INTERNATIONAL JOURNAL FOR NUMERICAL METHODS IN ENGINEERING
Int. J. Numer. Meth. Engng 2007; **69**:2441–2468
Published online 5 September 2006 in Wiley InterScience (www.interscience.wiley.com). DOI: 10.1002/nme.1798

Large-scale topology optimization using preconditioned Krylov subspace methods with recycling

Shun Wang[1,‡], Eric de Sturler[2,§] and Glaucio H. Paulino[3,*,†]

[1]*Department of Computer Science, University of Illinois at Urbana-Champaign, Urbana, IL 61801, U.S.A.*
[2]*Department of Mathematics, Virginia Tech, Blacksburg, VA 24061, U.S.A.*
[3]*Department of Civil and Environmental Engineering, University of Illinois at Urbana-Champaign, Urbana, IL 61801, U.S.A.*

SUMMARY

The computational bottleneck of topology optimization is the solution of a large number of linear systems arising in the finite element analysis. We propose fast iterative solvers for large three-dimensional topology optimization problems to address this problem. Since the linear systems in the sequence of optimization steps change slowly from one step to the next, we can significantly reduce the number of iterations and the runtime of the linear solver by recycling selected search spaces from previous linear systems. In addition, we introduce a MINRES (minimum residual method) version with recycling (and a short-term recurrence) to make recycling more efficient for symmetric problems. Furthermore, we discuss preconditioning to ensure fast convergence. We show that a proper rescaling of the linear systems reduces the huge condition numbers that typically occur in topology optimization to roughly those arising for a problem with constant density. We demonstrate the effectiveness of our solvers by solving a topology optimization problem with more than a million unknowns on a fast PC. Copyright © 2006 John Wiley & Sons, Ltd.

Received 5 January 2006; Revised 3 May 2006; Accepted 5 May 2006

KEY WORDS: topology optimization; three-dimensional analysis; iterative methods; Krylov subspace recycling; preconditioning; large-scale computation

1. INTRODUCTION

The goal of topology optimization is to find a material distribution in terms of design variables such that a given objective function, e.g. compliance, is minimized subject to certain constraints. We give

*Correspondence to: Glaucio H. Paulino, Department of Civil and Environmental Engineering, University of Illinois at Urbana-Champaign, Urbana, IL 61801, U.S.A.
†E-mail: paulino@uiuc.edu
‡E-mail: shunwang@uiuc.edu
§E-mail: sturler@cs.uiuc.edu

Contract/grant sponsor: National Science Foundation; contract/grant number: DMR-0325939

a brief introduction to topology optimization in the next section. To make topology optimization a truly effective tool in the design of large structures and complex materials, we must be able to use large three-dimensional models. Most work on topology optimization for continuum structures has emphasized developing new formulations and applications, designing suitable elements, studying existence and uniqueness issues, and solving (modest size) problems. However, the computational aspect of large-scale topology optimization, specifically the high cost of solving many large linear systems, has not received much attention. This is the focus of this paper.

The finite element analysis step in topology optimization requires the solution of a long sequence of linear systems of the type

$$\boldsymbol{K}(\boldsymbol{\rho}^{(i)})\boldsymbol{u}^{(i)} = \boldsymbol{f} \tag{1}$$

where $\boldsymbol{K}$ is the stiffness matrix as a function of the density distribution $\boldsymbol{\rho}$ at the ith optimization step, $\boldsymbol{f}$ is the load vector, and $\boldsymbol{u}$ is the displacement vector. Currently, direct solvers are most commonly used, because of the very large condition numbers arising in topology optimization. Unfortunately, direct solvers cannot effectively handle large 3D problems, since their large storage and computational requirements make them prohibitively expensive. Iterative solvers have low storage requirements and the computational cost per iteration is small. Therefore, as long as convergence is reasonably fast, we can solve very large problems.

Iterative solvers offer a number of additional advantages compared with direct solvers. First, we do not need to solve very accurately in the early phase of the topology optimization process. Second, iterative solvers are easy to parallelize, which is important for very large problems. For instance, parallelization of topology optimization was studied in References [1–3]. Third, iterative solvers can use solutions from previous systems as starting guesses, which leads to smaller initial residuals. Last, for a sequence of linear systems that change slowly, we can reduce the total number of iterations by recycling subspaces of earlier search spaces [4, 5].

In topology optimization, the change in the design variables becomes small after the first few optimization steps. Therefore, the change in the system matrix $\boldsymbol{K}(\boldsymbol{\rho})$ from one optimization step to the next is also small, and the Krylov subspace recycling methods introduced in Reference [4] are likely to be effective. We give more background on recycling in Section 3. Some topology optimization problems lead to a non-linear system in each optimization step [6]. If we use a Newton or quasi-Newton method for the non-linear system, the (approximate) Jacobians often change sufficiently slowly, and we can further exploit recycling; see, for example, Reference [5].

In most structural problems, the matrices are symmetric but not necessarily positive definite. For example, in vibration problems symmetric indefinite matrices arise [7]. For such matrices, MINRES (minimum residual method) [8] is the method of choice. Therefore, we focus on MINRES in the present paper. We extend the idea of subspace recycling to MINRES and make it more efficient by exploiting symmetry and short-term recurrences. We discuss the recycling MINRES in detail in Section 4. For topology optimization problems with non-symmetric matrices, e.g. Reference [9], the Krylov subspace recycling methods from [4], outlined briefly in Section 3, can be used.

To achieve fast convergence we do need to use preconditioning. As the material distribution in a structure is being optimized, some elements become nearly void (we set a small positive lower bound on the densities to avoid singularity). This makes the linear system very ill-conditioned. First, we show that the ill-conditioning is largely a problem of poor scaling. We reduce the condition number by 6 orders of magnitude by rescaling the system matrices with two diagonal matrices. Since diagonal scaling does not introduce numerical errors, this also mitigates the serious potential accuracy problems of ill-conditioning. Next, we combine the rescaling with other preconditioners

DOI: 10.1002/nme

to make the condition numbers of the linear systems even smaller. We discuss preconditioning in Section 5.

In Section 6, we give some implementation details of our methods. In Section 7, we analyse our methods for a model problem, and we present the numerical results to demonstrate the significant improvements our solvers achieve. In the last section, we provide the conclusions.

2. TOPOLOGY OPTIMIZATION

Topology optimization is a powerful structural optimization method that combines a numerical solution method, usually the finite element method (FEM), with an optimization algorithm to find the optimal material distribution inside a given domain [10–15]. In designing the topology of a structure we determine which points of space should be material and which points should be void (i.e. no material). However, it is well known that an optimum result of topology optimization consists in a structure with intermediate (or composite) material. So, continuous values between 0 and 1 replace the discrete 0/1 numbers to represent the relative densities of the elements, while some form of penalization is used to steer the solution back to discrete 0/1 values [16]. The objective function is the compliance and there is a volume constraint. This is the basic set-up of a topology optimization problem. We specify the problem mathematically as follows:

$$\begin{aligned} \min_{\boldsymbol{\rho},\boldsymbol{u}} \quad & c(\boldsymbol{\rho},\boldsymbol{u}) = \boldsymbol{u}^{\mathrm{T}}\boldsymbol{K}(\boldsymbol{\rho})\boldsymbol{u} \\ \text{s.t.:} \quad & \boldsymbol{K}(\boldsymbol{\rho})\boldsymbol{u} = \boldsymbol{f} \\ & 0 < \rho_0 \leqslant \rho_e \leqslant 1 \quad e = 1, 2, \ldots, n_e \\ & \int_\Omega \rho \, \mathrm{d}\Omega \leqslant V \end{aligned} \tag{2}$$

where c is the compliance, $\boldsymbol{K}(\boldsymbol{\rho})$ is the stiffness matrix as a function of the density distribution $\boldsymbol{\rho}$, $\boldsymbol{u}$ and $\boldsymbol{f}$ are the displacement vector and load vector, ρ_0 is a chosen, small, positive lower bound for the density to avoid singularity of the stiffness matrix, and V is the total volume in use. The solid isotropic material with penalization (SIMP) method [16, 17] uses one design variable to represent the density in each element, while the recent method of continuous approximation of material distribution (CAMD) [18–21] uses multiple variables per element. Since the focus of this paper is the linear solver in the finite element analysis (FEA), we use the SIMP method as a simple set-up.

The basic scheme of topology optimization is described in Figure 1. First, we set up the geometry and the loading, and initialize the density distribution $\boldsymbol{\rho}$. Then, we start the optimization loop. We need a linear solver to solve the equilibrium equations $\boldsymbol{K}\boldsymbol{u} = \boldsymbol{f}$ in the finite element analysis. In the sensitivity analysis, we compute the derivatives of the objective function $\partial c/\partial \rho_e$. After this, we can apply an optional low-pass filter to remedy the checkerboard problem [22]. The next step is the kernel of the optimization. There are various optimization algorithms that can be used for topology optimization. For instance, optimality criteria (OC) is a simple approach based on a set of intuitive criteria [23], while the method of moving asymptotes (MMA) is a mathematical programming algorithm which is more robust and well established in theory [24]. Since this paper deals mainly with the FEA in topology optimization, the choice of the optimization method is less relevant

DOI: 10.1002/nme

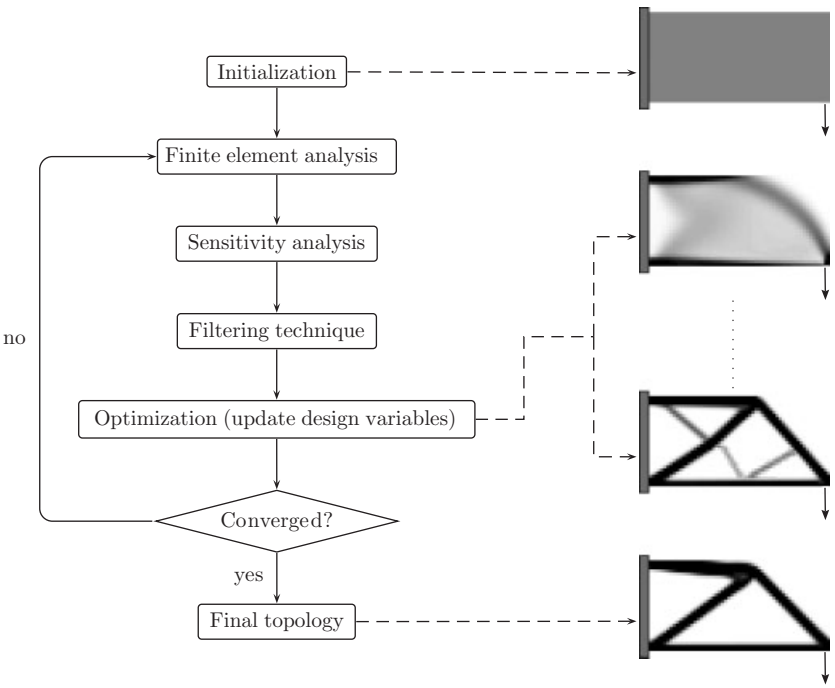

Figure 1. The general flow of computations for topology design.

for our discussion. Therefore, we choose the OC method for its simplicity. However, our Krylov subspace recycling method and preconditioning techniques are general and can be used with other optimization methods.

3. KRYLOV SUBSPACE RECYCLING

Consider the linear system $\boldsymbol{K}\boldsymbol{u}=\boldsymbol{f}$ and an initial guess $\boldsymbol{u}_0$. A Krylov subspace method, such as the generalized minimum residual method (GMRES) [25], builds the Krylov subspace, span$\{\boldsymbol{r}_0, \boldsymbol{K}\boldsymbol{r}_0, \boldsymbol{K}^2\boldsymbol{r}_0, \ldots, \boldsymbol{K}^{m-1}\boldsymbol{r}_0\}$, where $\boldsymbol{r}_0=\boldsymbol{f}-\boldsymbol{K}\boldsymbol{u}_0$, and computes the optimal solution over that subspace. We use the Arnoldi recurrence [26] to obtain an orthonormal basis of the Krylov subspace:

$$\begin{aligned} \boldsymbol{v}_1 &= \boldsymbol{r}_0/\|\boldsymbol{r}_0\|_2 \\ h_{i,i+1}\boldsymbol{v}_{i+1} &= \boldsymbol{K}\boldsymbol{v}_i - h_{i,i}\boldsymbol{v}_i - h_{i-1,i}\boldsymbol{v}_{i-1} - \cdots - h_{1,i}\boldsymbol{v}_1 \end{aligned} \tag{3}$$

which in matrix form is written as

$$\boldsymbol{K}\boldsymbol{V}_m = \boldsymbol{V}_{m+1}\underline{\boldsymbol{H}}_m \tag{4}$$

where the columns of $\boldsymbol{V}_m$ are $\boldsymbol{v}_1, \ldots, \boldsymbol{v}_m$; the columns of $\boldsymbol{V}_{m+1}$ are $\boldsymbol{v}_1, \ldots, \boldsymbol{v}_{m+1}$; and $\underline{\boldsymbol{H}}_m$ is an $(m+1)\times m$ upper Hessenberg matrix with coefficients $\{h_{ij}\}$.

DOI: 10.1002/nme

For a symmetric matrix, we have $\boldsymbol{K}=\boldsymbol{S}\boldsymbol{\Lambda}\boldsymbol{S}^{\mathrm{T}}$, where $\boldsymbol{S}$ is an orthogonal matrix, and $\boldsymbol{\Lambda}$ is a diagonal matrix whose coefficients are the eigenvalues of $\boldsymbol{K}$. The convergence rate of GMRES for a symmetric problem is bounded by [27, p. 206]

$$\frac{\|\boldsymbol{r}_m\|_2}{\|\boldsymbol{r}_0\|_2} \leqslant \min_{p_m\in\Pi_m^{(0)}} \max_{\lambda\in\lambda(\boldsymbol{K})} |p_m(\lambda)| \tag{5}$$

where $\Pi_m^{(0)}$ is the set of polynomials p_m of degree m such that $p_m(0)=1$, and $\lambda(\boldsymbol{K})$ is the set of eigenvalues of $\boldsymbol{K}$. So, the bound depends on the spectrum of the matrix. Therefore, if we remove an appropriate subset of the eigenvalues, M,

$$\min_{p_m\in\Pi_m^{(0)}} \max_{\lambda\in(\lambda(\boldsymbol{K})\setminus M)} |p_m(\lambda)|$$

can be significantly smaller than (5), and then the rate of convergence will be greatly improved. This is the motivation for recycling approximate invariant subspaces; other subspaces of the Krylov space may also be effective as a recycle space [4, 5]. Given the normalization condition, $p_m(0)=1$, it is often effective to remove the eigenvalues close to the origin. This filtering of eigenvalues is achieved by including the corresponding (approximate) invariant subspace in the Krylov subspace over which we minimize. We typically recycle harmonic Ritz vectors with respect to the Krylov subspace to approximate an invariant subspace [4].

When solving the next linear system, $\boldsymbol{K}(\boldsymbol{\rho}^{(i+1)})\boldsymbol{u}^{(i+1)}=\boldsymbol{f}$, we include the recycle space as follows. We choose a basis for the recycle space to be the columns of a matrix $\boldsymbol{U}$, such that $\boldsymbol{C}=\boldsymbol{K}\boldsymbol{U}$ and $\boldsymbol{C}^{\mathrm{T}}\boldsymbol{C}=\boldsymbol{I}$. In addition, we adapt the Arnoldi process to make each new Krylov vector $\boldsymbol{v}$ orthogonal to range($\boldsymbol{C}$). This leads to the following recurrence:

$$\begin{aligned}(\boldsymbol{I}-\boldsymbol{C}\boldsymbol{C}^{\mathrm{T}})\boldsymbol{K}\boldsymbol{V}_m=\boldsymbol{V}_{m+1}\underline{\boldsymbol{H}}_m \Leftrightarrow\\ \boldsymbol{K}\boldsymbol{V}_m=\boldsymbol{C}\boldsymbol{C}^{\mathrm{T}}\boldsymbol{K}\boldsymbol{V}_m+\boldsymbol{V}_{m+1}\underline{\boldsymbol{H}}_m\end{aligned} \tag{6}$$

where $\underline{\boldsymbol{H}}_m$ is still an $(m+1)\times m$ upper Hessenberg matrix. Next, we compute the vector $\boldsymbol{\varepsilon}_m=\boldsymbol{U}\boldsymbol{z}_m+\boldsymbol{V}_m\boldsymbol{y}_m$, such that $\boldsymbol{u}_m=\boldsymbol{u}_0+\boldsymbol{\varepsilon}_m$ minimizes $\|\boldsymbol{r}_m\|_2$. This gives

$$\begin{aligned}\|\boldsymbol{r}_m\|_2&=\left\|\boldsymbol{r}_0-\boldsymbol{K}[\boldsymbol{U}\ \ \boldsymbol{V}_m]\begin{pmatrix}\boldsymbol{z}_m\\ \boldsymbol{y}_m\end{pmatrix}\right\|_2\\ &=\left\|[\boldsymbol{C}\ \ \boldsymbol{V}_{m+1}]\left(\begin{pmatrix}\boldsymbol{C}^{\mathrm{T}}\boldsymbol{r}_0\\ \beta\boldsymbol{e}_1\end{pmatrix}-\begin{bmatrix}\boldsymbol{I} & \boldsymbol{B}_m\\ \boldsymbol{0} & \underline{\boldsymbol{H}}_m\end{bmatrix}\begin{pmatrix}\boldsymbol{z}_m\\ \boldsymbol{y}_m\end{pmatrix}\right)\right\|_2\\ &=\left\|\begin{pmatrix}\boldsymbol{C}^{\mathrm{T}}\boldsymbol{r}_0\\ \beta\boldsymbol{e}_1\end{pmatrix}-\begin{bmatrix}\boldsymbol{I} & \boldsymbol{B}_m\\ 0 & \underline{\boldsymbol{H}}_m\end{bmatrix}\begin{pmatrix}\boldsymbol{z}_m\\ \boldsymbol{y}_m\end{pmatrix}\right\|_2\end{aligned} \tag{7}$$

where $\beta=\|(\boldsymbol{I}-\boldsymbol{C}\boldsymbol{C}^{\mathrm{T}})\boldsymbol{r}_0\|_2$ and $\boldsymbol{B}_m=\boldsymbol{C}^{\mathrm{T}}\boldsymbol{K}\boldsymbol{V}_m$. This least squares problem can be solved using the QR decomposition of $\underline{\boldsymbol{H}}_m$. This approach derives from the GCRO method [28] and is also used in the GCRODR method and GCROT with recycling [4, 5, 29].

DOI: 10.1002/nme

An important issue for GMRES is that it relies (for general matrices) on a complete orthogonalization of the Krylov subspace. Therefore, as the Krylov subspace expands, the memory needed for the orthogonal basis vectors and the computational time for orthogonalization increase. As a result, normally restarting is required for GMRES, and we call the solution steps between two restarts a *cycle*. To mitigate the reduced convergence rate due to the loss of orthogonality caused by restarting, we use the recycle space immediately in the next cycle for the same system.

4. RECYCLING MINRES

While the recycling methods like GCRODR and GCROT, which are based on GMRES and GCRO, solve systems with general matrices, in most topology optimization problems the system matrices are symmetric. In most cases they are also positive definite. However, for some applications, e.g. topology design with dynamic vibrations, they can be indefinite [7]. So, in general, the MINRES method [8] is the most suitable iterative solver for topology optimization problems. In this section, we study recycling in the MINRES method.

Both MINRES and GMRES minimize the two-norm of the residual over the Krylov subspace. The difference is that MINRES utilizes the symmetry of the matrix, and the resulting Lanczos three-term recurrence leads to significant reductions in memory requirements and computational cost.

We can use the matrices $\boldsymbol{U}$ and $\boldsymbol{C}$ defining the recycle space, obtained from solving previous linear systems, in the same way as in GCRODR. This leads to the same recurrence as in (6). However, the symmetry of $\boldsymbol{K}$ implies the symmetry of H_m, the leading $m \times m$ submatrix of $\underline{\boldsymbol{H}}_m$.[¶] Since $\underline{\boldsymbol{H}}_m$ is also an upper Hessenberg matrix, this gives a tridiagonal $\underline{\boldsymbol{H}}_m$, which we will denote as $\underline{\boldsymbol{T}}_m$ from now on. So, including the recycle space into the Krylov subspace does not affect the Lanczos recurrence of MINRES. Now we explain how we adapt the MINRES method as described in Reference [30, p. 84–86] or [31, p. 41–44] to include the recycle space. Similarly as in GCRODR, we need to compute the vector $\boldsymbol{\varepsilon}_m = \boldsymbol{U}\boldsymbol{z}_m + \boldsymbol{V}_m\boldsymbol{y}_m$, such that $\boldsymbol{u}_m = \boldsymbol{u}_0 + \boldsymbol{\varepsilon}_m$ minimizes $\|\boldsymbol{r}_m\|_2$. For symmetric $\boldsymbol{K}$, (7) becomes

$$\|\boldsymbol{r}_m\|_2 = \left\| \begin{pmatrix} \boldsymbol{C}^{\mathrm{T}}\boldsymbol{r}_0 \\ \beta\boldsymbol{e}_1 \end{pmatrix} - \begin{bmatrix} \boldsymbol{I} & \boldsymbol{B}_m \\ 0 & \underline{\boldsymbol{T}}_m \end{bmatrix} \begin{pmatrix} \boldsymbol{z}_m \\ \boldsymbol{y}_m \end{pmatrix} \right\|_2 \tag{8}$$

where $\beta = \|(\boldsymbol{I} - \boldsymbol{C}\boldsymbol{C}^{\mathrm{T}})\boldsymbol{r}_0\|_2$ and $\boldsymbol{B}_m = \boldsymbol{C}^{\mathrm{T}}\boldsymbol{K}\boldsymbol{V}_m$. The QR decomposition of $\underline{\boldsymbol{T}}_m$ gives

$$\underline{\boldsymbol{T}}_m = \hat{\boldsymbol{G}}_m^{\mathrm{T}}\underline{\boldsymbol{X}}_m \tag{9}$$

where $\hat{\boldsymbol{G}}_m$ is an orthogonal matrix of size $(m+1) \times (m+1)$, and $\underline{\boldsymbol{X}}_m$ is an upper triangular matrix of size $(m+1) \times m$ with bandwidth 3. $\hat{\boldsymbol{G}}$ is the product of a series of orthogonal matrices defining plane rotations, also called Given's rotations, $\hat{\boldsymbol{G}}_m = \boldsymbol{G}_m \cdots \boldsymbol{G}_2\boldsymbol{G}_1$ (see Algorithm 1). Let $\boldsymbol{X}_m$ be the leading $m \times m$ submatrix of $\underline{\boldsymbol{X}}_m$, and $\{x_{ij}\}$ and $\{t_{ij}\}$ be the coefficients of $\underline{\boldsymbol{X}}_m$ and $\underline{\boldsymbol{T}}_m$, respectively. The solution of the least squares problem (8) is then

$$\boldsymbol{y}_m = \boldsymbol{X}_m^{-1}\hat{\boldsymbol{G}}_m\beta\boldsymbol{e}_1, \quad \boldsymbol{z}_m = \boldsymbol{C}^{\mathrm{T}}\boldsymbol{r}_0 - \boldsymbol{B}_m\boldsymbol{y}_m \tag{10}$$

[¶]Note that $(I-CC^{\mathrm{T}})Kx=(I-CC^{\mathrm{T}})K(I-CC^{\mathrm{T}})x$ for $x\in$ range $(C)^{\perp}$, so that $(I-CC^{\mathrm{T}})K$ is *symmetric* over range$(C)^{\perp}$.

DOI: 10.1002/nme

This leads to

$$\begin{aligned}\boldsymbol{u}_m &= \boldsymbol{u}_0 + \boldsymbol{U}\boldsymbol{z}_m + \boldsymbol{V}_m\boldsymbol{y}_m \\ &= \boldsymbol{u}_0 + \boldsymbol{U}\boldsymbol{C}^{\mathrm{T}}\boldsymbol{r}_0 - \boldsymbol{U}\boldsymbol{B}_m\boldsymbol{y}_m + \boldsymbol{V}_m\boldsymbol{y}_m \\ &= \hat{\boldsymbol{u}}_0 - \boldsymbol{U}\boldsymbol{B}_m\boldsymbol{y}_m + \boldsymbol{V}_m\boldsymbol{y}_m \end{aligned} \tag{11}$$

where $\hat{\boldsymbol{u}}_0 = \boldsymbol{u}_0 + \boldsymbol{U}\boldsymbol{C}^{\mathrm{T}}\boldsymbol{r}_0$. Since $\boldsymbol{V}_m$ can be computed by a three-term recurrence, we only need the last two columns of $\boldsymbol{V}_m$ for the recurrence. However, $\boldsymbol{y}_m$ and $\boldsymbol{y}_{m-1}$ may differ in each coefficient, so that we still need all the columns of $\boldsymbol{V}_m$ and $\boldsymbol{U}\boldsymbol{B}_m$ to update $\boldsymbol{u}_m$. Let $\{\boldsymbol{v}_i\}$ and $\{\hat{\boldsymbol{b}}_i\}$ be the columns of $\boldsymbol{V}_m$ and $\boldsymbol{U}\boldsymbol{B}_m$, respectively. To allow us to discard the old $\boldsymbol{v}_i$ and $\hat{\boldsymbol{b}}_i$ vectors we use the same transformations as in MINRES. Let

$$\hat{\boldsymbol{B}}_m = \boldsymbol{U}\boldsymbol{B}_m, \quad \tilde{\boldsymbol{B}}_m = \hat{\boldsymbol{B}}_m\boldsymbol{X}_m^{-1}, \quad \tilde{\boldsymbol{V}}_m = \boldsymbol{V}_m\boldsymbol{X}_m^{-1}, \quad \tilde{\boldsymbol{y}}_m = \boldsymbol{X}_m\boldsymbol{y}_m \tag{12}$$

Then

$$\tilde{\boldsymbol{y}}_m = \hat{\boldsymbol{G}}_m\beta\boldsymbol{e}_1 = \boldsymbol{G}_m\boldsymbol{G}_{m-1}\cdots\boldsymbol{G}_1\beta\boldsymbol{e}_1 = \boldsymbol{G}_m\tilde{\boldsymbol{y}}_{m-1}$$

and only the mth and $(m+1)$th coefficients of $\tilde{\boldsymbol{y}}_{m-1}$ and $\tilde{\boldsymbol{y}}_m$ differ. The update (11) becomes

$$\begin{aligned}\boldsymbol{u}_m &= \hat{\boldsymbol{u}}_0 - \tilde{\boldsymbol{B}}_m\tilde{\boldsymbol{y}}_m + \tilde{\boldsymbol{V}}_m\tilde{\boldsymbol{y}}_m \\ &= \hat{\boldsymbol{u}}_0 - (\tilde{\boldsymbol{B}}_{m-1}\tilde{\boldsymbol{y}}_{m-1} + \tilde{\boldsymbol{b}}_m\tilde{y}_{m,m}) + (\tilde{\boldsymbol{V}}_{m-1}\tilde{\boldsymbol{y}}_{m-1} + \tilde{\boldsymbol{v}}_m\tilde{y}_{m,m}) \\ &= \boldsymbol{u}_{m-1} - \tilde{\boldsymbol{b}}_m\tilde{y}_{m,m} + \tilde{\boldsymbol{v}}_m\tilde{y}_{m,m}\end{aligned}$$

where $\tilde{\boldsymbol{b}}_m$ and $\tilde{\boldsymbol{v}}_m$ are the mth columns of $\tilde{\boldsymbol{B}}_m$ and $\tilde{\boldsymbol{V}}_m$, respectively, and $\tilde{y}_{m,m}$ is the mth coefficient of vector $\tilde{\boldsymbol{y}}_m$. Therefore, we only need the last column of $\tilde{\boldsymbol{B}}_m$ and $\tilde{\boldsymbol{V}}_m$ to update $\boldsymbol{u}$. From the definition of $\tilde{\boldsymbol{B}}_m$ and $\tilde{\boldsymbol{V}}_m$ in (12), we have

$$\tilde{\boldsymbol{b}}_{m-2}x_{m-2,m} + \tilde{\boldsymbol{b}}_{m-1}x_{m-1,m} + \tilde{\boldsymbol{b}}_m x_{m,m} = \hat{\boldsymbol{b}}_m \tag{13}$$

$$\tilde{\boldsymbol{v}}_{m-2}x_{m-2,m} + \tilde{\boldsymbol{v}}_{m-1}x_{m-1,m} + \tilde{\boldsymbol{v}}_m x_{m,m} = \boldsymbol{v}_m \tag{14}$$

so that the columns of $\tilde{\boldsymbol{B}}_m$ and $\tilde{\boldsymbol{V}}_m$ can be computed by three-term recurrences as well.

Algorithm 1 outlines this modified MINRES that includes the recycle space into the search space. We give some further implementation details in Section 6. In this algorithm, because of the three-term recurrences, we do not need to restart. So, in exact arithmetic, there is no need to use the recycle space generated during the solution of a linear system in the solution of that same system.[‖] As a consequence, we can derive a more efficient method for recycling for symmetric

[‖] In floating point arithmetic, including the recycle space obtained from the current Krylov subspace may help remedy the loss of orthogonality that generally occurs due to rounding errors.

DOI: 10.1002/nme

Algorithm 1 Modified MINRES

1: $\boldsymbol{r}_0 \leftarrow \boldsymbol{f} - \boldsymbol{K}\boldsymbol{u}_0$
2: $\boldsymbol{u}_0 \leftarrow \boldsymbol{u}_0 + \boldsymbol{U}\boldsymbol{C}^{\mathrm{T}}\boldsymbol{r}_0$; $\quad \boldsymbol{r}_0 \leftarrow \boldsymbol{r}_0 - \boldsymbol{C}\boldsymbol{C}^{\mathrm{T}}\boldsymbol{r}_0$
3: $\boldsymbol{v}_1 \leftarrow \boldsymbol{r}_0/\|\boldsymbol{r}_0\|_2$; $\quad \tilde{\boldsymbol{y}} \leftarrow \|\boldsymbol{r}_0\|_2\boldsymbol{e}_1$
4: **for** $m = 1, \ldots$ **do**
5: $\quad \hat{\boldsymbol{v}}_{m+1} \leftarrow \boldsymbol{K}\boldsymbol{v}_m$
6: $\quad \hat{\boldsymbol{v}}_{m+1} \leftarrow \hat{\boldsymbol{v}}_{m+1} - \boldsymbol{C}(\boldsymbol{C}^{\mathrm{T}}\hat{\boldsymbol{v}}_{m+1})$; $\quad \hat{\boldsymbol{b}}_m \leftarrow \boldsymbol{U}(\boldsymbol{C}^{\mathrm{T}}\hat{\boldsymbol{v}}_{m+1})$
$\quad \ll$ use modified Gram–Schmidt orthogonalization for updating $\hat{\boldsymbol{v}}_{m+1} \gg$
7: $\quad t_{m-1,m} \leftarrow t_{m,m-1}$; $\quad \hat{\boldsymbol{v}} \leftarrow \hat{\boldsymbol{v}} - t_{m-1,m}\hat{\boldsymbol{v}}_{m-1}$
8: $\quad t_{m,m} \leftarrow \langle \hat{\boldsymbol{v}}, \boldsymbol{v}_m \rangle$; $\quad \hat{\boldsymbol{v}} \leftarrow \hat{\boldsymbol{v}} - t_{m,m}\boldsymbol{v}_m$
9: $\quad t_{m+1,m} \leftarrow \|\hat{\boldsymbol{v}}\|_2$; $\quad \boldsymbol{v}_{m+1} \leftarrow \hat{\boldsymbol{v}}/t_{m+1,m}$
10: $\quad \boldsymbol{X}_{:,m} \leftarrow \boldsymbol{G}_{m-1}\boldsymbol{G}_{m-2}\underline{\boldsymbol{T}}_{:,m}$
$\quad \ll$ apply the Given's rotations from the previous two iterations to $\gg$
$\quad \ll$ the new column of $\underline{\boldsymbol{T}}_m \gg$
11: $\quad$ Compute Given's rotation $\boldsymbol{G}_m$ such that $\boldsymbol{X}_{:,m} \leftarrow \boldsymbol{G}_m\boldsymbol{X}_{:,m}$ has a zero coefficient at position $(m+1, m)$ $\quad \ll$ see MINRES [31, p. 41–44] $\gg$
12: $\quad \tilde{\boldsymbol{y}} \leftarrow \boldsymbol{G}_m\tilde{\boldsymbol{y}}$
13: $\quad \tilde{\boldsymbol{v}}_m \leftarrow x_{m,m}^{-1}(\boldsymbol{v}_m - \tilde{\boldsymbol{v}}_{m-1}x_{m-1,m} - \tilde{\boldsymbol{v}}_{m-2}x_{m-2,m})$
14: $\quad \tilde{\boldsymbol{b}}_m \leftarrow x_{m,m}^{-1}(\hat{\boldsymbol{b}}_m - \tilde{\boldsymbol{b}}_{m-1}x_{m-1,m} - \tilde{\boldsymbol{b}}_{m-2}x_{m-2,m})$
15: $\quad \boldsymbol{u}_m \leftarrow \boldsymbol{u}_{m-1} + \tilde{\boldsymbol{v}}_m\tilde{y}_m - \tilde{\boldsymbol{b}}_m\tilde{y}_m$ $\quad \ll \tilde{y}_m$ is the mth entry of vector $\tilde{\boldsymbol{y}} \gg$
16: **end for**

matrices. Although the Lanczos recurrence requires only the latest two basis vectors from the Krylov subspace (Lanczos vectors) for orthogonalization and restarting is not necessary, we do need all the Lanczos vectors to compute a recycle space. Therefore, to limit the memory requirements, we update the selected recycle space periodically. In this case, a *cycle* refers to the solution process between two updates of the recycle space.

We use s to denote the maximum length of a cycle (and hence the maximum number of Lanczos vectors kept), and k to denote the number of linearly independent vectors selected for recycling. We use RMINRES(s, k) to indicate the recycling MINRES method with the parameters s and k. The matrix $\boldsymbol{V}_j$ contains the Lanczos vectors generated in the jth cycle, $\boldsymbol{V}_j = [\boldsymbol{v}_{(j-1)s+1}, \ldots, \boldsymbol{v}_{js}]$, and the matrix $\underline{\boldsymbol{V}}_j = [\boldsymbol{v}_{(j-1)s}, \ldots, \boldsymbol{v}_{js+1}]$ denotes $\boldsymbol{V}_j$ extended with one previous and one subsequent Lanczos vector. Then, for the jth cycle, the modified Lanczos process gives

$$(\boldsymbol{I} - \boldsymbol{C}\boldsymbol{C}^{\mathrm{T}})\boldsymbol{K}\boldsymbol{V}_j = \underline{\boldsymbol{V}}_j\underline{\boldsymbol{T}}_j \tag{15}$$

where $\overline{\underline{\boldsymbol{T}}}_j$ is the tridiagonal matrix $\underline{\boldsymbol{T}}_j$ with an additional row corresponding to $\boldsymbol{v}_{(j-1)s}$ at the top. The bottom row corresponds to $\boldsymbol{v}_{js+1}$. To be specific, $\overline{\underline{\boldsymbol{T}}}_j$ has the non-zero pattern shown in Figure 2.

Let $\boldsymbol{U}_{j-1}$ give the basis of a subspace that was selected at the end of cycle $j-1$ for the current linear system. $\boldsymbol{U}_{j-1}$ is used only to compute $\boldsymbol{U}_j$ after cycle j; it is not used in solving the current linear system. The final $\boldsymbol{U}_j$ will be used for the next linear system. Below, we discuss several options to compute $\boldsymbol{U}_j$ from $\boldsymbol{U}$, $\boldsymbol{U}_{j-1}$, and the matrix $\boldsymbol{V}_j$ containing the Lanczos vectors generated in the latest cycle for the current system.

DOI: 10.1002/nme

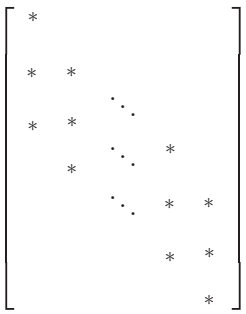

Figure 2. Non-zero pattern of $\underline{\overline{T}}_j$.

The modified Lanczos recurrence that includes the orthogonalization against $\boldsymbol{C}$ gives

$$\boldsymbol{K}[\boldsymbol{U}\ \boldsymbol{U}_{j-1}\ \boldsymbol{V}_j] = [\boldsymbol{C}\ \boldsymbol{C}_{j-1}\ \underline{\overline{\boldsymbol{V}}}_j] \begin{bmatrix} \boldsymbol{I} & \boldsymbol{0} & \boldsymbol{B}_j \\ \boldsymbol{0} & \boldsymbol{I} & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{0} & \underline{\overline{\boldsymbol{T}}}_j \end{bmatrix} \tag{16}$$

where $\boldsymbol{B}_j = \boldsymbol{C}^{\mathrm{T}}\boldsymbol{K}\boldsymbol{V}_j$ has been computed in the course of the iteration (see (15)). Now, we have several options for selecting a new matrix $\boldsymbol{U}_j$ for recycling. The first option is to compute the harmonic Ritz vectors of $\boldsymbol{K}$ with respect to range $([\boldsymbol{U}\ \boldsymbol{U}_{j-1}\ \boldsymbol{V}_j])$. The second option is similar to option 1, but we drop the $\boldsymbol{U}$ components from $\boldsymbol{U}_j$ if the linear solver has not converged. This is possible since $\boldsymbol{U}$ will be included again for the update of the recycle space after the next cycle for the same linear system. The third option is to obtain $\boldsymbol{U}_1$ from range $([\boldsymbol{U}\ \boldsymbol{V}_1])$ after the first cycle and $\boldsymbol{U}_j$ from range $([\boldsymbol{U}_{j-1}\ \boldsymbol{V}_j])$ after the jth cycle. In the last approach, the reappearance of $\boldsymbol{U}$ can be avoided as well. Since the formulation and the performance of these options are similar, we only discuss the third option in this paper.

Let

$$\boldsymbol{W}_j = [\boldsymbol{U}_{j-1}\ \boldsymbol{V}_j], \quad \widetilde{\boldsymbol{W}}_j = [\boldsymbol{C}\ \boldsymbol{C}_{j-1}\ \underline{\overline{\boldsymbol{V}}}_j], \quad \underline{\widetilde{\boldsymbol{H}}}_j = \begin{bmatrix} \boldsymbol{0} & \boldsymbol{B}_j \\ \boldsymbol{I} & \boldsymbol{0} \\ \boldsymbol{0} & \underline{\overline{\boldsymbol{T}}}_j \end{bmatrix} \tag{17}$$

Then, (16) gives

$$\boldsymbol{K}\boldsymbol{W}_j = \widetilde{\boldsymbol{W}}_j \underline{\widetilde{\boldsymbol{H}}}_j \tag{18}$$

Now, we compute the harmonic Ritz values and vectors of $\boldsymbol{K}$ with respect to the subspace range $(\boldsymbol{W}_j)$. These harmonic Ritz pairs $(\theta, \boldsymbol{w})$ are defined by the condition

$$\boldsymbol{K}\boldsymbol{w} - \theta\boldsymbol{w} \perp \text{range}(\boldsymbol{K}\boldsymbol{W}_j) \tag{19}$$

where $\boldsymbol{w} \in \text{range}\,(\boldsymbol{W}_j)$. If we write $\boldsymbol{w} = \boldsymbol{W}_j\boldsymbol{p}$, computing harmonic Ritz pairs is equivalent to solving the generalized eigenvalue problem

$$\underline{\widetilde{\boldsymbol{H}}}_j^{\mathrm{T}} \widetilde{\boldsymbol{W}}_j^{\mathrm{T}} \widetilde{\boldsymbol{W}}_j \underline{\widetilde{\boldsymbol{H}}}_j \boldsymbol{p} = \theta \underline{\widetilde{\boldsymbol{H}}}_j^{\mathrm{T}} \widetilde{\boldsymbol{W}}_j^{\mathrm{T}} \boldsymbol{W}_j \boldsymbol{p} \tag{20}$$

DOI: 10.1002/nme

After solving (20), we choose the k harmonic Ritz vectors with the (absolute) smallest harmonic Ritz values for recycling, and set $\tilde{\boldsymbol{U}}_j = \boldsymbol{W}_j \boldsymbol{P}_j$, where the columns of $\boldsymbol{P}_j$ are the chosen harmonic Ritz vectors. Now we have $\tilde{\boldsymbol{C}}_j = \boldsymbol{K}\tilde{\boldsymbol{U}}_j = \widetilde{\boldsymbol{W}}_j \underline{\widetilde{\boldsymbol{H}}}_j \boldsymbol{P}_j$. To obtain $\boldsymbol{C}_j$ with orthonormal columns, we compute the QR decomposition of $\widetilde{\boldsymbol{W}}_j$ (note that by construction almost all columns are already orthogonal),

$$\widetilde{\boldsymbol{W}}_j = \hat{\boldsymbol{W}}_j \boldsymbol{F}_j \tag{21}$$

and of $\boldsymbol{F}_j \underline{\widetilde{\boldsymbol{H}}}_j \boldsymbol{P}_j$,

$$\boldsymbol{F}_j \underline{\widetilde{\boldsymbol{H}}}_j \boldsymbol{P}_j = \boldsymbol{Q}_j \boldsymbol{R}_j \tag{22}$$

Next, we set

$$\boldsymbol{U}_j = \boldsymbol{W}_j \hat{\boldsymbol{P}}_j, \quad \boldsymbol{C}_j = \hat{\boldsymbol{W}}_j \boldsymbol{Q}_j = \widetilde{\boldsymbol{W}}_j \hat{\boldsymbol{Q}}_j \tag{23}$$

where $\hat{\boldsymbol{P}}_j = \boldsymbol{P}_j \boldsymbol{R}_j^{-1}$, and $\hat{\boldsymbol{Q}}_j = \boldsymbol{F}_j^{-1} \boldsymbol{Q}_j$. Then $\boldsymbol{C}_j$ is orthogonal and $\boldsymbol{K}\boldsymbol{U}_j = \boldsymbol{C}_j$. The two QR decompositions (21)–(22) are cheap to compute, because $\boldsymbol{F}_j$ has very few non-zeros, whose positions are known in advance, and $\boldsymbol{F}_j \underline{\widetilde{\boldsymbol{H}}}_j \boldsymbol{P}_j$ is a product of matrices of small dimensions.

Finally, to solve the generalized eigenvalue problem (20), we need the matrices $\underline{\widetilde{\boldsymbol{H}}}_j^{\mathrm{T}} \widetilde{\boldsymbol{W}}_j^{\mathrm{T}} \widetilde{\boldsymbol{W}}_j \underline{\widetilde{\boldsymbol{H}}}_j$ and $\underline{\widetilde{\boldsymbol{H}}}_j^{\mathrm{T}} \widetilde{\boldsymbol{W}}_j^{\mathrm{T}} \boldsymbol{W}_j$. We can simplify $\widetilde{\boldsymbol{W}}_j^{\mathrm{T}} \widetilde{\boldsymbol{W}}_j$ and $\widetilde{\boldsymbol{W}}_j^{\mathrm{T}} \boldsymbol{W}_j$ as follows:

$$\widetilde{\boldsymbol{W}}_j^{\mathrm{T}} \widetilde{\boldsymbol{W}}_j = \begin{bmatrix} \boldsymbol{I} & \boldsymbol{C}^{\mathrm{T}} \boldsymbol{C}_{j-1} & \boldsymbol{0} \\ \boldsymbol{C}_{j-1}^{\mathrm{T}} \boldsymbol{C} & \boldsymbol{I} & \boldsymbol{C}_{j-1}^{\mathrm{T}} \overline{\boldsymbol{V}}_j \\ \boldsymbol{0} & \overline{\boldsymbol{V}}_j^{\mathrm{T}} \boldsymbol{C}_{j-1} & \boldsymbol{I} \end{bmatrix} \tag{24}$$

$$\widetilde{\boldsymbol{W}}_j^{\mathrm{T}} \boldsymbol{W}_j = \begin{bmatrix} \boldsymbol{C}^{\mathrm{T}} \boldsymbol{U}_{j-1} & \boldsymbol{0} \\ \boldsymbol{C}_{j-1}^{\mathrm{T}} \boldsymbol{U}_{j-1} & \boldsymbol{C}_{j-1}^{\mathrm{T}} \boldsymbol{V}_j \\ \overline{\boldsymbol{V}}_j^{\mathrm{T}} \boldsymbol{U}_{j-1} & \overline{\boldsymbol{I}} \end{bmatrix} \tag{25}$$

where $\overline{\boldsymbol{I}}$ is an extended identity matrix with an additional row of zeros at the top and at the bottom. We can simplify the computation of most blocks in these two matrices further.

$$\boldsymbol{C}^{\mathrm{T}} \boldsymbol{C}_{j-1} = \boldsymbol{C}^{\mathrm{T}} \widetilde{\boldsymbol{W}}_{j-1} \hat{\boldsymbol{Q}}_{j-1} = [\boldsymbol{I} \;\; \boldsymbol{C}^{\mathrm{T}} \boldsymbol{C}_{j-2} \;\; \boldsymbol{0}] \hat{\boldsymbol{Q}}_{j-1} \tag{26}$$

$$\overline{\boldsymbol{V}}_j^{\mathrm{T}} \boldsymbol{C}_{j-1} = \overline{\boldsymbol{V}}_j^{\mathrm{T}} \widetilde{\boldsymbol{W}}_{j-1} \hat{\boldsymbol{Q}}_{j-1} = \begin{bmatrix} 0 & \cdots & 0 & 1 & 0 \\ 0 & \cdots & 0 & 0 & 1 \\ \vdots & \ddots & \ddots & \vdots & \vdots \\ 0 & \cdots & \cdots & 0 & 0 \end{bmatrix} \hat{\boldsymbol{Q}}_{j-1} \tag{27}$$

DOI: 10.1002/nme

$$\boldsymbol{C}^{\mathrm{T}}\boldsymbol{U}_{j-1} = \boldsymbol{C}^{\mathrm{T}}\boldsymbol{W}_{j-1}\hat{\boldsymbol{P}}_{j-1} = [\boldsymbol{C}^{\mathrm{T}}\boldsymbol{U}_{j-2} \;\; \boldsymbol{0}]\hat{\boldsymbol{P}}_{j-1} \tag{28}$$

$$\boldsymbol{C}_{j-1}^{\mathrm{T}}\boldsymbol{U}_{j-1} = \hat{\boldsymbol{Q}}_{j-1}^{\mathrm{T}}(\widetilde{\boldsymbol{W}}_{j-1}^{\mathrm{T}}\boldsymbol{W}_{j-1})\hat{\boldsymbol{P}}_{j-1} \tag{29}$$

$$\boldsymbol{C}_{j-1}^{\mathrm{T}}\boldsymbol{V}_j = \hat{\boldsymbol{Q}}_{j-1}^{\mathrm{T}}\begin{bmatrix}\boldsymbol{C}^{\mathrm{T}}\\ \boldsymbol{C}_{j-2}^{\mathrm{T}}\\ \underline{\overline{\boldsymbol{V}}}_{j-1}\end{bmatrix}\boldsymbol{V}_j = \hat{\boldsymbol{Q}}_{j-1}^{\mathrm{T}}\begin{bmatrix}0 & 0 & \cdots & 0\\ 0 & 0 & \cdots & 0\\ \vdots & \vdots & \ddots & \vdots\\ 1 & 0 & \cdots & 0\end{bmatrix} \tag{30}$$

From the above derivation, we can obtain $\underline{\overline{\boldsymbol{V}}}_j^{\mathrm{T}}\boldsymbol{C}_{j-1}$ and $\boldsymbol{C}_{j-1}^{\mathrm{T}}\boldsymbol{V}_j$ simply from $\hat{\boldsymbol{Q}}_{j-1}$ (known from the previous cycle), and we can compute $\boldsymbol{C}^{\mathrm{T}}\boldsymbol{C}_{j-1}$, $\boldsymbol{C}^{\mathrm{T}}\boldsymbol{U}_{j-1}$ and $\boldsymbol{C}_{j-1}^{\mathrm{T}}\boldsymbol{U}_{j-1}$ recursively with $2k^3$, $2k^3$ and $2k(k+s)(3k+s)$ flops, respectively. Therefore, the computation of these submatrices is very cheap. Only $\underline{\overline{\boldsymbol{V}}}_j^{\mathrm{T}}\boldsymbol{U}_{j-1}$ must be computed explicitly by matrix–matrix product, which takes about $2ksn$ flops. In summary, the cost of each update of the recycle space is about $(12k^2+6ks+6k+4)n$ flops, ignoring the terms that do not have a factor n. Compared with the cost of MINRES, which is mainly determined by the matrix–vector product and the forward and backward solve for the preconditioner for each iteration, the overhead of the subspace selection is modest (see the timing results in Section 7).

The general form of the RMINRES is outlined in Algorithms 2 and 3, below. For brevity, we do not explicitly deal with the slight changes for the first linear system in the sequence, when we do not have a recycle space $\hat{\boldsymbol{U}}$ yet, and for the first cycle for each linear system, i.e. $j=1$, when we do not have $\boldsymbol{U}_{j-1}$ and $\boldsymbol{C}_{j-1}$ yet. For the first system in the sequence, we define

$$\begin{aligned}&\boldsymbol{W}_1 = \boldsymbol{V}_1, \quad \widetilde{\boldsymbol{W}}_1 = \underline{\boldsymbol{V}}_1\\ &\boldsymbol{W}_j = [\boldsymbol{U}_{j-1} \;\; \boldsymbol{V}_j], \quad \widetilde{\boldsymbol{W}}_j = [\boldsymbol{C}_{j-1} \;\; \underline{\overline{\boldsymbol{V}}}_j] \quad \text{for } j>1\end{aligned} \tag{31}$$

And for the first cycle of each subsequent system, we let

$$\boldsymbol{W}_1 = [\boldsymbol{U} \;\; \boldsymbol{V}_1], \quad \widetilde{\boldsymbol{W}}_1 = [\boldsymbol{C} \;\; \underline{\boldsymbol{V}}_1] \tag{32}$$

The $\boldsymbol{U}_j$ and $\boldsymbol{C}_j$ for these special cases can be easily derived using the simplified definitions of $\boldsymbol{W}$ and $\widetilde{\boldsymbol{W}}$ in (31) and (32) following the approach that we give for general case, and Algorithm 3 can be modified correspondingly. Again, we provide some further implementation details in Section 6.

5. PRECONDITIONING FOR TOPOLOGY OPTIMIZATION

The convergence rate of Krylov methods for a symmetric matrix depends only on the spectrum of the matrix. In fact, the ratio between the absolute largest and smallest eigenvalues governs a worst-case upper bound on the convergence rate. In large-scale finite element simulations in physics and engineering, the linear systems tend to be ill-conditioned. In topology optimization, this problem is exacerbated by the wide range of magnitudes of the element densities.

DOI: 10.1002/nme

Algorithm 2 Solve $\boldsymbol{Ku}=\boldsymbol{f}$ with $\boldsymbol{u}_0$ and $\hat{\boldsymbol{U}}$ from the previous system, and s and k

$\hat{\boldsymbol{C}} \leftarrow \boldsymbol{K}\hat{\boldsymbol{U}}$
compute the QR decomposition of $\hat{\boldsymbol{C}}$ as $\hat{\boldsymbol{C}}=\boldsymbol{C}\tilde{\boldsymbol{R}}$
$\boldsymbol{U} \leftarrow \hat{\boldsymbol{U}}\tilde{\boldsymbol{R}}^{-1}$ $\ll$ use backward substitution $\gg$
$\boldsymbol{u} \leftarrow \boldsymbol{u}_0$; $\quad \boldsymbol{r} \leftarrow \boldsymbol{f}-\boldsymbol{Ku}$
$\boldsymbol{u} \leftarrow \boldsymbol{u}+\boldsymbol{U}(\boldsymbol{C}^{\mathrm{T}}\boldsymbol{r})$; $\quad \boldsymbol{r} \leftarrow \boldsymbol{r}-\boldsymbol{C}(\boldsymbol{C}^{\mathrm{T}}\boldsymbol{r})$ $\ll$ use Modified Gram–Schmidt $\gg$
$\gamma=\|\boldsymbol{r}\|_2$; $\quad \boldsymbol{v}_1 \leftarrow \boldsymbol{r}/\gamma$; $\quad \tilde{\boldsymbol{y}} \leftarrow \gamma\boldsymbol{e}_1$
$m \leftarrow 0$; $\quad j \leftarrow 0$
while $\gamma/\|\boldsymbol{f}\|_2 >$ tol **do**
  $m \leftarrow m+1$
  $\hat{\boldsymbol{v}}=\boldsymbol{Kv}_m$
  $\hat{\boldsymbol{v}} \leftarrow \hat{\boldsymbol{v}}-\boldsymbol{C}(\boldsymbol{C}^{\mathrm{T}}\hat{\boldsymbol{v}})$; $\quad \hat{\boldsymbol{b}} \leftarrow \boldsymbol{U}(\boldsymbol{C}^{\mathrm{T}}\hat{\boldsymbol{v}})$ $\ll$ use Modified Gram–Schmidt $\gg$
  $t_{m-1,m} \leftarrow t_{m,m-1}$; $\quad \hat{\boldsymbol{v}} \leftarrow \hat{\boldsymbol{v}}-t_{m-1,m}\hat{\boldsymbol{v}}_{m-1}$
  $t_{m,m} \leftarrow \langle \hat{\boldsymbol{v}}, \boldsymbol{v}_m \rangle$; $\quad \hat{\boldsymbol{v}} \leftarrow \hat{\boldsymbol{v}}-t_{m,m}\boldsymbol{v}_m$
  $t_{m+1,m} \leftarrow \|\hat{\boldsymbol{v}}\|_2$; $\quad \boldsymbol{v}_{m+1} \leftarrow \hat{\boldsymbol{v}}/t_{m+1,m}$
  $\boldsymbol{X}_{:,m} \leftarrow \boldsymbol{G}_{m-1}\boldsymbol{G}_{m-2}\underline{\boldsymbol{T}}_{:,m}$
  compute Given's rotation $\boldsymbol{G}_m$ such that $\boldsymbol{X}_{:,m} \leftarrow \boldsymbol{G}_m\boldsymbol{X}_{:,m}$ has a zero coefficient at position $(m+1, m)$
  $\tilde{\boldsymbol{y}} \leftarrow \boldsymbol{G}_m\tilde{\boldsymbol{y}}$
  $\tilde{\boldsymbol{v}}_m \leftarrow x_{m,m}^{-1}(\boldsymbol{v}_m-\tilde{\boldsymbol{v}}_{m-1}x_{m-1,m}-\tilde{\boldsymbol{v}}_{m-2}x_{m-2,m})$; then drop $\tilde{\boldsymbol{v}}_{m-2}$
  $\tilde{\boldsymbol{b}}_m \leftarrow x_{m,m}^{-1}(\hat{\boldsymbol{b}}-\tilde{\boldsymbol{b}}_{m-1}x_{m-1,m}-\tilde{\boldsymbol{b}}_{m-2}x_{m-2,m})$; then drop $\tilde{\boldsymbol{b}}_{m-2}$
  $\boldsymbol{u} \leftarrow \boldsymbol{u}+\tilde{\boldsymbol{v}}_m\tilde{y}_m-\tilde{\boldsymbol{b}}_m\tilde{y}_m$; $\quad \gamma \leftarrow \tilde{y}_{m+1}$ $\ll \tilde{y}_m$ and $\tilde{y}_{m+1}$ are the mth and $(m+1)$th entries of vector $\tilde{\boldsymbol{y}}$, so $\gamma=\|\boldsymbol{r}\|_2 \gg$
  if $(\gamma/\|\boldsymbol{f}\|_2 \leqslant$ tol) or $(\mathrm{mod}(m, s)=0)$ **then**
    $j \leftarrow j+1$
    compute $\boldsymbol{U}_j$ and $\boldsymbol{C}_j$ following Algorithm 3
    drop $\boldsymbol{U}_{j-1}$, $\boldsymbol{C}_{j-1}$, and all $\boldsymbol{v}$ vectors except $\boldsymbol{v}_{m+1}$ and $\boldsymbol{v}_m$
  end if
end while
return $\boldsymbol{u}$ as the solution and initial guess $\boldsymbol{u}_0$ for the next system,
and return $\boldsymbol{U}_j$ as the $\hat{\boldsymbol{U}}$ for the next system

Ill-conditioning creates two problems for numerical simulation. First, ill-conditioning may seriously affect the accuracy of the computed solution. Second, the convergence of iterative methods is poor for ill-conditioned problems. The second problem is generally addressed by proper preconditioning. In principle, preconditioning does not alleviate the potential accuracy problem, because a preconditioner that is effective for an ill-conditioned matrix has to be fairly ill-conditioned itself. This leads to two multiplications by ill-conditioned matrices in each iteration (or three for two-sided preconditioning), which may lead in turn to serious accumulation of numerical errors. However, in certain cases the accuracy problem can be relieved by properly scaling the linear system. We show that this is the case for topology optimization. This leads to a preprocessing step and a preconditioning step (or two preconditioners depending on one's view).

In the next section, we discuss the preprocessing and preconditioner that we used for our numerical experiments. We illustrate the idea of rescaling from a mechanical point of view for

DOI: 10.1002/nme

Algorithm 3 Compute $\boldsymbol{U}_j$ and $\boldsymbol{C}_j$

1: compute $\boldsymbol{C}^{\mathrm{T}}\boldsymbol{C}_{j-1}$ following (26) with $\boldsymbol{C}^{\mathrm{T}}\boldsymbol{C}_{j-2}$ and $\hat{\boldsymbol{Q}}_{j-1}$ already available from the $(j-1)$th cycle
2: compute $\underline{\overline{\boldsymbol{V}}}_j^{\mathrm{T}}\boldsymbol{C}_{j-1}$ following (27) with $\hat{\boldsymbol{Q}}_{j-1}$ already available from the $(j-1)$th cycle
$\ll$ Copy the last two rows of $\hat{\boldsymbol{Q}}_{j-1}$ $\gg$
3: compute $\boldsymbol{C}^{\mathrm{T}}\boldsymbol{U}_{j-1}$ following (28) with $\boldsymbol{C}^{\mathrm{T}}\boldsymbol{U}_{j-2}$ and $\hat{\boldsymbol{P}}_{j-1}$ already available from the $(j-1)$th cycle
4: compute $\boldsymbol{C}_{j-1}^{\mathrm{T}}\boldsymbol{U}_{j-1}$ following (29) with $\hat{\boldsymbol{Q}}_{j-1}$, $\widetilde{\boldsymbol{W}}_{j-1}^{\mathrm{T}}\boldsymbol{W}_{j-1}$ and $\hat{\boldsymbol{P}}_{j-1}$ already available from the $(j-1)$th cycle
5: compute $\boldsymbol{C}_{j-1}^{\mathrm{T}}\boldsymbol{V}_j$ following (30) with $\hat{\boldsymbol{Q}}_{j-1}$ already available from the $(j-1)$th cycle
$\ll$ Copy the last column of $\hat{\boldsymbol{Q}}_{j-1}^{\mathrm{T}}$ $\gg$
6: compute $\underline{\overline{\boldsymbol{V}}}_j^{\mathrm{T}}\boldsymbol{U}_{j-1}$ by matrix-matrix product
7: assemble $\widetilde{\boldsymbol{W}}_j^{\mathrm{T}}\widetilde{\boldsymbol{W}}_j$ and $\widetilde{\boldsymbol{W}}_j^{\mathrm{T}}\boldsymbol{W}_j$ following (24) and (25)
8: solve the generalized eigenvalue problem (20) and pick the k generalized eigenvectors corresponding to the k smallest eigenvalues to form the columns of $\boldsymbol{P}_j$
9: compute the QR decomposition of $\widetilde{\boldsymbol{W}}_j$ as $\widetilde{\boldsymbol{W}}_j = \hat{\boldsymbol{W}}_j\boldsymbol{F}_j$
$\ll$ Orthogonalize the first two columns of $\underline{\overline{\boldsymbol{V}}}_j$ against $\boldsymbol{C}_{j-1}$ $\gg$
10: compute the QR decomposition of $\boldsymbol{F}_j\underline{\widetilde{\boldsymbol{H}}}_j\boldsymbol{P}_j$ as $\boldsymbol{F}_j\underline{\widetilde{\boldsymbol{H}}}_j\boldsymbol{P}_j = \boldsymbol{Q}_j\boldsymbol{R}_j$
11: $\hat{\boldsymbol{P}}_j \leftarrow \boldsymbol{P}_j\boldsymbol{R}_j^{-1}$; $\quad \hat{\boldsymbol{Q}}_j \leftarrow \boldsymbol{F}_j^{-1}\boldsymbol{Q}_j$
12: $\boldsymbol{U}_j \leftarrow \boldsymbol{W}_j\hat{\boldsymbol{P}}_j$; $\quad \boldsymbol{C}_j \leftarrow \widetilde{\boldsymbol{W}}_j\hat{\boldsymbol{Q}}_j$

a 1D problem in Section 5.2. Borrvall and Petersson [1] suggested, without further discussion, that the condition number of the matrix can be as large as the ratio of maximum to minimum density. We show that this ratio provides only a lower bound on the condition number and that the actual condition number typically is even larger. The actual conditioning is a combination of this ratio and the conditioning of a corresponding problem with constant density.

5.1. Preconditioning

The following analysis shows how ill-conditioned the stiffness matrices can be.

The two-norm condition number of a matrix $\boldsymbol{K}$ can be defined as

$$\kappa(\boldsymbol{K}) = \frac{\max_{\|\boldsymbol{u}\|=1} \|\boldsymbol{K}\boldsymbol{u}\|}{\min_{\|\boldsymbol{u}\|=1} \|\boldsymbol{K}\boldsymbol{u}\|}$$

Since

$$\min_{\|\boldsymbol{u}\|=1} \|\boldsymbol{K}\boldsymbol{u}\| \leqslant \|\boldsymbol{K}\boldsymbol{e}_\ell\| = \|\mathbf{k}_\ell\| \leqslant \max_{\|\boldsymbol{u}\|=1} \|\boldsymbol{K}\boldsymbol{u}\| \quad \text{for any } \ell = 1, \ldots, n$$

where $\mathbf{k}_\ell$ is the ℓth column of $\boldsymbol{K}$ and $\boldsymbol{e}_\ell$ is the Cartesian basis vector with the ℓth coefficient equal to 1, we have

$$\kappa(\boldsymbol{K}) \geqslant \frac{\|\mathbf{k}_{\ell_1}\|}{\|\mathbf{k}_{\ell_2}\|} \quad \text{for any } \ell_1, \ell_2 = 1, \ldots, n$$

DOI: 10.1002/nme

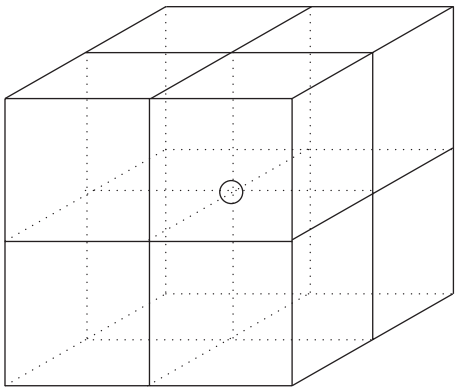

Figure 3. N_ℓ: the set of elements associated with the ℓth d.o.f. indicated by the circle in the middle.

In topology optimization, a column of the stiffness matrix is given by

$$\mathbf{k}_\ell = \sum_{e \in N_\ell} \rho_e^p \boldsymbol{L}_e^{\mathrm{T}} \boldsymbol{K}_0 \boldsymbol{L}_e \boldsymbol{e}_\ell$$

where $\boldsymbol{K}_0$ is the unit element stiffness matrix, L_e is the local-to-global transformation matrix, and N_ℓ is the set of elements that are associated with the ℓth d.o.f. These usually form a $2 \times 2 \times 2$ block in the 3D mesh (see Figure 3). If the blocks associated with d.o.f. ℓ_1 and ℓ_2 are solid and void, respectively, namely $\rho_e = 1$ for $e \in N_{\ell_1}$ and $\rho_e = \rho_0$ for $e \in N_{\ell_2}$, we have

$$\mathbf{k}_{\ell_1} = \sum_{e \in N_\ell} \boldsymbol{L}_e^{\mathrm{T}} \boldsymbol{K}_0 \boldsymbol{L}_e \boldsymbol{e}_{\ell_1}$$

$$\mathbf{k}_{\ell_2} = \rho_0^p \sum_{e \in N_\ell} \boldsymbol{L}_e^{\mathrm{T}} \boldsymbol{K}_0 \boldsymbol{L}_e \boldsymbol{e}_{\ell_2}$$

Then, assuming that the elements are uniform and isotropic, we have

$$\kappa(\boldsymbol{K}) \geqslant \frac{\|\mathbf{k}_{\ell_1}\|}{\|\mathbf{k}_{\ell_2}\|} = \frac{1}{\rho_0^p} \tag{33}$$

For $\rho_0 = 10^{-3}$ and $p = 3$, which are commonly used in topology optimization, the condition number of the stiffness matrix will be greater than 10^9 when solid and void areas begin to appear in the design domain.

Note that this analysis provides only a lower bound on the condition number, and that structures from homogeneous material can also have large condition numbers. However, the analysis suggests that, to a significant degree, the ill-conditioning comes from the poor scaling of the material densities over the design domain. We can understand this intuitively as follows. A change in an algebraic degree of freedom, say the Cartesian basis vector $\boldsymbol{e}_j$, associated with a nodal basis function in a region with very small density corresponds to a displacement that requires a very small amount of energy ($\boldsymbol{e}_j^{\mathrm{T}} \boldsymbol{K}(\boldsymbol{\rho}) \boldsymbol{e}_j$ small). However, that same change in an algebraic degree of freedom, $\boldsymbol{e}_i$, associated with a nodal basis function in a region with large density corresponds to a

DOI: 10.1002/nme

displacement of the same magnitude that requires a large amount of energy ($\boldsymbol{e}_i^{\mathrm{T}}\boldsymbol{K}(\boldsymbol{\rho})\boldsymbol{e}_i$ large). Since for symmetric $\boldsymbol{K}$

$$\kappa(\boldsymbol{K}(\boldsymbol{\rho})) \geqslant \frac{\boldsymbol{e}_i^{\mathrm{T}}\boldsymbol{K}(\boldsymbol{\rho})\boldsymbol{e}_i}{\boldsymbol{e}_j^{\mathrm{T}}\boldsymbol{K}(\boldsymbol{\rho})\boldsymbol{e}_j}$$

this shows that the system is inherently ill-conditioned. Therefore, we expect that we can reduce the ill-conditioning due to the large variation in density by scaling the linear system such that changes of equal magnitude in algebraic degrees of freedom yield equal changes in energy ($\boldsymbol{e}_i^{\mathrm{T}}\boldsymbol{K}(\boldsymbol{\rho})\boldsymbol{e}_i = \boldsymbol{e}_j^{\mathrm{T}}\boldsymbol{K}(\boldsymbol{\rho})\boldsymbol{e}_j$ for all i and j). Since this is the case for a problem with homogeneous density, we expect that this scaling reduces the condition number of the stiffness matrix to roughly that for a similar problem with homogeneous density. Indeed, in general we obtain a condition number that is slightly better than that for a problem with constant density. Alternatively, in light of (33) we may want to scale the linear system such that all columns have the same norm. In the next section, we discuss the effects of rescaling for a simple 1D problem with heterogeneous density.

We propose to rescale the stiffness matrices $\boldsymbol{K}$ by multiplying with a diagonal matrix on both sides (for symmetry),

$$\tilde{\boldsymbol{K}} = \boldsymbol{D}^{-1/2}\boldsymbol{K}\boldsymbol{D}^{-1/2}$$

where the entries of the diagonal matrix $\boldsymbol{D}$ are either the diagonal coefficients of $\boldsymbol{K}$ or the absolute column sums of $\boldsymbol{K}$, i.e. $d_i = \|\mathbf{k}_i\|_1$. In Figure 4 we compare the condition numbers of stiffness matrices that arise in topology optimization for a model problem on a $18 \times 6 \times 3$ mesh with the condition numbers of the rescaled stiffness matrices. The model problem is the same as that used in the numerical results section. The condition numbers of the stiffness matrices quickly rise to about 10^{11} after only a few optimization steps. However, the condition

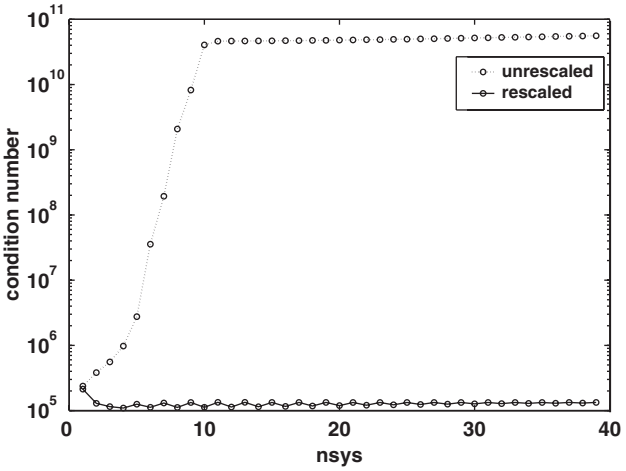

Figure 4. Condition numbers of stiffness matrices and rescaled stiffness matrices for the model problem in Figure 7 on a $18 \times 6 \times 3$ mesh.

DOI: 10.1002/nme

numbers of the rescaled matrices remain at about the same level as those at the beginning (at about 10^5).

To obtain rapid convergence for iterative methods, it is important to further reduce the condition number after rescaling by more general preconditioning techniques. In our numerical experiments, we use an incomplete Cholesky decomposition with zero fill-in of the rescaled stiffness matrix as a preconditioner [32],

$$\tilde{\boldsymbol{K}} = \boldsymbol{D}^{-1/2}\boldsymbol{K}\boldsymbol{D}^{-1/2} \approx \boldsymbol{L}\boldsymbol{L}^{\mathrm{T}}$$

Finally, we note that diagonal scaling does not decrease the relative accuracy of the matrix coefficients, and hence such scaling leads to a real improvement in the worst-case numerical error in the computed solution. The second type of preconditioning, using the incomplete Cholesky decomposition, improves the rate of convergence, but will typically not affect the accuracy of the computed solutions. Since the Cholesky decomposition may fail for a very ill-conditioned matrix, we explicitly rescale the stiffness matrix before we compute the incomplete Cholesky decomposition.

We solve the preconditioned system

$$\boldsymbol{L}^{-1}\tilde{\boldsymbol{K}}\boldsymbol{L}^{-\mathrm{T}}\tilde{\boldsymbol{u}} = \tilde{\boldsymbol{f}}$$

to get $\tilde{\boldsymbol{u}}$, where $\tilde{\boldsymbol{f}} = \boldsymbol{L}^{-1}\boldsymbol{D}^{-1/2}\boldsymbol{f}$. Then, we compute

$$\boldsymbol{u} = \boldsymbol{D}^{-1/2}\boldsymbol{L}^{-\mathrm{T}}\tilde{\boldsymbol{u}}$$

to obtain the solution of the original system $\boldsymbol{K}\boldsymbol{u} = \boldsymbol{f}$.

Preconditioners other than incomplete Cholesky have been proposed for topology optimization, e.g. block Jacobi preconditioners [3], especially for use on parallel computers.

5.2. *Rescaling for a 1D elasticity problem*

We use an idealized 1D elasticity problem with piecewise constant modulus of elasticity to explain the idea of rescaling. Consider the following problem:

Find $u(x)$ with boundary conditions $u(0) = 0$ and $u(1) = 1$, such that

$$\boldsymbol{a}(u, v) \equiv \int_0^1 E(x) u_x v_x \, \mathrm{d}x = 0 \quad \text{with } E(x) \geqslant E_0 > 0$$

for all v with $v(0) = v(1) = 0$. Furthermore, following the typical case of topology optimization, we assume that E is piecewise constant (see Figure 5) and varies over a large range of values.

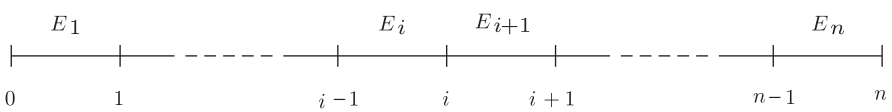

Figure 5. Piecewise constant modulus of elasticity E_i.

DOI: 10.1002/nme

For simplicity, we discretize the problem using piecewise linear nodal basis functions and a mesh with equal length elements. This yields the following linear system:

$$\begin{bmatrix} E_1+E_2 & -E_2 & & & \\ -E_2 & E_2+E_3 & -E_3 & & \\ & \ddots & \ddots & \ddots & \\ & & -E_{n-1} & E_{n-1}+E_n \end{bmatrix} \begin{pmatrix} u_1 \\ u_2 \\ \vdots \\ u_{n-1} \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ \vdots \\ E_n \end{pmatrix} \tag{34}$$

We can write this system of equations as follows (note $(Eu_x)_x = 0 \Leftrightarrow Eu_x = \text{constant}$):

$$E_i(u_i - u_{i-1}) - E_{i+1}(u_{i+1} - u_i) = 0 \quad \text{for } i = 1, \ldots, n-1$$

where we have used $u_0 = 0$ and $u_n = 1$. Introducing the difference matrix

$$\mathbf{D}_1 = \begin{bmatrix} 1 & & & \\ -1 & 1 & & \\ & \ddots & \ddots & \\ & & -1 & 1 \end{bmatrix}$$

and the diagonal matrix $\mathbf{\Omega} = \text{diag}(E_1, E_2, \ldots, E_{n-1})$, we can write (34) as

$$(\mathbf{D}_1^{\mathrm{T}} \mathbf{\Omega} \mathbf{D}_1 + E_n \boldsymbol{e}_{n-1} \boldsymbol{e}_{n-1}^{\mathrm{T}})\mathbf{u} = E_n \boldsymbol{e}_{n-1} \tag{35}$$

For a problem with constant modulus of elasticity, E, this equation gives

$$E(\mathbf{D}_1^{\mathrm{T}} \mathbf{D}_1 + \boldsymbol{e}_{n-1} \boldsymbol{e}_{n-1}^{\mathrm{T}})\mathbf{u} = E \boldsymbol{e}_{n-1} \tag{36}$$

where $\mathbf{D}_1^{\mathrm{T}} \mathbf{D}_1 + \boldsymbol{e}_{n-1} \boldsymbol{e}_{n-1}^{\mathrm{T}}$ is the well-known tridiagonal matrix with coefficients $[-1\ 2\ -1]$.

Next, we want to demonstrate two issues. Comparing (35) with (36), it is clear that extreme ill-conditioning in (35) must arise from the scaling introduced by $\mathbf{\Omega}$. First, we demonstrate that this leads to a condition number (bound) that is roughly the product of the condition number of the constant elasticity problem and the condition number of $\mathbf{\Omega}$. Second, we show how a proper (re)scaling brings the condition number down to that for the constant elasticity case, if the solution is properly defined. We note that for general choices of $\mathbf{\Omega}$ there may be no diagonal scaling that reduces the condition number. For example, in 1D if we have two non-adjacent 'holes' the displacement for material in between the holes is not properly defined (as the modulus of elasticity goes to zero), since there is no connection to any point with a fixed displacement. In higher dimensions this is rarely a problem, as the topology optimization algorithm leads to energetically favourable solutions that do not have such anomalies.

Below, we need the following well-known result for symmetric positive definite matrices $\mathbf{A}, \mathbf{B} \in \mathbb{R}^{n \times n}$ and $\alpha, \beta \in \mathbb{R}^+$ [33, pp. 338–389]. Let $\mathbf{A}$ and $\mathbf{B}$ be such that for all $\mathbf{u} \neq 0$

$$\alpha \leqslant \frac{\boldsymbol{u}^{\mathrm{T}} \boldsymbol{A} \boldsymbol{u}}{\boldsymbol{u}^{\mathrm{T}} \boldsymbol{B} \boldsymbol{u}} \leqslant \beta \tag{37}$$

Then

$$\kappa(\boldsymbol{B}^{-1/2} \boldsymbol{A} \boldsymbol{B}^{-1/2}) \leqslant \frac{\beta}{\alpha} \tag{38}$$

where κ denotes the condition number.

DOI: 10.1002/nme

Using (37) and (38), we can bound the condition number of the matrix in (35) as follows:

$$\kappa(\mathbf{D}_1^{\mathrm{T}}\boldsymbol{\Omega}\mathbf{D}_1 + E_n\boldsymbol{e}_{n-1}\boldsymbol{e}_{n-1}^{\mathrm{T}}) = \frac{\max_{\|\mathbf{u}\|=1}\mathbf{u}^{\mathrm{T}}(\mathbf{D}_1^{\mathrm{T}}\boldsymbol{\Omega}\mathbf{D}_1 + E_n\boldsymbol{e}_{n-1}\boldsymbol{e}_{n-1}^{\mathrm{T}})\mathbf{u}}{\min_{\|\mathbf{u}\|=1}\mathbf{u}^{\mathrm{T}}(\mathbf{D}_1^{\mathrm{T}}\boldsymbol{\Omega}\mathbf{D}_1 + E_n\boldsymbol{e}_{n-1}\boldsymbol{e}_{n-1}^{\mathrm{T}})\mathbf{u}}$$

Let $\mathbf{y} = \mathbf{D}_1\mathbf{u}$ and hence $\mathbf{u} = \mathbf{D}_1^{-1}\mathbf{y}$. Then

$$\begin{aligned}\mathbf{u}^{\mathrm{T}}\mathbf{D}_1^{\mathrm{T}}\boldsymbol{\Omega}\mathbf{D}_1\mathbf{u} + E_n\mathbf{u}^{\mathrm{T}}\boldsymbol{e}_{n-1}\boldsymbol{e}_{n-1}^{\mathrm{T}}\mathbf{u} &= (\mathbf{D}_1\mathbf{u})^{\mathrm{T}}\boldsymbol{\Omega}(\mathbf{D}_1\mathbf{u}) + E_n u_{n-1}^2 \\ &= \frac{\mathbf{y}^{\mathrm{T}}\boldsymbol{\Omega}\mathbf{y} + E_n(y_1 + \cdots + y_{n-1})^2}{(\mathbf{D}_1^{-1}\mathbf{y})^{\mathrm{T}}(\mathbf{D}_1^{-1}\mathbf{y})} \end{aligned} \tag{39}$$

Let $E_{\min} = \min_i(E_i)$, $E_{\max} = \max_i(E_i)$, and let $\lambda_{\min}$ be the smallest eigenvalue of the matrix $\mathbf{D}_1\mathbf{D}_1^{\mathrm{T}}$ and $\lambda_{\max}$ its largest eigenvalue. Furthermore, note that $\mathbf{D}_1\mathbf{D}_1^{\mathrm{T}}$ and $\mathbf{D}_1^{\mathrm{T}}\mathbf{D}_1$ have the same eigenvalues. Since $\mathbf{y}$ appears quadratically in both the numerator and the denominator, we can assume $\mathbf{y}$ to be normalized. Then, (39) gives

$$E_{\min}\lambda_{\min} \leqslant \frac{\mathbf{y}^{\mathrm{T}}\boldsymbol{\Omega}\mathbf{y} + E_n(y_1 + \cdots + y_{n-1})^2}{(\mathbf{D}_1^{-1}\mathbf{y})^{\mathrm{T}}(\mathbf{D}_1^{-1}\mathbf{y})} \leqslant n E_{\max}\lambda_{\max}$$

which finally gives

$$\kappa(\mathbf{D}_1^{\mathrm{T}}\boldsymbol{\Omega}\mathbf{D}_1 + E_n\boldsymbol{e}_{n-1}\boldsymbol{e}_{n-1}^{\mathrm{T}}) \leqslant n\frac{E_{\max}}{E_{\min}}\frac{\lambda_{\max}}{\lambda_{\min}} = n\kappa(\boldsymbol{\Omega})\kappa(\mathbf{D}_1^{\mathrm{T}}\mathbf{D}_1)$$

Next, we show that scaling a problem (without non-adjacent 'holes') reduces the condition number of the linear system to roughly that of a problem with constant elasticity. Let

$$\mathbf{S} = \mathrm{diag}(E_1 + E_2, E_2 + E_3, \ldots, E_{n-1} + E_n) \tag{40}$$

We have

$$\begin{aligned}&\frac{\mathbf{u}^{\mathrm{T}}(\mathbf{D}_1^{\mathrm{T}}\boldsymbol{\Omega}\mathbf{D}_1 + E_n\boldsymbol{e}_{n-1}\boldsymbol{e}_{n-1}^{\mathrm{T}})\mathbf{u}}{\mathbf{u}^{\mathrm{T}}\mathbf{S}\mathbf{u}} \\ &\quad= \frac{E_1u_1^2 + E_2(u_1 - u_2)^2 + \cdots + E_{n-1}(u_{n-2} - u_{n-1})^2 + E_nu_{n-1}^2}{E_1u_1^2 + E_2(u_1^2 + u_2^2) + \cdots + E_{n-1}(u_{n-2}^2 + u_{n-1}^2) + E_nu_{n-1}^2} \\ &\quad= \frac{(E_1 + E_2)u_1^2 + \cdots + (E_{n-1} + E_n)u_{n-1}^2 - 2(E_2u_1u_2 + \cdots + E_{n-1}u_{n-2}u_{n-1})}{(E_1 + E_2)u_1^2 + \cdots + (E_{n-1} + E_n)u_{n-1}^2} \\ &\quad= 1 - \frac{2(E_2u_1u_2 + \cdots + E_{n-1}u_{n-2}u_{n-1})}{(E_1 + E_2)u_1^2 + \cdots + (E_{n-1} + E_n)u_{n-1}^2} \\ &\quad= 1 - \frac{2(E_2u_1u_2 + \cdots + E_{n-1}u_{n-2}u_{n-1})}{E_1u_1^2 + E_2(u_1^2 + u_2^2) + \cdots + E_{n-1}(u_{n-2}^2 + u_{n-1}^2) + E_nu_{n-1}^2}\end{aligned} \tag{41}$$

DOI: 10.1002/nme

It is easy to see that the maximum of (41) is bounded by 2. The condition number therefore depends mainly on the minimum of (3). We consider three examples.

The first example considers the case of constant modulus of elasticity. The second example demonstrates that the case of a bar with variable modulus (solid bar with a 'hole') leads to about the same condition number after scaling as the case of a bar with constant modulus (homogeneous). The third example shows that for the hypothetical case of a 1D bar with two non-adjacent 'holes' scaling cannot remove the actual singularity.

Example 1 (*Constant modulus*)
For a constant modulus of elasticity, (41) leads to

$$\frac{\mathbf{u}^{\mathrm{T}}(E\mathbf{D}_1^{\mathrm{T}}\mathbf{D}_1 + E\boldsymbol{e}_{n-1}\boldsymbol{e}_{n-1}^{\mathrm{T}})\mathbf{u}}{\mathbf{u}^{\mathrm{T}}\mathbf{S}\mathbf{u}} = 1 - \frac{2u_1u_2 + \cdots + 2u_{n-2}u_{n-1}}{u_1^2 + (u_1^2 + u_2^2) + \cdots + (u_{n-2}^2 + u_{n-1}^2) + u_{n-1}^2} \tag{42}$$

The minimum for (42) is obtained for $u_i = \sin(\pi i h)$ which gives $u_{i-1}u_i \approx u_i^2$ and minimizes the influence of the terms Eu_1^2 and Eu_{n-1}^2. This leads to a condition number for the preconditioned system of $O(h^{-2})$.

Example 2 (*Variable modulus*)
Now consider a problem with a 'hole' at the end of the bar; $E_i = 1$ for $i = 1, \ldots, n-5$, where $n \gg 5$, and $E_i = \varepsilon$ (small) for the remaining elements. The minimum for (41) is obtained for a vector u such that $|u_i| = O(1)$ for $i = 1, \ldots n-5$ and $|u_i| = O(\varepsilon)$ for the remaining elements. After substituting for the E_i in (41) and dropping the u_i terms that are $O(\varepsilon)$, we need to minimize the following expression:

$$1 - \frac{2u_1u_2 + \cdots + 2u_{n-6}u_{n-5}}{u_1^2 + (u_1^2 + u_2^2) + \cdots + (u_{n-6}^2 + u_{n-5}^2) + \varepsilon u_{n-5}^2} \tag{43}$$

Comparing (43) with (42), we see that this minimization problem is essentially the same as the one for the constant modulus (with a few terms of small magnitude dropped). Therefore, the resulting condition number will be about the same.

Example 3 (*Hypothetical case*)
Finally, consider a hypothetical problem of a 1D bar with two non-adjacent 'holes'. Let $n = 5$, and let $E_1 = E_3 = E_5 = 1$ and $E_2 = E_4 = \varepsilon$. Now taking $u_1 = u_4 = 0$ and $u_2 = u_3 = 1$ in (41) gives

$$\begin{aligned}\min_{\mathbf{u}\neq 0} \frac{\mathbf{u}^{\mathrm{T}}(\mathbf{D}_1^{\mathrm{T}}\boldsymbol{\Omega}\mathbf{D}_1 + E_n\boldsymbol{e}_{n-1}\boldsymbol{e}_{n-1}^{\mathrm{T}})\mathbf{u}}{\mathbf{u}^{\mathrm{T}}\mathbf{S}\mathbf{u}} &\leqslant 1 - \frac{2E_3u_2u_3}{E_2u_2^2 + E_3(u_2^2 + u_3^2) + E_4u_3^2} \\ &= 1 - \frac{2}{\varepsilon + 2 + \varepsilon} = \frac{\varepsilon}{1+\varepsilon}\end{aligned} \tag{44}$$

DOI: 10.1002/nme

So, (44) can be made arbitrarily small, and therefore the condition number $\kappa(\mathbf{D}_1^{\mathrm{T}}\mathbf{\Omega}\mathbf{D}_1 + E_n \boldsymbol{e}_{n-1}\boldsymbol{e}_{n-1}^{\mathrm{T}})$ can be made arbitrarily large.

6. SOME IMPLEMENTATION ISSUES

We have developed our C/C++ code in an object-oriented fashion, since we intend to make the RMINRES solver available as public domain software. This makes it easy to integrate the software in other packages, and also facilitates maintenance and extension, while retaining high efficiency. We store sparse matrices in compressed sparse row format (CSR). The (column) vectors in $\boldsymbol{U}$, $\boldsymbol{C}$, $\boldsymbol{V}_j$, and so on, are stored as 1D arrays linked by a 1D array of pointers. The memory required by the system matrix and the incomplete Cholesky factor is linear in the number of unknowns, n, since the number of non-zero coefficients per row is never greater than 81. The RMINRES method requires only matrix–vector multiplications, dot products, vector updates, forward and backward solves with the incomplete Cholesky factors, and the incomplete Cholesky decomposition itself. All of these operations have linear computational cost.

The small matrices, e.g. the matrices in (22), are all stored as dense matrices in column-wise ordering (F77 format), so that dense matrix routines from LAPACK and BLAS can be used. For the generalized eigenvalue problem (20) we use the LAPACK routine DSYGV, and for QR decompositions we use the LAPACK routine DGEQRF. We use the BLAS routines DROTG, to compute Given's rotations, and DROT, to apply Given's rotations. We use the BLAS routine DGEMM for (small) dense matrix–matrix products. For convenience we use the CLAPACK library, which provides an interface for C programs to LAPACK. However, since CLAPACK routines call the corresponding LAPACK routines, we still need to adhere to F77 storage formats. The computational cost of the work with these small matrices is negligible. Finally, we note that the computational cost is significantly reduced by taking the simplifications in (24)–(30) into account.

7. NUMERICAL RESULTS

We demonstrate the performance of the MINRES iterative solver with the recycling and preconditioning techniques discussed in Sections 4 and 5 on a large 3D design problem. We also provide some analysis for the selection of the linear solver parameters.

Figure 6 shows our model problem. The volume fraction is 50%, and the radius of the filter is 10% of Y. We use continuation on the density penalization, ranging from 1 to 3 with increments of 0.5. As stated before, we use the OC method as the optimization algorithm. The convergence criterion is that the maximum change in the design variables is less than 0.01 or the relative change of the compliance is less than 10^{-6}. For all the iterative solvers discussed, we always use the solution of the previous system as the initial guess of the next system to reduce the initial error. We test three discretizations of increasing mesh resolution. Exploiting the symmetry of the problem, we model and simulate only half of the domain. For each test case, Table I lists the mesh size (for half of the domain), the number of unknowns, the overall solution time, the number of optimization steps, and the parameters used for the recycling MINRES solver. Figure 7 shows the final topologies.

First, we analyse the convergence properties of RMINRES for several parameter choices on the medium size ($84 \times 28 \times 14$) mesh. The number of optimization steps to compute the optimal design is 139, requiring the solution of 139 linear systems.

DOI: 10.1002/nme

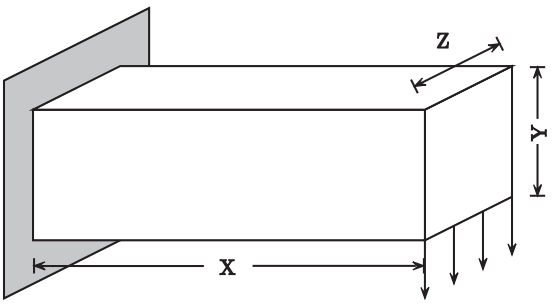

Figure 6. Design problem: finding optimal material distribution in a hexahedron with the left end fixed and a distributed load applied on the right bottom edge ($X{:}Y{:}Z = 3{:}1{:}1$).

Table I. Three discretizations used for the example in Figure 6.

Problem	Mesh size	No. of unknowns (in simulation)	Solution time (h)	Optimization steps	Iterative solver
Small	$36 \times 12 \times 6$	9360	0.1	142	RMINRES(100, 10)
Medium	$84 \times 28 \times 14$	107 184	2.4	139	RMINRES(100, 10)
Large	$180 \times 60 \times 30$	1 010 160	45.7	130	RMINRES(200, 10)

As mentioned in the first section, we can vary the tolerance for the iterative solver, since less accurate finite element solutions are sufficient at the beginning of the topology optimization process. So, we can also apply a continuation approach to the tolerance of the linear solver, which reduces the number of iterations in the early phase of the optimization process, as shown in Figure 8. The jumps in the iteration counts correspond to the steps where the tolerance of the linear solver τ is decreased or the penalization parameter p is increased. We start with $\tau = 10^{-4}$ and $p = 1$. We decrease τ by a factor of $1/10$ and increase p by 0.5 every time the maximum change of the design variables drops below 0.1. And we stop updating them when $\tau = 10^{-10}$ and $p = 3$. Finally, we note that allowing a higher tolerance for the linear solver in the beginning of the optimization process did not affect the number of optimization steps required.

Next, we consider the parameters that govern the recycling for the MINRES solver, namely k, the dimension of the subspace that is recycled from one linear system to the next, and s, the maximum dimension of the Krylov subspace kept to periodically update the approximate invariant subspace that will be recycled. We carry out two sets of experiments to analyse the effects of varying these two parameters. To make a fair comparison, we use the solution from the previous system as the initial guess of the next system and we use continuation on the tolerance for both RMINRES and MINRES.

In the first set of experiments, we fix $k = 10$ and vary s. Figure 9 compares the number of iterations and computation time for each linear system, for several choices of s. In the first few optimization steps, the topology changes significantly, and the effect of recycling is modest. After this, the recycling approach greatly reduces the number of iterations to solve each linear system. We see that if we keep a larger Krylov subspace to update the approximate invariant subspace, the recycling becomes more effective in reducing iteration counts. Since the dimension of the

DOI: 10.1002/nme

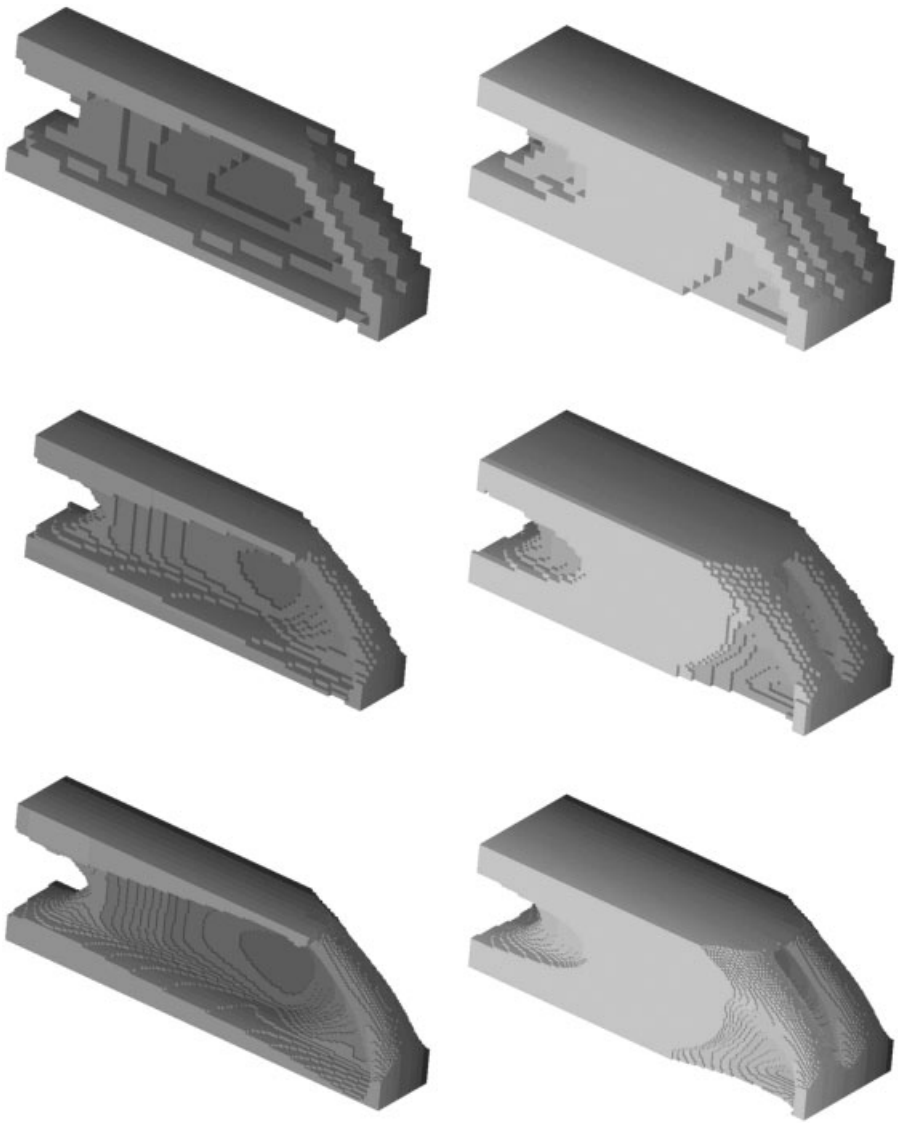

Figure 7. Final topologies of the model problem. Left: half domain; right: full domain: top row: small mesh; middle row: medium size mesh; and bottom row: large mesh.

recycle space itself does not change, this suggests that we obtain a more accurate approximation to the invariant subspace this way. This reduction in iterations significantly reduces the computation time for RMINRES, in spite of the computational overhead from the orthogonalizations against the recycle space and from the updates of the recycle space. Towards the end of the optimization process, recycling leads to a 40% reduction in computation time and a 50% reduction in iterations.

DOI: 10.1002/nme

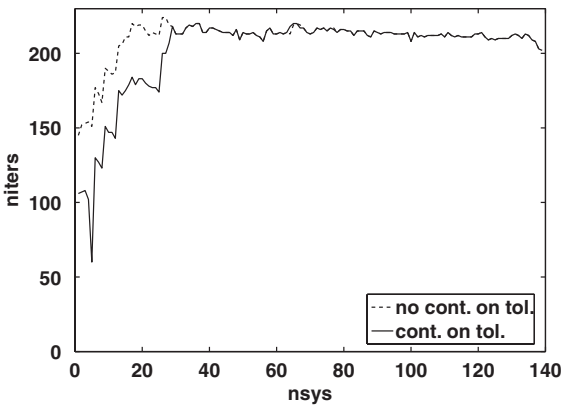

Figure 8. Reduction in the number of iterations using a relaxed tolerance for the linear solver (MINRES without recycling). The jumps in the iteration counts over the first 30 iterations are caused by continuation on the solver tolerance and the penalization parameter.

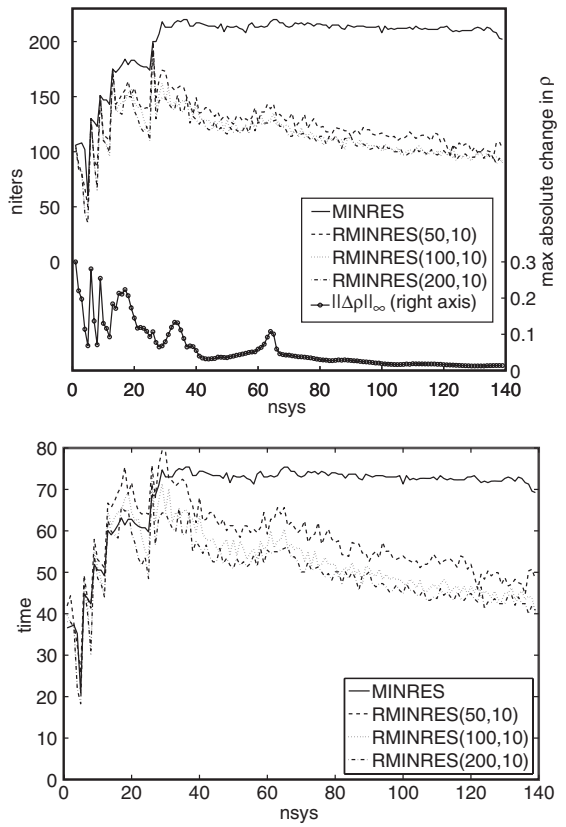

Figure 9. Number of iterations (niters) and time (s) of RMINRES(s, k) with fixed $k = 10$ and varying s.

DOI: 10.1002/nme

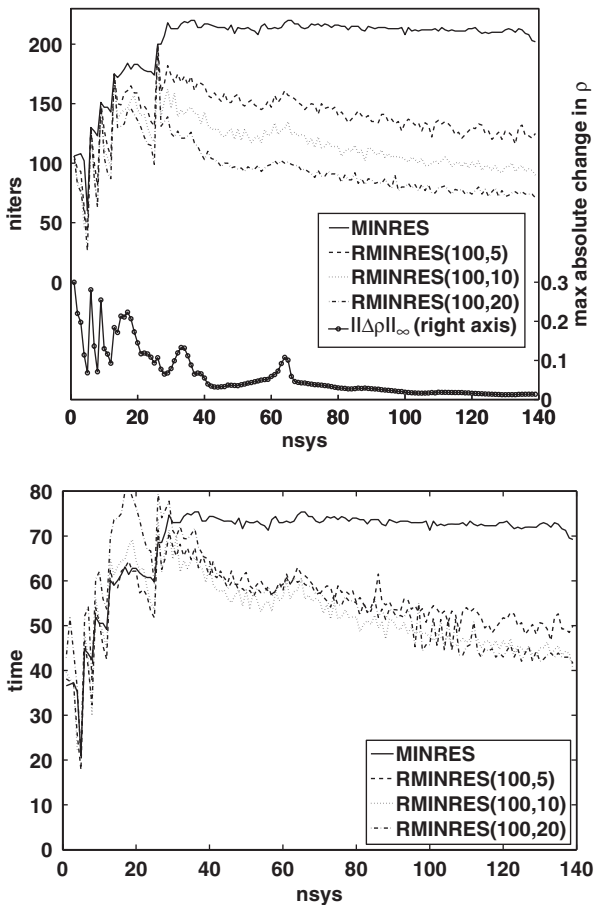

Figure 10. Number of iterations (niters) and time (s) of RMINRES(s, k) with varying k and fixed $s = 100$.

Notice that increasing s beyond 100 has limited effect since RMINRES rarely takes more than 100 iterations for this problem. However, for harder problems, e.g. for finer meshes, the solver may not converge so fast. In that case, larger values for s can be helpful. Note that increasing s does not increase the computational cost of RMINRES. The only limit on s is the memory size.

In the second set of experiments, we fix $s = 100$ and vary k. The parameter k affects both the computational cost per iteration, specifically the number of orthogonalizations and the cost of subspace selection, and the total number of iterations for the solver. There is a trade-off between these two factors, and in Figure 10 we compare the number of iterations and computational time for several values of k. Increasing k leads to a significant improvement in the convergence rate; towards the end we obtain a factor 3 reduction in the number of iterations. Timewise, we obtain a 40% improvement. We also see that the computation time is not overly sensitive to the choice of k.

Finally, we compare a state-of-the-art sparse direct solver with our iterative method, including recycling and the continuation on the solver tolerance, the initial scaling, and the preconditioner.

DOI: 10.1002/nme

Table II. Run time comparison (s) of direct and iterative solvers. The direct solver is the multifrontal, supernodal Cholesky factorization in TAUCS; the iterative solver is RMINRES with rescaling and incomplete Cholesky preconditioner and the continuation on the solver tolerance. These timings were obtained on a PC with an AMD Opteron ^{TM}252 2.6 GHz 64-bit processor, 8 GB RAM of memory, and the SuSE Linux system.

Problem size	No. of unknowns in simulation	Direct solver time (s)			Iterative solver time (s)			
		Decompose	Solve	Total	min	max	Average	IC(0)
Small	9360	0.95	0.01	0.96	0.94	2.25	1.66	0.02
Medium	107 184	179.0	0.3	179.3	21.2	71.9	50.5	3.6
Large	1 010 160	21 241	4904	26 154	254	1546	1170	26.3

We use the multifrontal, supernodal Cholesky factorization from the TAUCS package [34]. We ran the comparison on a PC with an AMD Opteron ^{TM}252 2.6 GHz 64-bit processor, 8 GB RAM of memory, and the SuSE Linux system. Table II lists the computation times of both the direct solver and the iterative solver for a single linear system, for several problem sizes. For the direct solver, the data include the time for the Cholesky decomposition, and for the forward and backward substitution to solve one system. For the iterative solver, to make a fair comparison the data include the minimum, maximum, and average solver time taken over all linear systems, and separately the time to compute the incomplete Cholesky preconditioner for one system. The time to compute the preconditioner is very small compared with the solver time. We note that the run time of the direct solver seems to scale worse than quadratic, whereas the iterative solver scales slightly worse than linear. Since the matrix changes at every optimization step, we cannot reuse the Cholesky factors of the direct solver, and a new factorization must be computed every optimization step. Finally, for the large system, the time for the forward and backward substitution of the direct solver alone is significantly larger than the (average) time for the iterative solver. Therefore, for large systems, our iterative solver is much faster than direct solvers even for additional right-hand sides for the same linear system.

8. CONCLUSIONS

In this paper, we investigate iterative solvers for the equilibrium equations in topology optimization. We address the main problem, the extreme ill-conditioning, by two approaches. First, we rescale the system to reduce the ill-conditioning due to the variation in the density. Second, we use an incomplete Cholesky preconditioner for the resulting linear system. In general, the rescaling improves the accuracy of the solution as well as the convergence rate. The Cholesky preconditioner improves convergence rate, but in general has no effect on the accuracy. Exploiting the slowly changing nature of the linear systems arising in topology optimization, the RMINRES method, which recycles selected subspaces, leads to a further significant reduction in iterations and run time.

We benchmark our methods for a design problem on a sequence of increasingly finer meshes. The largest problem has more than a million unknowns, resulting in a smooth solution. With the proposed methods, we can solve these problems on a single PC in a reasonably short time.

DOI: 10.1002/nme

APPENDIX

List of symbols

n	the size of the linear system
m	the current number of iterations of the iterative method
k	the dimension of the subspace to be recycled
s	the maximum number of Lanczos vectors kept for updating the recycle space
j	the index of the update cycle
$a(u, v)$	bilinear form of the 1D idealized elasticity problem
c	compliance
E, E_i	modulus of elasticity
$\boldsymbol{f}$	load vector
$\boldsymbol{K}, \boldsymbol{K}^{(i)}$	global stiffness matrix
$\boldsymbol{K}_0$	element stiffness matrix
K_{ij}	(i, j) coefficient of $\boldsymbol{K}$
$\mathbf{k}_j$	the jth column of $\boldsymbol{K}$
p	penalization parameter
$\boldsymbol{u}, \boldsymbol{u}^{(i)}$	displacement vector
V	total volume
$\boldsymbol{\rho}, \boldsymbol{\rho}^{(i)}$	density distribution (the design variables of topology optimization problems)
ρ_0	positive lower bound on the density to avoid singularity
$\boldsymbol{\Omega}$	diagonal matrix of material constants
$\boldsymbol{V}_m$	first m vectors of the Krylov subspace, $\boldsymbol{v}_1, \boldsymbol{v}_2, \ldots, \boldsymbol{v}_m$
$\boldsymbol{V}_j, \underline{\boldsymbol{V}}_j$	the Krylov subspace generated in jth cycle, $\boldsymbol{v}_{(j-1)s+1}, \ldots, \boldsymbol{v}_{js}$, and its extended version, $\boldsymbol{v}_{(j-1)s}, \ldots, \boldsymbol{v}_{js+1}$
$\boldsymbol{U}, \boldsymbol{U}_j$	basis for the recycle space
$\boldsymbol{C}, \boldsymbol{C}_j$	orthogonal basis of $\boldsymbol{KU}$
$\underline{\boldsymbol{H}}_m, \boldsymbol{H}_m$	$(m + 1) \times m$ Hessenberg matrix generated in the Arnoldi iteration, and the first m rows of $\underline{\boldsymbol{H}}_m$
$\boldsymbol{r}_0, \boldsymbol{r}_m$	residual
$\underline{\boldsymbol{T}}_j, \overline{\underline{\boldsymbol{T}}}_j$	tridiagonal matrix generated by Lanczos iteration and its extended version with an additional top row
$\kappa(\cdot)$	condition number of a matrix
$(\theta, \boldsymbol{w})$	an harmonic Ritz pair of $\boldsymbol{K}$ with Ritz value θ and Ritz vector $\boldsymbol{w}$

Abbreviations

GMRES	generalized minimum residual method [25]
MINRES	minimum residual method [8]
GCRODR	generalized conjugate residual method with inner orthogonalization and deflated restarting [4]
GCRO	generalized conjugate residual method with inner orthogonalization [28]
GCROT	generalized conjugate residual method with inner orthogonalization and optimal truncation [29]
IC(0)	incomplete Cholesky decomposition with zero fill-in [32]

DOI: 10.1002/nme

REFERENCES

1. Borrvall T, Petersson J. Large-scale topology optimization in 3D using parallel computing. *Computer Methods in Applied Mechanics and Engineering* 2001; **190**:6201–6229.
2. Vemaganti K, Lawrence WE. Parallel methods for optimality criteria-based topology optimization. *Computer Methods in Applied Mechanics and Engineering* 2005; **194**:3637–3667.
3. Kim TS, Kim JE, Kim YY. Parallelized structural topology optimization for eigenvalue problems. *International Journal of Solids and Structures* 2004; **41**:2623–2641.
4. Parks ML, de Sturler E, Mackey G, Johnson DD, Maiti S. Recycling Krylov subspaces for sequences of linear systems. *Technical Report UIUC DCS-R-2004-2421*, available from http://www-faculty.cs.uiuc.edu/~sturler, 2004, accepted for publication in SISC.
5. Kilmer ME, de Sturler E. Recycling subspace information for diffuse optical tomography. *SIAM Journal on Scientific Computing* 2006; **27**(6):2140–2166.
6. Gersborg-Hansen A, Sigmund O, Haber RB. Topology optimization of channel flow problems. *Journal for Structural and Multidisciplinary Optimization* 2005; **30**:181–192.
7. Sigmund O, Jensen JS. Systematic design of phononic band gap materials and structures. *Philosophical Transactions of the Royal Society London*, *Series A* (*Mathematical*, *Physical and Engineering Sciences*) 2003; **361**:1001–1019.
8. Paige CC, Saunders MA. Solution of sparse indefinite systems of linear equations. *SIAM Journal on Numerical Analysis* 1975; **12**:617–629.
9. Guest JK, Prévost JH. Topology optimization of creeping fluid flows using a Darcy–Stokes finite element. *International Journal for Numerical Methods in Engineering* 2006; **66**:461–484.
10. Bendsøe MP, Sigmund O. *Topology Optimization*: *Theory*, *Methods and Applications*. Springer: Berlin, Germany, 2003.
11. Sigmund O. Topology optimization: a tool for the tailoring of structures and materials. *Transactions of the Royal Society*: *Science into the next Millennium* (*Issue III*, *Mathematics*, *Physics and Engineering*) 2000; **358**(1765): 211–228 (Special issue).
12. Rozvany GIN. Aims, scope, methods, history and unified terminology of computer-aided topology optimization in structural mechanics. *Journal for Structural and Multidisciplinary Optimization* 2001; **21**:90–108.
13. Mackerle J. Topology and shape optimization of structures using FEM and BEM: a bibliography (1999–2001). *Finite Elements in Analysis and Design* 2003; **39**:243–253.
14. Paulino GH, Page III RC, Silva ECN. A Java-based topology optimization program with web access: nodal design variable approach. In *Proceedings of the 6th World Congress on Structural and Multidisciplinary Optimization* (*WCSMO6*), International Society for Structural and Multidisciplinary Optimization, Rio de Janeiro, Brazil, May 30–June 3, 2005.
15. Stump FV, Paulino GH, Silva ECN. Material distribution design of functionally graded rotating disks with stress constraint. In *Proceedings of the 6th World Congress on Structural and Multidisciplinary Optimization* (*WCSMO6*), International Society for Structural and Multidisciplinary Optimization, Rio de Janeiro, Brazil, May 30–June 3 2005.
16. Bendsøe MP. Optimal shape design as a material distribution problem. *Structural Optimization* 1989; **1**:193–202.
17. Bendsøe MP, Sigmund O. Material interpolation schemes in topology optimization. *Archives of Applied Mechanics* 1999; **69**(9–10):635–654.
18. Matsui K, Terada K. Continuous approximation of material distribution for topology optimization. *International Journal for Numerical Methods in Engineering* 2004; **59**:1925–1944.
19. Paulino GH, Silva ECN. Design of functionally graded structures using topology optimization. *Materials Science Forum* 2005; **492–493**:435–440.
20. Jog CS, Haber RB. Stability of finite element models for distributed-parameter optimization and topology design. *Computer Methods in Applied Mechanics and Engineering* 1996; **130**:203–226.
21. Rahmatalla S, Swan CC. A Q4/Q4 continuum structural optimization implementation. *Journal for Structural and Multidisciplinary Optimization* 2004; **27**:130–135.
22. Sigmund O. On the design of compliant mechanisms using topology optimization. *Mechanics of Structures and Machines* 1997; **25**:495–526.
23. Rozvany GIN. *Structural Design via Optimality Criteria*. Kluwer Academic Publishers: Dordrecht, 1989.
24. Svanberg K. The method of moving asymptotes: a new method for structural optimization. *International Journal for Numerical Methods in Engineering* 1987; **24**(2):359–373.
25. Saad Y, Schultz MH. GMRES: a generalized minimal residual algorithm for solving nonsymmetric linear systems. *SIAM Journal on Scientific and Statistical Computing* 1986; **7**(3):856–869.

DOI: 10.1002/nme

26. Arnoldi WE. The principle of minimized iterations in the solution of the matrix eigenvalue problem. *Quarterly of Applied Mathematics* 1951; **9**:17–29.
27. Saad Y. *Iterative Methods for Sparse Linear Systems* (2nd edn). SIAM: Philadelphia, PA, U.S.A., 2003.
28. de Sturler E. Nested Krylov methods based on GCR. *Journal of Computational and Applied Mathematics* 1996; **67**:15–41.
29. de Sturler E. Truncation strategies for optimal Krylov subspace methods. *SIAM Journal on Numerical Analysis* 1999; **36**:864–889. Electronically available from http://epubs.siam.org
30. van de Vorst HA. *Iterative Krylov Methods for Large Linear Systems*. Cambridge Monographs on Applied and Computational Mathematics. Cambridge University Press: Cambridge, MA, 2003.
31. Greenbaum A. *Iterative Methods for Solving Linear Systems*. SIAM: Philadelphia, PA, 1997.
32. Meijerink J, van der Vorst HA. An iterative solution method for linear equations systems of which the coefficient matrix is a symmetric M-matrix. *Mathematics of Computation* 1977; **31**:148–162.
33. Axelsson O, Barker VA. *Finite Element Solution of Boundary Value Problems. Classics in Applied Mathematics*, vol. 35. SIAM: Philadelphia, PA, U.S.A., 2001.
34. Toledo S. *TAUCS: A Library of Sparse Linear Solvers*. School of Computer Science, Tel-Aviv University, September 2003.

DOI: 10.1002/nme