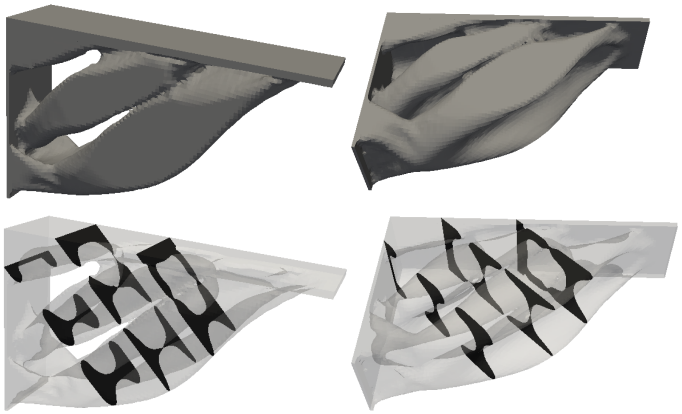

Figure 13. Topology optimization solution.

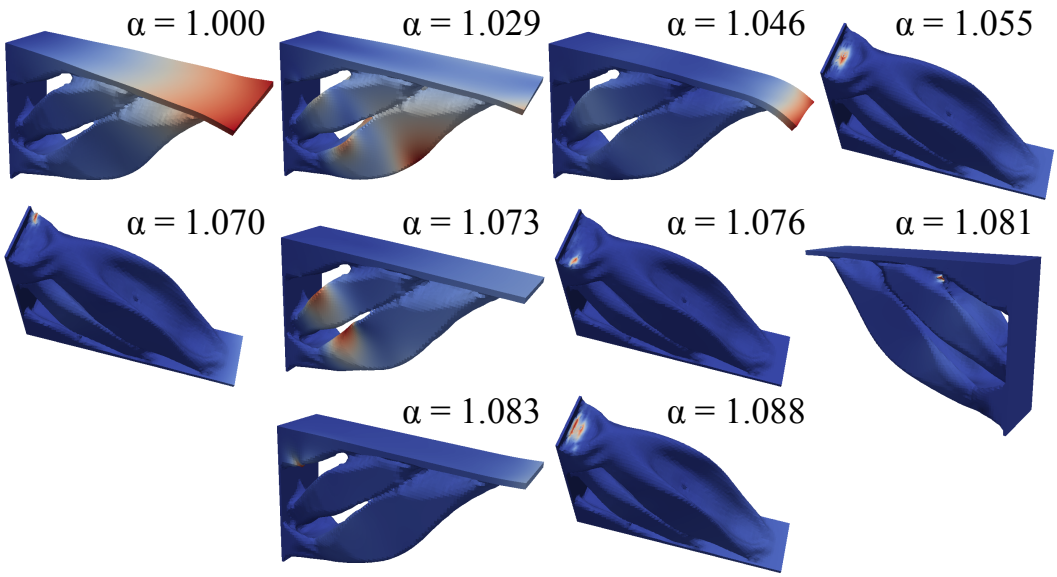

Figure 14. Buckling mode shapes of the solution. Contour colours indicate the relative magnitude of the eigenvector, with blue being zero and red being maximum.

iteration are used as the initial guess for the buckling eigenvalue problem, the reduction in average buckle solve time reduces from 83 to 70 seconds, a saving of more than 15%.

The results of using different convergence tolerances are summarised in Table VIII. It is clear that using a relaxed tolerance gives a significant reduction in the buckle solve time, although the accuracy of the eigenvectors also reduces. The eigenvectors are used to compute gradients of the buckling load factors, (31) and (32). Thus, inaccurate eigenvectors can lead to errors in the gradients, which can slow down or prevent convergence. This effect is observed for all tolerance values greater than 10^{-12}, as the optimizer fails to find a feasible design as the optimum is approached. As suggested in Section 4, a variable tolerance scheme can be considered, where a larger value is used in the early outer iterations for efficiency and then reduced to increase the eigenvector accuracy as the optimum is approached. To demonstrate this, a simple strategy is used where a tolerance of 10^{-4} is used for 200 outer iterations and then reduced to 10^{-12}. Using this strategy, a feasible solution is found in 349 iterations and the overall average buckle solve time is 30% lower than using a tolerance of 10^{-12} throughout, Table VIII.

DOI: 10.1002/nme

The error in the gradients arising from inaccurate displacement vectors was studied by Amir et al. [43] in the context of the termination criterion of iterative solvers. Large inaccuracies can be tolerated by using consistent sensitivities, where the approximation error is included in the sensitivity calculation. Unfortunately, this involves performing an adjoint solve for all expressions used in the iterative solve, which can add significant computational cost in practice. The suggested alternative is to use termination criteria that are correlated with the function value and sensitivity, instead of the usual practice of defining the termination criterion in terms of the residual. This ensures that the approximation is sufficiently accurate for optimization, although not necessarily accurate in terms of the residual. Obviously, any scheme that uses an adaptive tolerance would have to ensure a sufficient level of accuracy for convergence.

Table VIII. Times for different eigenvector error tolerances for 25 modes and reusing eigenvectors.

tolerance	Ave. buckle solve time (s)
10^{-4}	33.3
10^{-6}	40.9
10^{-9}	52.2
10^{-12}	69.7
variable	48.8

7. CONCLUSIONS

In this paper, the linear buckling constrained mass minimizaton problem was studied in the context of topology optimization. A method for including linear buckling constraints in level-set based topology optimization was introduced. The buckling constraints were handled by defining the velocity function as a weighted sum of the shape sensitivities and solving an optimization sub-problem to find a velocity function that reduced the objective whilst maintaining constraint feasibility.

We demonstrated that the switching of the critical buckling eigenmode during optimization can result in slow convergence, or a failure to find a feasible solution. To remedy this we propose using more buckling eigenmodes during optimization to provide the optimizer with more information and hence prevent mode switching from disrupting convergence. In the context of topology optimization, this can lead to a large number of eigenmodes of ill-conditioned matrices being computed at each iteration.

The reliable and efficient computation of a large number eigenmodes presents a significant computational challenge. The BJCG eigensolver method has several features that can be exploited to address this challenge. First, eigenvalue and eigenvector information from the previous optimization iteration can be used to speed up the convergence of the BJCG method in the current iteration. The eigenvalues can be used to estimate the optimal shift value and previous eigenvectors can be used as the initial guess for the new eigenvectors. Also, a larger eigenvector convergence tolerance can be used in the early stages of optimization to reduce the overall computation time. However, in our example it was found that a smaller tolerance was required near the optimum to increase gradient accuracy and achieve convergence. This suggests that an adaptive tolerance scheme may be an efficient strategy, where the tolerance is reduced from a large starting value during optimization.

A 3D optimization problem was used to demonstrate the effectiveness of the level-set topology optimization method and the efficiency of the BJCG eigenvalue solver, implemented in the package `SPRAL_SSMFE`. The level-set method was able to obtain feasible designs for the mass minimization problem with as many as 50 linear buckling constraints. Buckling eigenvalue solution times using `SPRAL_SSMFE` were compared with APRACK. The results show that a significant reduction in the buckling solve time can be achieved by using `SPRAL_SSMFE`, especially when eigenvalue and eigenvector information is reused and an adaptive eigenvector tolerance scheme employed. For a

DOI: 10.1002/nme

very large number of modes, SPRAL_SSMFE can also employ a multiple shift strategy to further reduce overall computation time.

ACKNOWLEDGEMENTS

The authors acknowledge the support from Engineering and Physical Sciences Research Council, fellowship grant EP/M002322/1 and grant EP/J010553/1. We also thank Dr. Bret Stanford (NASA Langley) for useful discussions.

REFERENCES

1. Haftka RT, Gürdal Z. *Elements of Structural Optimization*. Third Revised and Expanded edn., Kluwer Academic Publishers: Dordrecht, 1992.
2. Spillers WR, MacBain KM. *Structural Optimization*. Springer: Dordrecht, 2009.
3. Deaton JD, Grandhi RV. A survey of structural and multidisciplinary continuum topology optimization: post 2000. *Structural and Multidisciplinary Optimization* 2014; **49**(1):1–38.
4. Sigmund O, Maute K. Topology optimization approaches a comparative review. *Structural and Multidisciplinary Optimization* 2013; **48**(6):1031–1055.
5. Rozvany GIN. Difficulties in truss topology optimization with stress, local buckling and system stability constraints. *Structural Optimization* 1996; **11**(3-4):213–217.
6. Zhou M. Difficulties in truss topology optimization with stress and local buckling constraints. *Structural Optimization* 1996; **11**(2):134–136.
7. Kocvara M. On the modelling and solving of the truss design problem with global stability constraints. *Structural and Multidisciplinary Optimization* 2002; **23**(3):189–203.
8. Rahmatalla S, Swan CC. Continuum topology optimization of buckling-sensitive structures. *AIAA Journal* 2003; **41**(6):1180–1189.
9. Lund E. Buckling topology optimization of laminated multi-material composite shell structures. *Composite Structures* 2009; **91**(2):158–167.
10. Rong JH, Xie YM, Yang XY. An improved method for evolutionary structural optimisation against buckling. *Computers and Structures* 2001; **79**(3):253–263.
11. Neves MM, Rodrigues H, Guedes JM. Generalized topology design of structures with a buckling load criterion. *Structural Optimization* 1995; **10**(2):71–78.
12. Lindgaard E, Dahl J. On compliance and buckling objective functions in topology optimization of snap-through problems. *Structural and Multidisciplinary Optimization* 2013; **47**(3):409–421.
13. Allaire G, Jouve F, Toader A. Structural optimization using sensitivity analysis and a level-set method. *Journal of Computational Physics* 2004; **194**(1):363–393.
14. Wang MY, Wang X, Guo D. A level set method for structural topology optimization. *Computer Methods in Applied Mechanics and Engineering* 2003; **192**(1-2):227–246.
15. Cook RD, Malkus DS, Plesha ME, Witt RJ. *Concepts and applications of finite element analysis*. 4th edn., John Wiley & Sons, Inc.: United States, 2002.
16. Bruyneel M, Colson B, Remouchamps A. Discussion on some convergence problems in buckling optimisation. *Structural and Multidisciplinary Optimization* 2008; **35**(2):181–186.
17. Seyranian AP, Lund E, Olhoff N. Multiple-eigenvalues in structural optimization problems. *Structural Optimization* 1994; **8**(4):207–227.
18. Xia Q, Shi T, Wang MY. A level set based shape and topology optimization method for maximizing the simple or repeated first eigenvalue of structure vibration. *Structural and Multidisciplinary Optimization* 2011; **43**(4):473–485.
19. Dunning PD, Stanford BK, Kim HA. Level-set topology optimization with aeroelastic constraints. *56th AIAA/ASCE/AHS/ASC Structures, Structural Dynamics, and Materials Conference*. AIAA SciTech, American Institute of Aeronautics and Astronautics, 2015; 1–19.
20. Wang S, de Sturler E, Paulino GH. Large-scale topology optimization using preconditioned krylov subspace methods with recycling. *International Journal for Numerical Methods in Engineering* 2007; **69**(12):2441–2468.
21. Makhija D, Maute K. Numerical instabilities in level set topology optimization with the extended finite element method. *Structural and Multidisciplinary Optimization* 2014; **49**(2):185–197.
22. Lehoucq RB, Sorensen DC, Yang C. *ARPACK Users Guide: Solution of Large-Scale Eigenvalue Problems with Implicitly Restarted Arnoldi Methods*. SIAM: Philadelphia, 1998.
23. MATLAB. *version 8.2.0.701 (R2013b)*. The MathWorks Inc.: Natick, Massachusetts, 2013.
24. The NAG library. The Numerical Algorithms Group (NAG), Oxford, United Kingdom www.nag.com.
25. Hogg JD, Scott JA. HSL_MA97: a bit-compatible multifrontal code for sparse symmetric systems. *Technical Report RAL-TR-2011-024*, Rutherford Appleton Laboratory 2011.
26. HSL. A collection of Fortran codes for large-scale scientific computation 2013. http://www.hsl.rl.ac.uk.
27. SPRAL. Sparse Parallel Robust Algorithms Library 2015. http://www.numerical.rl.ac.uk/spral/.
28. O'Leary DP. The block conjugate gradient algorithm and related methods. *Linear Algebra and its Applications* 1980; **29**:293–322.
29. Arbenz P, Hetmaniuk UL, Lehoucq RB, Tuminaro RS. A comparison of eigensolvers for large-scale 3D modal analysis using AMG-preconditioned iterative methods. *International Journal for Numerical Methods in Engineering* 2005; **64**(2):204–236.
30. Knyazev AV. Toward the optimal preconditioned eigensolver: Locally optimal block preconditioned conjugate gradient method. *SIAM Journal on Scientific Computing* 2001; **23**(2):517–541.

DOI: 10.1002/nme

31. Kraft D. A software package for sequential quadratic programming. *Technical Report DFVLR-FB 88-28*, Institut für Dynamik der Flugsysteme, Oberpfaffenhofen 1988.
32. Kraft D. TOMP - Fortran modules for optimal control calculations. *ACM Transactions on Mathematical Software* 1994; **20**(3):262–281.
33. The NLopt nonlinear optimization package 2014. `http://ab-initio.mit.edu/nlopt`.
34. van Zyl LH. Use of eigenvectors in the solution of the flutter equation. *Journal of Aircraft* 1993; **30**(4):553–554.
35. Ovtchinnikov EE. Jacobi correction equation, line search and conjugate gradients in Hermitian eigenvalue computation I: Computing an extreme eigenvalue. *SIAM Journal on Numerical Analysis* 2008; **46**(5):2567–2592.
36. Ovtchinnikov EE. Jacobi correction equation, line search and conjugate gradients in Hermitian eigenvalue computation II: Computing several extreme eigenvalues. *SIAM Journal on Numerical Analysis* 2008; **46**(5):2593–2619.
37. Ovtchinnikov EE. Computing several eigenpairs of Hermitian problems by conjugate gradient iterations. *Journal of Computational Physics* 2008; **227**(22):9477–9497.
38. Takahashi I. A note on the conjugate gradient method. *Information Processing in Japan* 1965; **5**:45–49.
39. Saad Y. Projection methods for solving large sparse eigenvalue problems. *Lecture Notes in Mathematics* 1983; **973**:121–144.
40. Dunning PD, Stanford BK, Kim HA. Coupled aerostructural topology optimization using a level set method for 3D aircraft wings. *Structural and Multidisciplinary Optimization* 2015; **51**(5):1113–1132.
41. Dunning PD, Kim HA. Introducing the sequential linear programming level-set method for topology optimization. *Structural and Multidisciplinary Optimization* 2015; **51**(3):631–643.
42. Dunning PD, Kim HA, Mullineux G. Investigation and improvement of sensitivity computation using the area-fraction weighted fixed grid FEM and structural optimization. *Finite Elements in Analysis and Design* 2011; **47**(8):933–941.
43. Amir O, Stolpe M, Sigmund O. Efficient use of iterative solvers in nested topology optimization. *Structural and Multidisciplinary Optimization* 2010; **42**(1):55–72.

DOI: 10.1002/nme