

1

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

3

Shun Wang^{1,‡}, Eric de Sturler^{2,§} and Glaucio H. Paulino^{3,*,†}

5

¹*Department of Computer Science, University of Illinois at Urbana-Champaign,
Urbana, IL 61801, U.S.A.*

7

²*Department of Mathematics, Virginia Tech, Blacksburg, VA 24061, U.S.A.*

9

³*Department of Civil and Environmental Engineering, University of Illinois at Urbana-Champaign,
Urbana, IL 61801, U.S.A.*

SUMMARY

11 The computational bottleneck of topology optimization is the solution of a large number of linear systems
13 arising in the finite element analysis. We propose fast iterative solvers for large three-dimensional topology
15 optimization problems to address this problem. Since the linear systems in the sequence of optimization
17 steps change slowly from one step to the next, we can significantly reduce the number of iterations and the
19 runtime of the linear solver by recycling selected search spaces from previous linear systems. In addition,
21 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 January 2006; Accepted 5 May 2006

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

1. INTRODUCTION

25 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

Copyright © 2006 John Wiley & Sons, Ltd.

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

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

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

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

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

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

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

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

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

3 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
 5 improvements our solvers achieve. In the last section, we provide the conclusions.

2. TOPOLOGY OPTIMIZATION

7 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
 9 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
 11 (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
 13 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
 15 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:

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

17 where c is the compliance, $\mathbf{K}(\rho)$ is the stiffness matrix as a function of the density distribution ρ , \mathbf{u}
 and \mathbf{f} are the displacement vector and load vector, ρ_0 is a chosen, small, positive lower bound for
 19 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
 21 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
 23 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
 25 and the loading, and initialize the density distribution ρ . Then, we start the optimization loop. We
 need a linear solver to solve the equilibrium equations $\mathbf{K}\mathbf{u} = \mathbf{f}$ in the finite element analysis. In the
 27 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
 29 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
 31 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
 33 with the FEA in topology optimization, the choice of the optimization method is less relevant

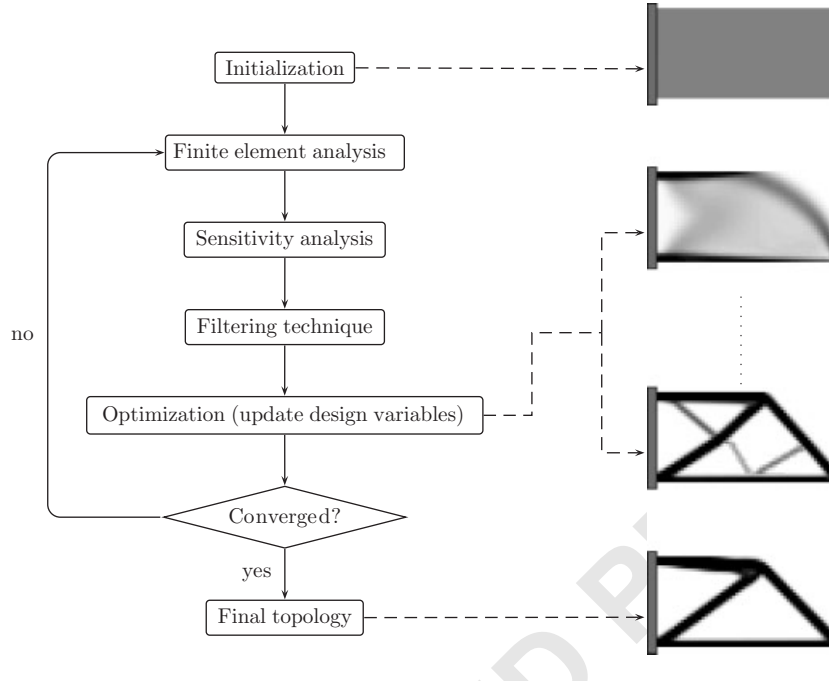


Figure 1. The general flow of computations for topology design.

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

3. KRYLOV SUBSPACE RECYCLING

5 Consider the linear system $\mathbf{K}\mathbf{u} = \mathbf{f}$ and an initial guess \mathbf{u}_0 . A Krylov subspace method, such as the
 6 generalized minimum residual method (GMRES) [25], builds the Krylov subspace, $\text{span}\{\mathbf{r}_0, \mathbf{K}\mathbf{r}_0,$
 7 $\mathbf{K}^2\mathbf{r}_0, \dots, \mathbf{K}^{m-1}\mathbf{r}_0\}$, where $\mathbf{r}_0 = \mathbf{f} - \mathbf{K}\mathbf{u}_0$, and computes the optimal solution over that subspace. We
 use the Arnoldi recurrence [26] to obtain an orthonormal basis of the Krylov subspace:

$$\mathbf{v}_1 = \mathbf{r}_0 / \|\mathbf{r}_0\|_2 \tag{3}$$

$$h_{i,i+1}\mathbf{v}_{i+1} = \mathbf{K}\mathbf{v}_i - h_{i,i}\mathbf{v}_i - h_{i-1,i}\mathbf{v}_{i-1} - \dots - h_{1,i}\mathbf{v}_1$$

which in matrix form is written as

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

where the columns of \mathbf{V}_m are $\mathbf{v}_1, \dots, \mathbf{v}_m$; the columns of \mathbf{V}_{m+1} are $\mathbf{v}_1, \dots, \mathbf{v}_{m+1}$; and \mathbf{H}_m is an
 13 $(m + 1) \times m$ upper Hessenberg matrix with coefficients $\{h_{ij}\}$.

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

$$\frac{\|\mathbf{r}_m\|_2}{\|\mathbf{r}_0\|_2} \leq \min_{p_m \in \Pi_m^{(0)}} \max_{\lambda \in \lambda(\mathbf{K})} |p_m(\lambda)| \quad (5)$$

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

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

9 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
 11 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
 13 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
 15 subspace to approximate an invariant subspace [4].

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

$$\begin{aligned} (\mathbf{I} - \mathbf{C}\mathbf{C}^T)\mathbf{K}\mathbf{V}_m &= \mathbf{V}_{m+1}\mathbf{H}_m \Leftrightarrow \\ \mathbf{K}\mathbf{V}_m &= \mathbf{C}\mathbf{C}^T\mathbf{K}\mathbf{V}_m + \mathbf{V}_{m+1}\mathbf{H}_m \end{aligned} \quad (6)$$

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

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

23 where $\beta = \|(\mathbf{I} - \mathbf{C}\mathbf{C}^T)\mathbf{r}_0\|_2$ and $\mathbf{B}_m = \mathbf{C}^T\mathbf{K}\mathbf{V}_m$. This least squares problem can be solved using the
 25 QR decomposition of \mathbf{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].

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

7 4. RECYCLING MINRES

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

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

18 We can use the matrices U and C defining the recycle space, obtained from solving previous linear
 19 systems, in the same way as in GCRODR. This leads to the same recurrence as in (6). However, the
 20 symmetry of K implies the symmetry of H_m , the leading $m \times m$ submatrix of \underline{H}_m .[¶] Since \underline{H}_m
 21 is also an upper Hessenberg matrix, this gives a tridiagonal \underline{H}_m , which we will denote as \underline{T}_m from now
 22 on. So, including the recycle space into the Krylov subspace does not affect the Lanczos recurrence
 23 of MINRES. Now we explain how we adapt the MINRES method as described in Reference [30,
 24 p. 84–86] or [31, p. 41–44] to include the recycle space. Similarly as in GCRODR, we need to
 25 compute the vector $\varepsilon_m = Uz_m + V_m y_m$, such that $u_m = u_0 + \varepsilon_m$ minimizes $\|r_m\|_2$. For symmetric
 26 K , (7) becomes

27
$$\|r_m\|_2 = \left\| \begin{pmatrix} C^T r_0 \\ \beta e_1 \end{pmatrix} - \begin{bmatrix} I & B_m \\ 0 & \underline{T}_m \end{bmatrix} \begin{pmatrix} z_m \\ y_m \end{pmatrix} \right\|_2 \quad (8)$$

28 where $\beta = \|(I - CC^T)r_0\|_2$ and $B_m = C^T K V_m$. The QR decomposition of \underline{T}_m gives

29
$$\underline{T}_m = \hat{G}_m^T \underline{X}_m \quad (9)$$

30 where \hat{G}_m is an orthogonal matrix of size $(m + 1) \times (m + 1)$, and \underline{X}_m is an upper triangular matrix
 31 of size $(m + 1) \times m$ with bandwidth 3. \hat{G} is the product of a series of orthogonal matrices defining
 32 plane rotations, also called Given's rotations, $\hat{G}_m = G_m \cdots G_2 G_1$ (see Algorithm 1). Let \underline{X}_m be the
 33 leading $m \times m$ submatrix of \underline{X}_m , and $\{x_{ij}\}$ and $\{t_{ij}\}$ be the coefficients of \underline{X}_m and \underline{T}_m , respectively.
 34 The solution of the least squares problem (8) is then

35
$$y_m = X_m^{-1} \hat{G}_m \beta e_1, \quad z_m = C^T r_0 - B_m y_m \quad (10)$$

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

1

This leads to

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

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

7
$$\hat{\mathbf{B}}_m = \mathbf{U}\mathbf{B}_m, \quad \tilde{\mathbf{B}}_m = \hat{\mathbf{B}}_m\mathbf{X}_m^{-1}, \quad \tilde{\mathbf{V}}_m = \mathbf{V}_m\mathbf{X}_m^{-1}, \quad \tilde{\mathbf{y}}_m = \mathbf{X}_m\mathbf{y}_m \tag{12}$$

Then

9
$$\tilde{\mathbf{y}}_m = \hat{\mathbf{G}}_m\beta\mathbf{e}_1 = \mathbf{G}_m\mathbf{G}_{m-1}\cdots\mathbf{G}_1\beta\mathbf{e}_1 = \mathbf{G}_m\tilde{\mathbf{y}}_{m-1}$$

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

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

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

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

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

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

15 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
 17 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
 19 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.

Algorithm 1 Modified MINRES

```

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

```

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

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

$$(\mathbf{I} - \mathbf{C}\mathbf{C}^T)\mathbf{K}\mathbf{V}_j = \overline{\mathbf{V}}_j\overline{\mathbf{T}}_j \tag{15}$$

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

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

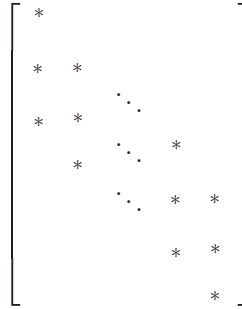


Figure 2. Non-zero pattern of \bar{T}_j .

1 The modified Lanczos recurrence that includes the orthogonalization against C gives

$$K[U \ U_{j-1} \ V_j] = [C \ C_{j-1} \ \bar{V}_j] \begin{bmatrix} I & 0 & B_j \\ 0 & I & 0 \\ 0 & 0 & \bar{T}_j \end{bmatrix} \quad (16)$$

3 where $B_j = C^T K V_j$ has been computed in the course of the iteration (see (15)). Now, we have
 5 several options for selecting a new matrix U_j for recycling. The first option is to compute the
 7 harmonic Ritz vectors of K with respect to $\text{range}([U \ U_{j-1} \ V_j])$. The second option is similar to
 9 option 1, but we drop the U components from U_j if the linear solver has not converged. This is
 11 possible since 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 U_1 from $\text{range}([U \ V_1])$ after the first cycle
 and U_j from $\text{range}([U_{j-1} \ V_j])$ after the j th cycle. In the last approach, the reappearance of 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

$$W_j = [U_{j-1} \ V_j], \quad \tilde{W}_j = [C \ C_{j-1} \ \bar{V}_j], \quad \tilde{H}_j = \begin{bmatrix} 0 & B_j \\ I & 0 \\ 0 & \bar{T}_j \end{bmatrix} \quad (17)$$

Then, (16) gives

$$K W_j = \tilde{W}_j \tilde{H}_j \quad (18)$$

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

$$K w - \theta w \perp \text{range}(K W_j) \quad (19)$$

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

$$\tilde{H}_j^T \tilde{W}_j^T \tilde{W}_j \tilde{H}_j p = \theta \tilde{H}_j^T \tilde{W}_j^T W_j p \quad (20)$$

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

$$\tilde{\mathbf{W}}_j = \hat{\mathbf{W}}_j \mathbf{F}_j \tag{21}$$

7 and of $\mathbf{F}_j \tilde{\mathbf{H}}_j \mathbf{P}_j$,

$$\mathbf{F}_j \tilde{\mathbf{H}}_j \mathbf{P}_j = \mathbf{Q}_j \mathbf{R}_j \tag{22}$$

9 Next, we set

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

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

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

$$\tilde{\mathbf{W}}_j^T \tilde{\mathbf{W}}_j = \begin{bmatrix} \mathbf{I} & \mathbf{C}^T \mathbf{C}_{j-1} & \mathbf{0} \\ \mathbf{C}_{j-1}^T \mathbf{C} & \mathbf{I} & \mathbf{C}_{j-1}^T \bar{\mathbf{V}}_j \\ \mathbf{0} & \bar{\mathbf{V}}_j^T \mathbf{C}_{j-1} & \mathbf{I} \end{bmatrix} \tag{24}$$

$$\tilde{\mathbf{W}}_j^T \mathbf{W}_j = \begin{bmatrix} \mathbf{C}^T \mathbf{U}_{j-1} & \mathbf{0} \\ \mathbf{C}_{j-1}^T \mathbf{U}_{j-1} & \mathbf{C}_{j-1}^T \mathbf{V}_j \\ \bar{\mathbf{V}}_j^T \mathbf{U}_{j-1} & \bar{\mathbf{I}} \end{bmatrix} \tag{25}$$

17 where $\bar{\mathbf{I}}$ 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.

$$\mathbf{C}^T \mathbf{C}_{j-1} = \mathbf{C}^T \tilde{\mathbf{W}}_{j-1} \hat{\mathbf{Q}}_{j-1} = [\mathbf{I} \quad \mathbf{C}^T \mathbf{C}_{j-2} \quad \mathbf{0}] \hat{\mathbf{Q}}_{j-1} \tag{26}$$

$$\bar{\mathbf{V}}_j^T \mathbf{C}_{j-1} = \bar{\mathbf{V}}_j^T \tilde{\mathbf{W}}_{j-1} \hat{\mathbf{Q}}_{j-1} = \begin{bmatrix} 0 & \cdots & 0 & 1 & 0 \\ 0 & \cdots & 0 & 0 & 1 \\ \vdots & \ddots & \ddots & \vdots & \vdots \\ 0 & \cdots & \cdots & 0 & 0 \end{bmatrix} \hat{\mathbf{Q}}_{j-1} \tag{27}$$

$$\mathbf{C}^T \mathbf{U}_{j-1} = \mathbf{C}^T \mathbf{W}_{j-1} \hat{\mathbf{P}}_{j-1} = [\mathbf{C}^T \mathbf{U}_{j-2} \quad \mathbf{0}] \hat{\mathbf{P}}_{j-1} \quad (28)$$

$$\mathbf{C}_{j-1}^T \mathbf{U}_{j-1} = \hat{\mathbf{Q}}_{j-1}^T (\tilde{\mathbf{W}}_{j-1}^T \mathbf{W}_{j-1}) \hat{\mathbf{P}}_{j-1} \quad (29)$$

$$\mathbf{C}_{j-1}^T \mathbf{V}_j = \hat{\mathbf{Q}}_{j-1}^T \begin{bmatrix} \mathbf{C}^T \\ \mathbf{C}_{j-2}^T \\ \underline{\mathbf{V}}_{j-1} \end{bmatrix} \mathbf{V}_j = \hat{\mathbf{Q}}_{j-1}^T \begin{bmatrix} 0 & 0 & \dots & 0 \\ 0 & 0 & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 1 & 0 & \dots & 0 \end{bmatrix} \quad (30)$$

1 From the above derivation, we can obtain $\underline{\mathbf{V}}_j^T \mathbf{C}_{j-1}$ and $\mathbf{C}_{j-1}^T \mathbf{V}_j$ simply from $\hat{\mathbf{Q}}_{j-1}$ (known from
 2 the previous cycle), and we can compute $\mathbf{C}^T \mathbf{C}_{j-1}$, $\mathbf{C}^T \mathbf{U}_{j-1}$ and $\mathbf{C}_{j-1}^T \mathbf{U}_{j-1}$ recursively with $2k^3$, $2k^3$
 3 and $2k(k+s)(3k+s)$ flops, respectively. Therefore, the computation of these submatrices is very
 4 cheap. Only $\underline{\mathbf{V}}_j^T \mathbf{U}_{j-1}$ must be computed explicitly by matrix–matrix product, which takes about
 5 $2ksn$ flops. In summary, the cost of each update of the recycle space is about $(12k^2 + 6ks + 6k + 4)n$
 6 flops, ignoring the terms that do not have a factor n . Compared with the cost of MINRES, which
 7 is mainly determined by the matrix–vector product and the forward and backward solve for the
 8 preconditioner for each iteration, the overhead of the subspace selection is modest (see the timing
 9 results in Section 7).

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

$$\begin{aligned} \mathbf{W}_1 &= \mathbf{V}_1, & \tilde{\mathbf{W}}_1 &= \underline{\mathbf{V}}_1 \\ \mathbf{W}_j &= [\mathbf{U}_{j-1} \quad \mathbf{V}_j], & \tilde{\mathbf{W}}_j &= [\mathbf{C}_{j-1} \quad \underline{\mathbf{V}}_j] \quad \text{for } j > 1 \end{aligned} \quad (31)$$

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

$$\mathbf{W}_1 = [\mathbf{U} \quad \mathbf{V}_1], \quad \tilde{\mathbf{W}}_1 = [\mathbf{C} \quad \underline{\mathbf{V}}_1] \quad (32)$$

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

5. PRECONDITIONING FOR TOPOLOGY OPTIMIZATION

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

Algorithm 2 Solve $Ku = f$ with u_0 and \hat{U} from the previous system, and s and k

```

1:  $\hat{C} \leftarrow K\hat{U}$ 
2: compute the QR decomposition of  $\hat{C}$  as  $\hat{C} = C\tilde{R}$ 
3:  $U \leftarrow \hat{U}\tilde{R}^{-1}$  « use backward substitution »
4:  $u \leftarrow u_0$ ;  $r \leftarrow f - Ku$ 
5:  $u \leftarrow u + U(C^T r)$ ;  $r \leftarrow r - C(C^T r)$  « use Modified Gram–Schmidt »
6:  $\gamma = \|r\|_2$ ;  $v_1 \leftarrow r/\gamma$ ;  $\tilde{y} \leftarrow \gamma e_1$ 
7:  $m \leftarrow 0$ ;  $j \leftarrow 0$ 
8: while  $\gamma/\|f\|_2 > \text{tol}$  do
9:    $m \leftarrow m + 1$ 
10:   $\hat{v} = Kv_m$ 
11:   $\hat{v} \leftarrow \hat{v} - C(C^T \hat{v})$ ;  $\hat{b} \leftarrow U(C^T \hat{v})$  « use Modified Gram–Schmidt »
12:   $t_{m-1,m} \leftarrow t_{m,m-1}$ ;  $\hat{v} \leftarrow \hat{v} - t_{m-1,m}\hat{v}_{m-1}$ 
13:   $t_{m,m} \leftarrow \langle \hat{v}, v_m \rangle$ ;  $\hat{v} \leftarrow \hat{v} - t_{m,m}v_m$ 
14:   $t_{m+1,m} \leftarrow \|\hat{v}\|_2$ ;  $v_{m+1} \leftarrow \hat{v}/t_{m+1,m}$ 
15:   $X_{:,m} \leftarrow G_{m-1}G_{m-2}\dots T_{:,m}$ 
16:  compute Given's rotation  $G_m$  such that  $X_{:,m} \leftarrow G_m X_{:,m}$  has a zero coefficient at position
     $(m + 1, m)$ 
17:   $\tilde{y} \leftarrow G_m \tilde{y}$ 
18:   $\tilde{v}_m \leftarrow x_{m,m}^{-1}(v_m - \tilde{v}_{m-1}x_{m-1,m} - \tilde{v}_{m-2}x_{m-2,m})$ ; then drop  $\tilde{v}_{m-2}$ 
19:   $\tilde{b}_m \leftarrow x_{m,m}^{-1}(\hat{b} - \tilde{b}_{m-1}x_{m-1,m} - \tilde{b}_{m-2}x_{m-2,m})$ ; then drop  $\tilde{b}_{m-2}$ 
20:   $u \leftarrow u + \tilde{v}_m \tilde{y}_m - \tilde{b}_m \tilde{y}_m$ ;  $\gamma \leftarrow \tilde{y}_{m+1}$  «  $\tilde{y}_m$  and  $\tilde{y}_{m+1}$  are the  $m$ th and  $(m + 1)$ th entries
    of vector  $\tilde{y}$ , so  $\gamma = \|r\|_2$  »
21:  if  $(\gamma/\|f\|_2 \leq \text{tol})$  or  $(\text{mod}(m, s) = 0)$  then
22:     $j \leftarrow j + 1$ 
23:    compute  $U_j$  and  $C_j$  following Algorithm 3
24:    drop  $U_{j-1}$ ,  $C_{j-1}$ , and all  $v$  vectors except  $v_{m+1}$  and  $v_m$ 
25:  end if
26: end while
27: return  $u$  as the solution and initial guess  $u_0$  for the next system,
    and return  $U_j$  as the  $\hat{U}$  for the next system

```

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

11 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

Algorithm 3 Compute U_j and C_j

- 1: compute $C^T C_{j-1}$ following (26) with $C^T C_{j-2}$ and \hat{Q}_{j-1} already available from the $(j-1)$ th cycle
- 2: compute $\bar{V}_j^T C_{j-1}$ following (27) with \hat{Q}_{j-1} already available from the $(j-1)$ th cycle
« Copy the last two rows of \hat{Q}_{j-1} »
- 3: compute $C^T U_{j-1}$ following (28) with $C^T U_{j-2}$ and \hat{P}_{j-1} already available from the $(j-1)$ th cycle
- 4: compute $C_{j-1}^T U_{j-1}$ following (29) with \hat{Q}_{j-1} , $\tilde{W}_{j-1}^T W_{j-1}$ and \hat{P}_{j-1} already available from the $(j-1)$ th cycle
- 5: compute $C_{j-1}^T V_j$ following (30) with \hat{Q}_{j-1} already available from the $(j-1)$ th cycle
« Copy the last column of \hat{Q}_{j-1}^T »
- 6: compute $\bar{V}_j^T U_{j-1}$ by matrix-matrix product
- 7: assemble $\tilde{W}_j^T \tilde{W}_j$ and $\tilde{W}_j^T W_j$ 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 P_j
- 9: compute the QR decomposition of \tilde{W}_j as $\tilde{W}_j = \hat{W}_j F_j$
« Orthogonalize the first two columns of \bar{V}_j against C_{j-1} »
- 10: compute the QR decomposition of $F_j \tilde{H}_j P_j$ as $F_j \tilde{H}_j P_j = Q_j R_j$
- 11: $\hat{P}_j \leftarrow P_j R_j^{-1}$; $\hat{Q}_j \leftarrow F_j^{-1} Q_j$
- 12: $U_j \leftarrow W_j \hat{P}_j$; $C_j \leftarrow \tilde{W}_j \hat{Q}_j$

- 1 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.
- 3 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
- 5 and the conditioning of a corresponding problem with constant density.

5.1. Preconditioning

- 7 The following analysis shows how ill-conditioned the stiffness matrices can be.
 The two-norm condition number of a matrix K can be defined as

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

Since

$$\min_{\|u\|=1} \|Ku\| \leq \|Ke_\ell\| = \|k_\ell\| \leq \max_{\|u\|=1} \|Ku\| \quad \text{for any } \ell = 1, \dots, n$$

where k_ℓ is the ℓ th column of K and e_ℓ is the Cartesian basis vector with the ℓ th coefficient equal to 1, we have

$$\kappa(K) \geq \frac{\|k_{\ell_1}\|}{\|k_{\ell_2}\|} \quad \text{for any } \ell_1, \ell_2 = 1, \dots, n$$

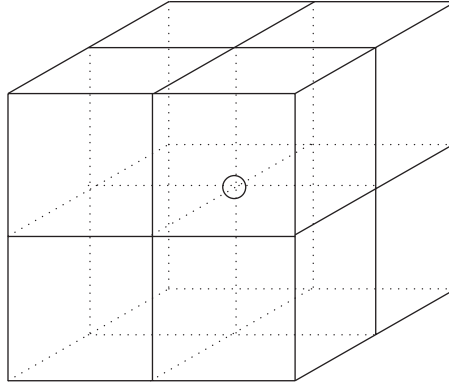


Figure 3. N_ℓ : the set of elements associated with the ℓ th d.o.f. indicated by the circle in the middle.

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

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

3 where \mathbf{K}_0 is the unit element stiffness matrix, L_e is the local-to-global transformation matrix, and
 5 N_ℓ is the set of elements that are associated with the ℓ th 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

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

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

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

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

9 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
 11 design domain.

Note that this analysis provides only a lower bound on the condition number, and that structures
 13 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
 15 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 \mathbf{e}_j , associated with a nodal basis
 17 function in a region with very small density corresponds to a displacement that requires a very
 small amount of energy ($\mathbf{e}_j^T \mathbf{K}(\rho) \mathbf{e}_j$ small). However, that same change in an algebraic degree of
 19 freedom, \mathbf{e}_i , associated with a nodal basis function in a region with large density corresponds to a

1 displacement of the same magnitude that requires a large amount of energy ($e_i^T \mathbf{K}(\rho) e_i$ large). Since
 for symmetric \mathbf{K}

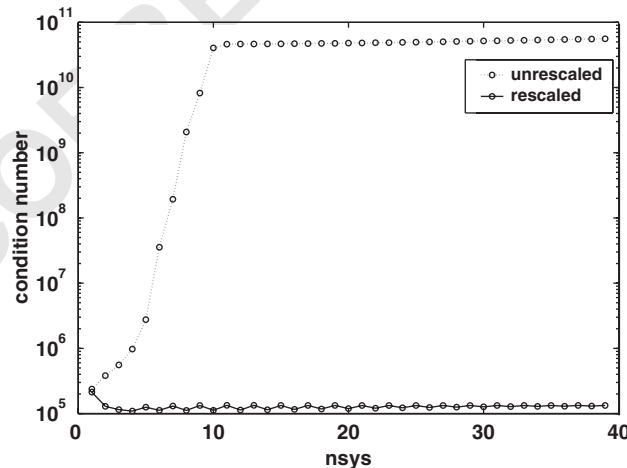
3
$$\kappa(\mathbf{K}(\rho)) \geq \frac{e_i^T \mathbf{K}(\rho) e_i}{e_j^T \mathbf{K}(\rho) e_j}$$

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

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

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

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



23 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.

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

3 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,
 5 we use an incomplete Cholesky decomposition with zero fill-in of the rescaled stiffness matrix as
 a preconditioner [32],

7
$$\tilde{K} = D^{-1/2} K D^{-1/2} \approx LL^T$$

9 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
 11 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
 13 matrix, we explicitly rescale the stiffness matrix before we compute the incomplete Cholesky
 decomposition.

15 We solve the preconditioned system

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

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

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

19 to obtain the solution of the original system $Ku = f$.

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

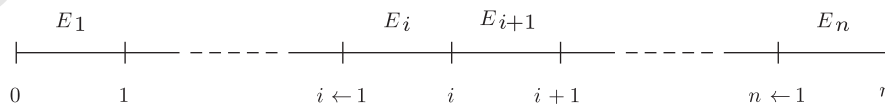
5.2. Rescaling for a 1D elasticity problem

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

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

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

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



29 Figure 5. Piecewise constant modulus of elasticity E_i .

1 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 $(Eu_x)_x = 0 \Leftrightarrow Eu_x = \text{constant}$):

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

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 $\mathbf{\Omega} = \text{diag}(E_1, E_2, \dots, E_{n-1})$, we can write (34) as

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

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

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

where $\mathbf{D}_1^T \mathbf{D}_1 + \mathbf{e}_{n-1} \mathbf{e}_{n-1}^T$ 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 $\mathbf{u} \neq 0$

$$\alpha \leq \frac{\mathbf{u}^T \mathbf{A} \mathbf{u}}{\mathbf{u}^T \mathbf{B} \mathbf{u}} \leq \beta \tag{37}$$

Then

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

where κ denotes the condition number.

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

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

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

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

5 Let $E_{\min} = \min_i (E_i)$, $E_{\max} = \max_i (E_i)$, and let λ_{\min} be the smallest eigenvalue of the matrix $\mathbf{D}_1 \mathbf{D}_1^T$ and λ_{\max} its largest eigenvalue. Furthermore, note that $\mathbf{D}_1 \mathbf{D}_1^T$ and $\mathbf{D}_1^T \mathbf{D}_1$ 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

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

9 which finally gives

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

11 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

$$13 \quad \mathbf{S} = \text{diag}(E_1 + E_2, E_2 + E_3, \dots, E_{n-1} + E_n) \quad (40)$$

We have

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

15

1 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.

3 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
5 the same condition number after scaling as the case of a bar with constant modulus (homogeneous).
7 The third example shows that for the hypothetical case of a 1D bar with two non-adjacent ‘holes’
scaling cannot remove the actual singularity.

9 *Example 1 (Constant modulus)*

For a constant modulus of elasticity, (41) leads to

$$\frac{\mathbf{u}^T(ED_1^T\mathbf{D}_1 + E\mathbf{e}_{n-1}\mathbf{e}_{n-1}^T)\mathbf{u}}{\mathbf{u}^T\mathbf{S}\mathbf{u}} = 1 - \frac{2u_1u_2 + \dots + 2u_{n-2}u_{n-1}}{u_1^2 + (u_1^2 + u_2^2) + \dots + (u_{n-2}^2 + u_{n-1}^2) + u_{n-1}^2} \quad (42)$$

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

13 *Example 2 (Variable modulus)*

Now consider a problem with a ‘hole’ at the end of the bar; $E_i = 1$ for $i = 1, \dots, n - 5$, where
15 $n \gg 5$, 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 $i = 1, \dots, n - 5$ and $|u_i| = O(\varepsilon)$ for the remaining elements. After
17 substituting for the E_i in (41) and dropping the u_i terms that are $O(\varepsilon)$, we need to minimize the
following expression:

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

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

23 *Example 3 (Hypothetical case)*

Finally, consider a hypothetical problem of a 1D bar with two non-adjacent ‘holes’. Let $n = 5$, and
25 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

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

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

3 6. SOME IMPLEMENTATION ISSUES

We have developed our C/C++ code in an object-oriented fashion, since we intend to make the
 5 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.
 7 We store sparse matrices in compressed sparse row format (CSR). The (column) vectors in \mathbf{U} , \mathbf{C} ,
 \mathbf{V}_j , and so on, are stored as 1D arrays linked by a 1D array of pointers. The memory required
 9 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
 11 requires only matrix–vector multiplications, dot products, vector updates, forward and backward
 solves with the incomplete Cholesky factors, and the incomplete Cholesky decomposition itself.
 13 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
 15 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
 17 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
 19 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
 21 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
 23 computational cost is significantly reduced by taking the simplifications in (24)–(30) into account.

7. NUMERICAL RESULTS

25 We demonstrate the performance of the MINRES iterative solver with the recycling and precondi-
 tioning techniques discussed in Sections 4 and 5 on a large 3D design problem. We also provide
 27 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%
 29 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
 31 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
 33 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
 35 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,
 37 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
 39 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.

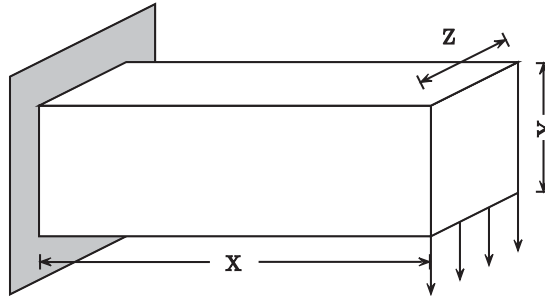


Figure 6. Design problem: finding optimal material distribution in a hexahedron with the left end fixed and a distributed load applied on the right bottom edge ($X:Y:Z = 3:1:1$).

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)

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

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

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

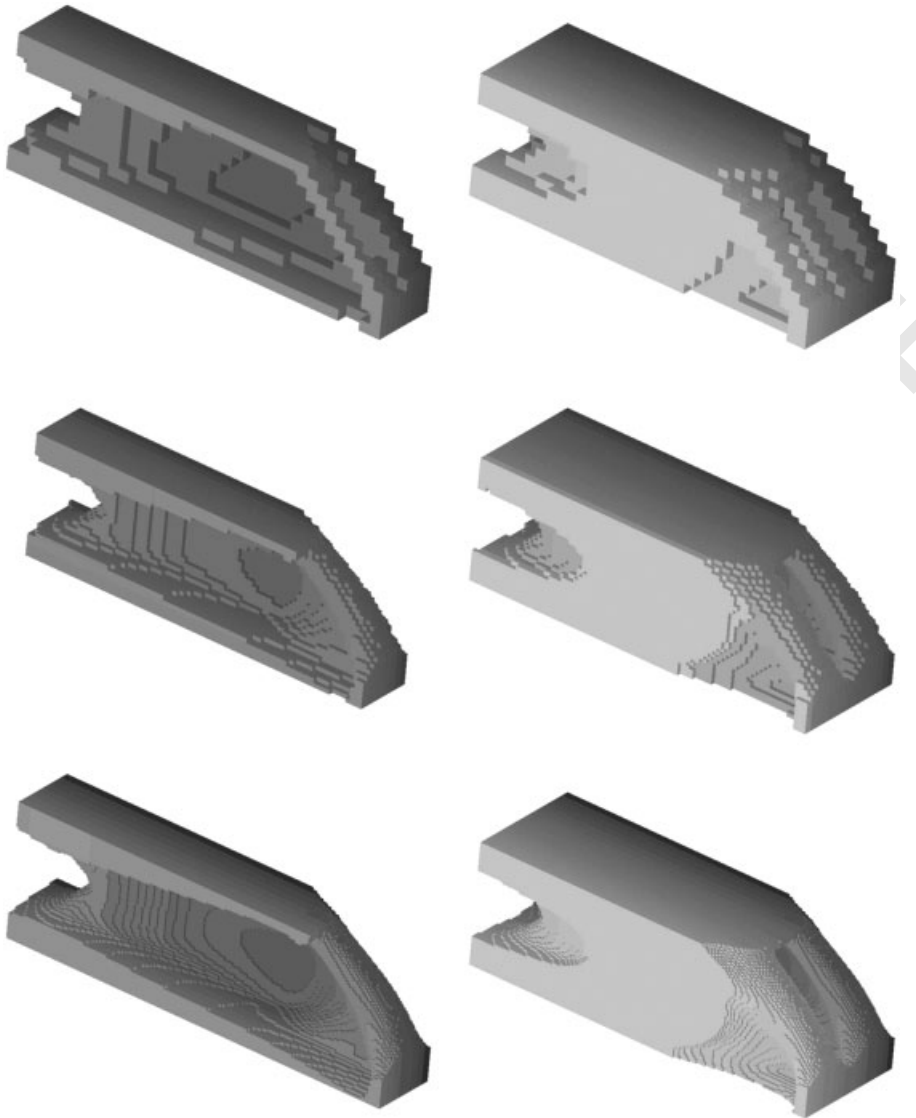


Figure 7. Final topologies of the model problem. Left: half domain; right: full domain: top row: small mesh; middle row: medium size mesh; and bottom row: large mesh.

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

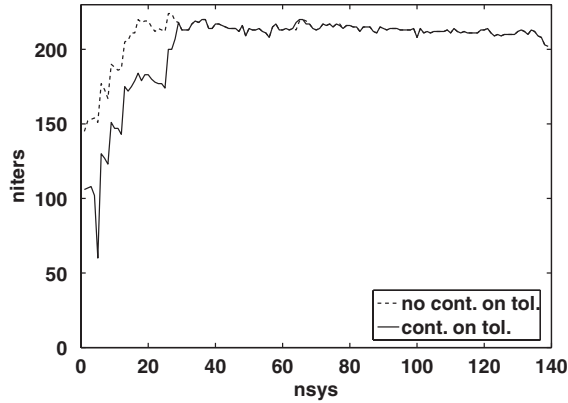


Figure 8. Reduction in the number of iterations using a relaxed tolerance for the linear solver (MINRES without recycling). The jumps in the iteration counts over the first 30 iterations are caused by continuation on the solver tolerance and the penalization parameter.

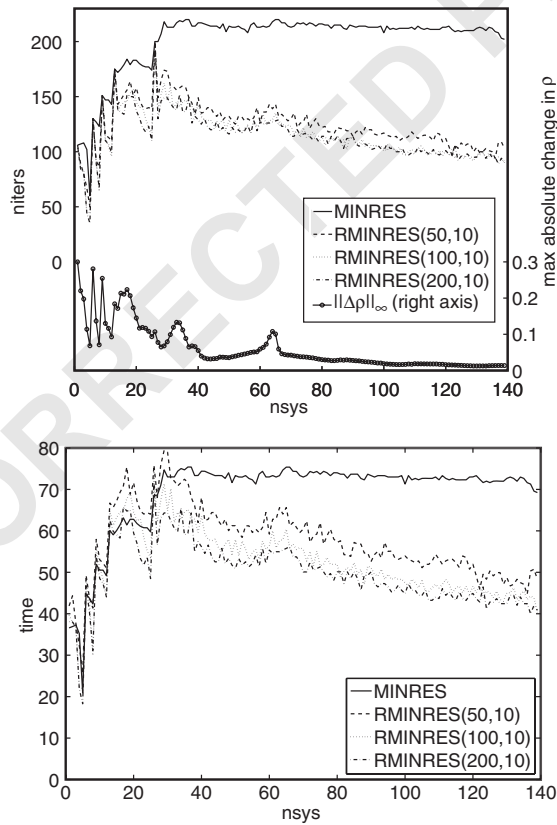


Figure 9. Number of iterations (nits) and time (s) of RMINRES(s, k) with fixed $k = 10$ and varying s .

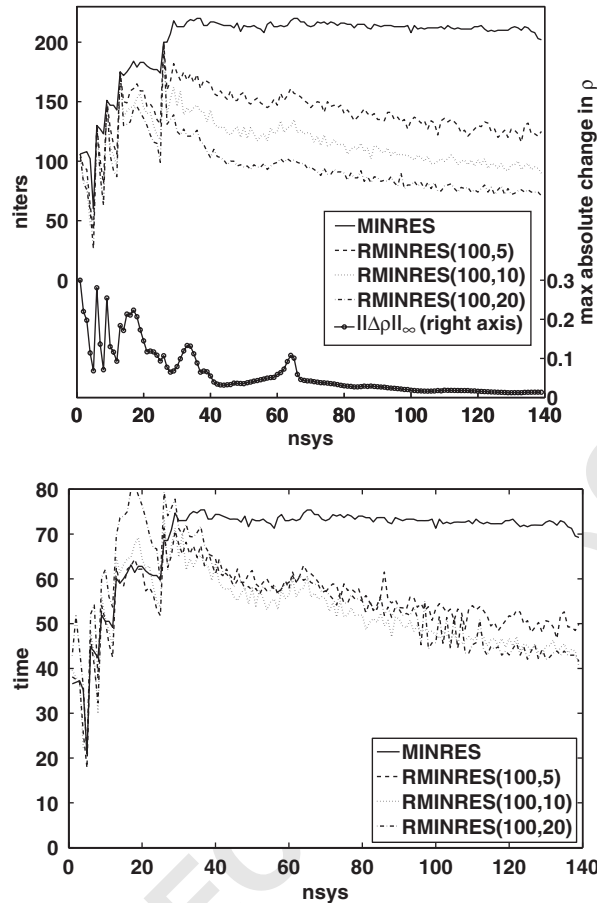


Figure 10. Number of iterations (nits) and time (s) of RMINRES(s, k) with varying k and fixed $s = 100$.

- 1 Notice that increasing s beyond 100 has limited effect since RMINRES rarely takes more than 100
- 3 iterations for this problem. However, for harder problems, e.g. for finer meshes, the solver may not
- 5 converge so fast. In that case, larger values for s can be helpful. Note that increasing s does not
- 7 increase the computational cost of RMINRES. The only limit on s is the memory size.
- 9 In the second set of experiments, we fix $s = 100$ and vary k . The parameter k affects both
- 11 the computational cost per iteration, specifically the number of orthogonalizations and the cost
- 13 of subspace selection, and the total number of iterations for the solver. There is a trade-off be-
- 15 tween these two factors, and in Figure 10 we compare the number of iterations and computational
- 17 time for several values of k . Increasing k leads to a significant improvement in the convergence
- 19 rate; towards the end we obtain a factor 3 reduction in the number of iterations. Timewise, we
- 21 obtain a 40% improvement. We also see that the computation time is not overly sensitive to the
- 23 choice of k .
- 25 Finally, we compare a state-of-the-art sparse direct solver with our iterative method, including
- 27 recycling and the continuation on the solver tolerance, the initial scaling, and the preconditioner.

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™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

1 We use the multifrontal, supernodal Cholesky factorization from the TAUCS package [34]. We
 3 ran the comparison on a PC with an AMD Opteron™252 2.6 GHz 64-bit processor, 8 GB RAM
 5 of memory, and the SuSE Linux system. Table II lists the computation times of both the direct
 7 solver and the iterative solver for a single linear system, for several problem sizes. For the direct
 9 solver, the data include the time for the Cholesky decomposition, and for the forward and backward
 11 substitution to solve one system. For the iterative solver, to make a fair comparison the data include
 13 the minimum, maximum, and average solver time taken over all linear systems, and separately the
 15 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.

17 8. CONCLUSIONS

19 In this paper, we investigate iterative solvers for the equilibrium equations in topology optimiza-
 21 tion. We address the main problem, the extreme ill-conditioning, by two approaches. First, we
 23 rescale the system to reduce the ill-conditioning due to the variation in the density. Second, we
 25 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 precondi-
 tioner 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.

27 We benchmark our methods for a design problem on a sequence of increasingly finer meshes.
 29 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.

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
f	load vector
$\mathbf{K}, \mathbf{K}^{(i)}$	global stiffness matrix
\mathbf{K}_0	element stiffness matrix
K_{ij}	(i, j) coefficient of \mathbf{K}
\mathbf{k}_j	the j th column of \mathbf{K}
p	penalization parameter
$\mathbf{u}, \mathbf{u}^{(i)}$	displacement vector
V	total volume
$\rho, \rho^{(i)}$	density distribution (the design variables of topology optimization problems)
ρ_0	positive lower bound on the density to avoid singularity
Ω	diagonal matrix of material constants
\mathbf{V}_m	first m vectors of the Krylov subspace, $\mathbf{v}_1, \mathbf{v}_2, \dots, \mathbf{v}_m$
$\mathbf{V}_j, \bar{\mathbf{V}}_j$	the Krylov subspace generated in j th cycle, $\mathbf{v}_{(j-1)s+1}, \dots, \mathbf{v}_{js}$, and its extended version, $\mathbf{v}_{(j-1)s}, \dots, \mathbf{v}_{js+1}$
\mathbf{U}, \mathbf{U}_j	basis for the recycle space
\mathbf{C}, \mathbf{C}_j	orthogonal basis of \mathbf{KU}
$\underline{\mathbf{H}}_m, \mathbf{H}_m$	$(m + 1) \times m$ Hessenberg matrix generated in the Arnoldi iteration, and the first m rows of $\underline{\mathbf{H}}_m$
$\mathbf{r}_0, \mathbf{r}_m$	residual
$\underline{\mathbf{T}}_j, \bar{\mathbf{T}}_j$	tridiagonal matrix generated by Lanczos iteration and its extended version with an additional top row
$\kappa(\cdot)$	condition number of a matrix
(θ, \mathbf{w})	an harmonic Ritz pair of \mathbf{K} with Ritz value θ and Ritz vector \mathbf{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]
3 IC(0)	incomplete Cholesky decomposition with zero fill-in [32]

REFERENCES

- 1
- 3 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.
- 5 2. Vemaganti K, Lawrence WE. Parallel methods for optimality criteria-based topology optimization. *Computer Methods in Applied Mechanics and Engineering* 2005; **194**:3637–3667.
- 7 3. Kim TS, Kim JE, Kim YY. Parallelized structural topology optimization for eigenvalue problems. *International Journal of Solids and Structures* 2004; **41**:2623–2641.
- 9 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.
- 11 5. Kilmer ME, de Sturler E. Recycling subspace information for diffuse optical tomography. *SIAM Journal on Scientific Computing* 2006; **27**(6):2140–2166.
- 13 6. Gersborg-Hansen A, Sigmund O, Haber RB. Topology optimization of channel flow problems. *Journal for Structural and Multidisciplinary Optimization* 2005; **30**:181–192.
- 15 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.
- 17 8. Paige CC, Saunders MA. Solution of sparse indefinite systems of linear equations. *SIAM Journal on Numerical Analysis* 1975; **12**:617–629.
- 19 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.
- 21 10. Bendsøe MP, Sigmund O. *Topology Optimization: Theory, Methods and Applications*. Springer: Berlin, Germany, 2003.
- 23 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).
- 25 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.
- 27 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.
- 29 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.
- 31 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.
- 33 16. Bendsøe MP. Optimal shape design as a material distribution problem. *Structural Optimization* 1989; **1**:193–202.
- 35 17. Bendsøe MP, Sigmund O. Material interpolation schemes in topology optimization. *Archives of Applied Mechanics* 1999; **69**(9–10):635–654.
- 37 18. Matsui K, Terada K. Continuous approximation of material distribution for topology optimization. *International Journal for Numerical Methods in Engineering* 2004; **59**:1925–1944.
- 39 19. Paulino GH, Silva ECN. Design of functionally graded structures using topology optimization. *Materials Science Forum* 2005; **492–493**:435–440.
- 41 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.
- 43 21. Rahmatalla S, Swan CC. A Q4/Q4 continuum structural optimization implementation. *Journal for Structural and Multidisciplinary Optimization* 2004; **27**:130–135.
- 45 22. Sigmund O. On the design of compliant mechanisms using topology optimization. *Mechanics of Structures and Machines* 1997; **25**:495–526.
- 47 23. Rozvany GIN. *Structural Design via Optimality Criteria*. Kluwer Academic Publishers: Dordrecht, 1989.
- 49 24. Svanberg K. The method of moving asymptotes: a new method for structural optimization. *International Journal for Numerical Methods in Engineering* 1987; **24**(2).
- 51 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.
- 53
- 55

- 1 26. Arnoldi WE. The principle of minimized iterations in the solution of the matrix eigenvalue problem. *Quarterly*
of *Applied Mathematics* 1951; **9**:17–29.
- 3 27. Saad Y. *Iterative Methods for Sparse Linear Systems* (2nd edn). SIAM: Philadelphia, PA, U.S.A., 2003.
- 5 28. de Sturler E. Nested Krylov methods based on GCR. *Journal of Computational and Applied Mathematics* 1996;
67:15–41.
- 7 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>
- 9 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.
- 11 31. Greenbaum A. *Iterative Methods for Solving Linear Systems*. SIAM: Philadelphia, PA, 1997.
- 13 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.
- 15 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.