Available online at www.sciencedirect.com
ScienceDirect
Comput. Methods Appl. Mech. Engrg. xxx (xxxx) xxx
Computer methods in applied mechanics and engineering
www.elsevier.com/locate/cma

Krylov subspace recycling for evolving structures

M. Bolten[a], E. de Sturler[b,*], C. Hahn[a], M.L. Parks[c]

[a] *Faculty of Mathematics und Natural Sciences, University of Wuppertal, 42119 Wuppertal, Germany*
[b] *Department of Mathematics, Virginia Tech, Blacksburg, VA, 24061, USA*
[c] *Center for Computing Research, Sandia National Laboratories, Albuquerque, NM, 87185, USA*

Received 18 October 2020; received in revised form 1 October 2021; accepted 1 October 2021
Available online xxxx

Abstract

Krylov subspace recycling is a powerful tool when solving a long series of large, sparse linear systems that change only slowly over time. In PDE constrained shape optimization, these series appear naturally, as typically hundreds or thousands of optimization steps are needed with only small changes in the geometry. In this setting, however, applying Krylov subspace recycling can be a difficult task. As the geometry evolves, in general, so does the finite element mesh defined on or representing this geometry, including the numbers of nodes and elements and element connectivity. This is especially the case if re-meshing techniques are used. As a result, the number of algebraic degrees of freedom in the system changes, and in general the linear system matrices resulting from the finite element discretization change size from one optimization step to the next. Changes in the mesh connectivity also lead to structural changes in the matrices. In the case of re-meshing, even if the geometry changes only a little, the corresponding mesh might differ substantially from the previous one. Obviously, this prevents any straightforward mapping of the approximate invariant subspace of the linear system matrix (the focus of recycling in this paper) from one optimization step to the next; similar problems arise for other selected subspaces. In this paper, we present an algorithm to map an approximate invariant subspace of the linear system matrix for the previous optimization step to an approximate invariant subspace of the linear system matrix for the current optimization step, for general meshes. This is achieved by exploiting the map from coefficient vectors to finite element functions on the mesh, combined with interpolation or approximation of functions on the finite element mesh. We demonstrate the effectiveness of our approach numerically with several proof of concept studies for a specific meshing technique.

Keywords: Computations on changing geometries and structures; Krylov subspace recycling; Recycling CG; Shape optimization

1. Introduction

In PDE constrained shape or topology optimization, usually hundreds or thousands of iterations are needed to reach a converged solution. Except for the first few steps, in most applications, in each optimization step the changes in shape are small. In principle, this leads to sequences of matrices with only small changes in each optimization step. These correspond to small changes in the governing PDE, as well. Therefore, we expect that (preconditioned)

* Corresponding author.
E-mail addresses: bolten@math.uni-wuppertal.de (M. Bolten), sturler@vt.edu (E. de Sturler), hahn@math.uni-wuppertal.de (C. Hahn), mlparks@sandia.gov (M.L. Parks).

https://doi.org/10.1016/j.cma.2021.114222
0045-7825/

Please cite this article as: M. Bolten, E. de Sturler, C. Hahn et al., Krylov subspace recycling for evolving structures, Computer Methods in Applied Mechanics and Engineering (2022) 114222, https://doi.org/10.1016/j.cma.2021.114222.

Krylov subspace recycling may significantly speed up the computations, as exploited in [1]. In this paper, we focus on recycling approximate invariant subspaces.

However, in the shape optimization application presented here a significant obstacle to Krylov recycling arises as re-meshing is often necessary. This leads to matrices from one optimization step to the next that have different dimensions, which makes the linear systems algebraically incompatible regarding their sizes. This problem precludes a straightforward application of Krylov subspace recycling, since we cannot transfer the algebraic basis vectors spanning an approximate invariant subspace (or spanning any another useful subspace to recycle) from one linear system to the next. However, we will show that through the sequence of finite element spaces from which the algebraic linear systems are derived, we can map approximate algebraic eigenvectors from one linear system to the next by mapping (through interpolation or other approximations) the corresponding approximate eigenvectors, or basis vectors of an approximate invariant subspace, from one finite element space to the next. In a straightforward way, on meshes stemming from adaptive mesh refinement (AMR) and hence on nested meshes, this idea was explored in [2], however, with relatively modest improvements for convergence [3]. We consider more general mappings and corrections in this paper. We will introduce the mapping for a generic mesh in Section 3. For our numerical examples, we give an example of a mapping for a specific meshing technique in Section 4, the corresponding numerical examples are given in Section 5.

2. Foundations

In shape optimization, the aim is to optimize a given two- or three-dimensional shape without changes in its inherent topology in order to maximize or minimize a given objective, for example efficiency or reliability. Depending on the problem considered, certain material laws have to be fulfilled, which are usually modeled by PDEs, such as the elasticity equations in structural mechanics or the heat equation. This leads, in principle, to a PDE constrained optimization problem, where the variable over which one optimizes is neither a function nor an element of a finite dimensional space, but a domain, i.e., in most cases a subset of $\mathbb{R}^2$ or $\mathbb{R}^3$. In this paper, we restrict ourselves to domains in $\mathbb{R}^2$. In a more formal way, let Ω be a bounded domain with Lipschitz boundary $\partial\Omega$ contained in a bounded set $\widetilde{\Omega} \subset \mathbb{R}^2$ such that each such $\Omega \subseteq \widetilde{\Omega}$ is admissible. This domain Ω is to be optimized subject to some functional $J(\Omega, u)$ that depends on Ω and the solution of a PDE $u \in H^1(\Omega)$, i.e.

$$\begin{aligned} \min_{\Omega} \quad & J(\Omega, u) \\ \text{s.t.} \quad & Lu = f. \end{aligned} \tag{1}$$

We restrict ourselves to changes in the geometry that can be described by a mapping $F_t : \Omega \to \mathbb{R}^2$ that is a perturbation of identity, that is,

$$F_t = \mathrm{id} + t\mathcal{V}, \quad t > 0,$$

where $\mathcal{V} \in (H^{1,\infty}(\Omega))^2$ is a velocity field.

2.1. Krylov subspace recycling

We briefly describe the main ideas behind Krylov subspace recycling and a recycling Conjugate Gradients algorithm (RCG) that combines a Deflated Conjugate Gradient algorithm with a very efficient method to compute, on the side, an approximate invariant subspace for the symmetric positive definite (SPD) system matrix $\mathbf{K}$ that can be used for the next linear system. A Deflated Conjugate Gradient method was first proposed in [4]. A projected CG algorithm was independently proposed in [5]. The implementation here follows [6]. In the next subsection, we present an efficient method to compute and update an approximate invariant subspace that extends the approach in [6] using the ideas for updating the recycle space in recycling MINRES (RMINRES) [1], but adjusted for working in the $\mathbf{K}$-inner product. The approximate invariant subspace update in [6] can only use the first ℓ (a chosen parameter) direction vectors from the Krylov space for a new linear system. Of course, ℓ can be chosen very large, but this requires storing ℓ direction vectors, presenting a possibly overwhelming memory cost and a substantial computational cost. The advantage of RCG is that it uses the whole new Krylov space to update the recycle space, but proceeding in cycles of length m (see Section 2.2) to keep memory requirements and computational costs low. This requires storing at most m direction vectors, independent of the number of iterations to convergence. We

provide some further details at the end of Section 2.2. For a more efficient version of RMINRES than presented in [1] see [7]; the first version of RMINRES was proposed in [8]. The CG and MINRES algorithms were proposed in [9] and [10], respectively. An interesting alternative to recycling is proposed in [11] for applications with a fixed, Hermitian, matrix and many right hand sides (so rather different from our objective here). It proposes to compute eigenvectors accurately in the course of solving multiple linear systems and then accurately deflate the corresponding invariant subspace from the remaining right hand sides to obtain good convergence. This avoids the cost of the orthogonalization in each iteration of RCG/deflated CG at the cost of many more iterations for accurate eigenvectors. In [11] these iterations can be spread over multiple right hand side solves; however, for shape optimization the method is not efficient as the matrix generally changes each optimization step. For similar ideas in the nonsymmetric case; see [12].

The purpose of recycling is to reuse a judiciously selected subspace from the search space generated by earlier linear solves to speed-up the convergence of subsequent linear solves. Here, we focus on approximate invariant subspaces associated with small (in absolute value) eigenvalues. Depending on whether the right hand sides change substantially or not, recycling may also give good initial guesses [8]. For several recent Krylov subspace recycling methods for engineering applications see [13–19], and for HPC implementations see [20–22]. A survey of Krylov subspace recycling methods is given in [23].

In this section, we do not consider the complication that our matrices from one optimization step to the next may be incompatible. We will address this issue in Section 3.

Algorithm 1 Preconditioned Deflated Conjugate Gradients (see [6])

Input $\mathbf{K}$, $\mathbf{W} = [\mathbf{w}_1, \ldots, \mathbf{w}_k]$ (**range**($\mathbf{W}$) is the recycle space), SPD preconditioner $\mathbf{M}^{-1}$, and initial guess $\mathbf{u}_0$ satisfying $\mathbf{r}_0 = \mathbf{f} - \mathbf{K}\mathbf{u}_0 \perp \mathbf{W}$.
Compute $\mathbf{z}_0 = \mathbf{M}^{-1}\mathbf{r}_0$; $\mathbf{p}_0 = \mathbf{z}_0 - \mathbf{W}(\mathbf{W}^T\mathbf{K}\mathbf{W})^{-1}\mathbf{W}^T\mathbf{K}\mathbf{z}_0$.
for $i = 1, 2, \ldots, \ell$ **do**
  $\alpha_{i-1} = \mathbf{r}_{i-1}^T\mathbf{z}_{i-1}/\mathbf{p}_{i-1}^T\mathbf{K}\mathbf{p}_{i-1}$
  $\mathbf{u}_i = \mathbf{u}_{i-1} + \alpha_{i-1}\mathbf{p}_{i-1}$
  $\mathbf{r}_i = \mathbf{r}_{i-1} - \alpha_{i-1}\mathbf{K}\mathbf{p}_{i-1}$
  $\mathbf{z}_i = \mathbf{M}^{-1}\mathbf{r}_i$
  $\beta_{i-1} = \mathbf{r}_i^T\mathbf{z}_i/\mathbf{r}_{i-1}^T\mathbf{z}_{i-1}$
  Solve $(\mathbf{W}^T\mathbf{K}\mathbf{W})\mathbf{t}_i = (\mathbf{K}\mathbf{W})^T\mathbf{z}_i$
  $\mathbf{p}_i = \beta_{i-1}\mathbf{p}_{i-1} + \mathbf{z}_i - \mathbf{W}\mathbf{t}_i$
end for

Consider at some optimization step the linear system $\mathbf{K}\mathbf{u} = \mathbf{f}$ with SPD $\mathbf{K} \in \mathbb{R}^{n\times n}$, a SPD preconditioner $\mathbf{M}^{-1}$, and a recycle space **range**($\mathbf{W}$) with $\mathbf{W} \in \mathbb{R}^{n\times k}$ computed in the previous optimization step. We give the preconditioned deflated CG [6] in Algorithm 1, focusing on the mathematical relations rather than efficient implementation. This algorithm solves the preconditioned system $\mathbf{M}^{-1}\mathbf{K}\mathbf{u} = \mathbf{M}^{-1}\mathbf{f}$, working in the $\mathbf{M}$-inner product, $\langle \mathbf{x}, \mathbf{y}\rangle_{\mathbf{M}} = \mathbf{y}^T\mathbf{M}\mathbf{x}$, and minimizing the error in the operator norm $\mathbf{M}^{-1}\mathbf{K}$, which is self-adjoint and positive definite with respect to this inner product. The algorithm still minimizes the error in the $\mathbf{K}$-norm (energy norm) [24] but over the augmented preconditioned Krylov space.

As for standard CG, the search direction vectors, $\mathbf{p}_i$, satisfy $\mathbf{p}_j^T\mathbf{K}\mathbf{p}_i = 0$ for $i \neq j$, and in addition $\mathbf{W}^T\mathbf{K}\mathbf{p}_i = \mathbf{0}$ (the zero vector) for all i; see [6].

2.2. Computing a recycle space

In our recycling (preconditioned) conjugate gradient algorithm, $\mathbf{W} \in \mathbb{R}^{n\times k}$ represents the initial k-dimensional recycle space, **range**($\mathbf{W}$). As RCG is optimal over **range**($\mathbf{W}$) $\oplus$ **range**($[\mathbf{p}_0\ \mathbf{p}_1\ \ldots\ \mathbf{p}_{\ell-1}]$) (after ℓ iterations), the recycle space itself is not changed during the solve. However, for the next linear system, we want to recycle an improved approximate invariant subspace, represented by $\mathbf{Y}$, which we compute while solving the current linear system. RCG, due to the short recurrence, does not need restarting or truncating; however, to update the recycle space, we need to keep a sequence of search vectors $\mathbf{p}_i$. To limit the memory requirements, we periodically update

the new recycle space $\mathbf{Y}$ using (max) m previous search vectors $\mathbf{p}_i$ before discarding them. We use the term *cycle* to denote the solution process between two updates of the new recycle space. Let $\mathbf{Y}_j \in \mathbb{R}^{n\times k}$ represent the recycle space selected after cycle j. Utilizing notation from [1], let $\mathbf{P}_j = [\mathbf{p}_{(j-1)m} \ \ldots \ \mathbf{p}_{jm-1}] \in \mathbb{R}^{n\times m}$ and $\underline{\mathbf{P}}_j = [\mathbf{p}_{(j-1)m-1} \ \ldots \ \mathbf{p}_{jm}] = [\mathbf{P}_{j-1}\mathbf{e}_m \mid \mathbf{P}_j \mid \mathbf{P}_{j+1}\mathbf{e}_1] \in \mathbb{R}^{n\times(m+2)}$, with standard basis vectors $\mathbf{e}_1, \mathbf{e}_m \in \mathbb{R}^m$; $\underline{\mathbf{P}}_1 = [\mathbf{0} \mid \mathbf{P}_1 \mid \mathbf{P}_2\mathbf{e}_1]$.

Here, we consider only approximate invariant subspaces using harmonic Ritz vectors [25] for the $\mathbf{M}$-inner product and the preconditioned matrix $\mathbf{M}^{-1}\mathbf{K}$ ($\mathbf{M} = \mathbf{I}$ for the unpreconditioned case). For any subspace $\mathcal{S} \subseteq \mathbb{R}^n$, $\boldsymbol{\xi} \in \mathcal{S}$ is a harmonic Ritz vector of $\mathbf{M}^{-1}\mathbf{K}$ with harmonic Ritz value θ if

$$\mathbf{M}^{-1}\mathbf{K}\boldsymbol{\xi} - \theta\boldsymbol{\xi} \ \perp_{\mathbf{M}} \ \mathbf{M}^{-1}\mathbf{K}\,\mathcal{S}. \tag{2}$$

Taking $\boldsymbol{\xi} = \mathbf{S}\mathbf{x}$ with $\mathbf{range}(\mathbf{S}) = \mathcal{S}$ leads to the generalized eigenvalue problem

$$\mathbf{S}^T\mathbf{K}\mathbf{M}^{-1}\mathbf{K}\mathbf{S}\mathbf{x} = \theta\mathbf{S}^T\mathbf{K}\mathbf{S}\mathbf{x}. \tag{3}$$

We discuss the computation of $\mathbf{Y}_j$ for the general case $j > 1$ and outline changes for other cases. We set $\mathbf{S} = [\mathbf{Y}_{j-1}\ \mathbf{P}_j]$, which leads to the $(m+k)\times(m+k)$ generalized eigenvalue problem

$$\mathbf{G}_j\mathbf{x} = \theta\mathbf{F}_j\mathbf{x}, \qquad \text{where} \tag{4}$$

$$\mathbf{G}_j = \begin{bmatrix} (\mathbf{K}\mathbf{Y}_{j-1})^T\mathbf{M}^{-1}(\mathbf{K}\mathbf{Y}_{j-1}) & (\mathbf{K}\mathbf{Y}_{j-1})^T\mathbf{M}^{-1}(\mathbf{K}\mathbf{P}_j) \\ (\mathbf{K}\mathbf{P}_j)^T\mathbf{M}^{-1}(\mathbf{K}\mathbf{Y}_{j-1}) & (\mathbf{K}\mathbf{P}_j)^T\mathbf{M}^{-1}(\mathbf{K}\mathbf{P}_j) \end{bmatrix},$$

$$\mathbf{F}_j = \begin{bmatrix} \mathbf{Y}_{j-1}^T\mathbf{K}\mathbf{Y}_{j-1} & \mathbf{Y}_{j-1}^T\mathbf{K}\mathbf{P}_j \\ \mathbf{P}_j^T\mathbf{K}\mathbf{Y}_{j-1} & \mathbf{P}_j^T\mathbf{K}\mathbf{P}_j \end{bmatrix}.$$

For $j = 1$, we take $\mathbf{Y}_0 = \mathbf{W}$ and $\mathbf{S} = [\mathbf{Y}_0\ \mathbf{P}_1]$ (this corresponds to the update in [6] for a new linear system after the first). If no recycle space is available, we take $\mathbf{S} = \mathbf{P}_1$ (with obvious changes for $\mathbf{G}_1$ and $\mathbf{F}_1$). For SPD $\mathbf{K}$ and $\mathbf{M}$, (4) has real solutions with positive eigenvalues. After solving (4), we choose the k harmonic Ritz vectors with the smallest magnitude harmonic Ritz values, and set $\mathbf{X}_j = [\mathbf{x}_1 \ \ldots \ \mathbf{x}_k]$, and

$$\mathbf{Y}_j \ = \ \mathbf{S}\mathbf{X}_j \ = \ [\mathbf{Y}_{j-1}\ \mathbf{P}_j]\mathbf{X}_j. \tag{5}$$

The main cost in solving for $\mathbf{X}_j$ is the computation of $\mathbf{G}_j$ and $\mathbf{F}_j$. Fortunately, we can compute $\mathbf{G}_j$ and $\mathbf{F}_j$ efficiently using recurrences, avoiding unnecessary storage and computation. Again, we only discuss the general case ($j > 1$) in detail.

We start with the recurrences for $\mathbf{G}_j$. From (5), we have for $\mathbf{Y}_{j-1}$

$$\mathbf{Y}_{j-1} = [\mathbf{Y}_{j-2}\ \ \mathbf{P}_{j-1}]\mathbf{X}_{j-1} \quad \text{and} \quad \mathbf{K}\mathbf{Y}_{j-1} = [\mathbf{K}\mathbf{Y}_{j-2}\ \ \mathbf{K}\mathbf{P}_{j-1}]\mathbf{X}_{j-1}. \tag{6}$$

For the (1,1) block of $\mathbf{G}_j$, using (6), we get $(\mathbf{K}\mathbf{Y}_{j-1})^T\mathbf{M}^{-1}(\mathbf{K}\mathbf{Y}_{j-1}) = \mathbf{X}_{j-1}^T\mathbf{G}_{j-1}\mathbf{X}_{j-1}$. Next, from Algorithm 1 lines 6–7, we have

$$\mathbf{M}^{-1}\mathbf{K}\mathbf{p}_{i-1} = (\mathbf{z}_{i-1} - \mathbf{z}_i)\alpha_{i-1}^{-1} = [\mathbf{z}_{i-1}\ \mathbf{z}_i]\begin{bmatrix} \alpha_{i-1}^{-1} \\ -\alpha_{i-1}^{-1} \end{bmatrix}, \tag{7}$$

and from line 10 (with $\mathbf{t}_i$ from line 9),

$$\mathbf{z}_i = \mathbf{W}\mathbf{t}_i + \mathbf{p}_i - \beta_{i-1}\mathbf{p}_{i-1} = \mathbf{W}\mathbf{t}_i + [\mathbf{p}_{i-1}\ \mathbf{p}_i]\begin{bmatrix} -\beta_{i-1} \\ 1 \end{bmatrix}, \tag{8}$$

which together give (at the end of cycle j)

$$\mathbf{M}^{-1}\mathbf{K}\mathbf{P}_j = (\mathbf{W}\mathbf{T}_j + \underline{\mathbf{P}}_j\boldsymbol{\Phi}_j)\boldsymbol{\Psi}_j, \tag{9}$$

where $\mathbf{T}_j = [\mathbf{t}_{(j-1)m} \ \ \mathbf{t}_{(j-1)m+1} \ \ \ldots \ \ \mathbf{t}_{jm}]$, $\boldsymbol{\Psi}_j^{(m+1)\times m}$ is a lower bidiagonal matrix with the m coefficients $\alpha_{(j-1)m}^{-1}$, $\alpha_{(j-1)m+1}^{-1}$, $\ldots$, α_{jm-1}^{-1} from (7) on the diagonal and $-\alpha_{(j-1)m}^{-1}$, $-\alpha_{(j-1)m+1}^{-1}$, $\ldots$, $-\alpha_{jm-1}^{-1}$ on the subdiagonal, and $\boldsymbol{\Phi}_j^{(m+2)\times(m+1)}$ is a lower bidiagonal matrix with the coefficients $-\beta_{(j-1)m-1}$, $-\beta_{(j-1)m}$, $\ldots$, $-\beta_{jm-1}$ from (8) on the main diagonal and 1's on the subdiagonal.

For the (1,2) block of $\mathbf{G}_j$ (and the (2,1) block by transposition), we have

$$\begin{aligned}(\mathbf{K}\mathbf{Y}_{j-1})^T\mathbf{M}^{-1}\mathbf{K}\mathbf{P}_j &= \mathbf{Y}_{j-1}^T\mathbf{K}(\mathbf{W}\mathbf{T}_j + \underline{\overline{\mathbf{P}}}_j\mathbf{\Phi}_j)\mathbf{\Psi}_j \\ &= (\mathbf{Y}_{j-1}^T\mathbf{K}\mathbf{W})\mathbf{T}_j\mathbf{\Psi}_j + (\mathbf{Y}_{j-1}^T\mathbf{K}\underline{\overline{\mathbf{P}}}_j)\mathbf{\Phi}_j\mathbf{\Psi}_j. \end{aligned} \tag{10}$$

The matrices in brackets can be computed by recurrences at the end of the previous iteration. Using (6) and $\mathbf{P}_{j-1}^T\mathbf{K}\mathbf{W} = \mathbf{O}$ (the zero matrix), as $\mathbf{p}_i \perp_{\mathbf{K}} \mathbf{W}$ for all i, we get

$$\mathbf{Y}_{j-1}^T\mathbf{K}\mathbf{W} = \mathbf{X}_{j-1}^T\begin{bmatrix}\mathbf{Y}_{j-2}^T \\ \mathbf{P}_{j-1}^T\end{bmatrix}\mathbf{K}\mathbf{W} = \mathbf{X}_{j-1}^T\begin{bmatrix}\mathbf{Y}_{j-2}^T\mathbf{K}\mathbf{W} \\ \mathbf{O}\end{bmatrix}, \tag{11}$$

$$\text{with} \quad \mathbf{Y}_1^T\mathbf{K}\mathbf{W} = \mathbf{X}_1^T\begin{bmatrix}\mathbf{W}^T\mathbf{K}\mathbf{W} \\ \mathbf{O}\end{bmatrix}, \tag{12}$$

where $\mathbf{Y}_{j-2}^T\mathbf{K}\mathbf{W}$ is available from the previous iteration. For $j = 1$, $\mathbf{Y}_0^T\mathbf{K}\mathbf{W} = \mathbf{W}^T\mathbf{K}\mathbf{W}$, and $\mathbf{W}^T\mathbf{K}\mathbf{W}$ is available from the recycling CG iteration. For $\mathbf{Y}_{j-1}^T\mathbf{K}\underline{\overline{\mathbf{P}}}_j$, we have, again using (6),

$$\mathbf{Y}_{j-1}^T\mathbf{K}\underline{\overline{\mathbf{P}}}_j = \mathbf{X}_{j-1}^T\begin{bmatrix}\mathbf{Y}_{j-2}^T\mathbf{K}\underline{\overline{\mathbf{P}}}_j \\ \mathbf{P}_{j-1}^T\mathbf{K}\underline{\overline{\mathbf{P}}}_j\end{bmatrix}.$$

Using $\mathbf{p}_i \perp_{\mathbf{K}} \mathbf{W}$ $(= \mathbf{Y}_0)$, $\mathbf{P}_i^T\mathbf{K}\mathbf{P}_j = \mathbf{O}$ for $i \neq j$, and the recurrence implied by (6), $\mathbf{range}(\mathbf{Y}_{j-2}) \subseteq \mathbf{range}([\mathbf{Y}_{j-3} \ \mathbf{P}_{j-2}])$, we have $\mathbf{Y}_{j-2}^T\mathbf{K}\underline{\overline{\mathbf{P}}}_j = \mathbf{O}$. For the second block, we have $\mathbf{P}_{j-1}^T\mathbf{K}\underline{\overline{\mathbf{P}}}_j = \mathbf{P}_{j-1}^T\mathbf{K}[\,\mathbf{P}_{j-1}\mathbf{e}_m \mid \mathbf{P}_j \mid \mathbf{P}_{j+1}\mathbf{e}_1] = d_{(j-1)m-1}\mathbf{e}_m\mathbf{e}_1^T$, with standard basis vectors $\mathbf{e}_m \in \mathbb{R}^m$ and $\mathbf{e}_1 \in \mathbb{R}^{m+2}$, and $d_{(j-1)m-1} = \mathbf{p}_{(j-1)m-1}^T\mathbf{K}\mathbf{p}_{(j-1)m-1}$. Hence,

$$\mathbf{Y}_{j-1}^T\mathbf{K}\underline{\overline{\mathbf{P}}}_j = \mathbf{X}_{j-1}^T\begin{bmatrix}\mathbf{O} \\ d_{(j-1)m-1}\mathbf{e}_m\mathbf{e}_1^T\end{bmatrix}, \tag{13}$$

$$\text{with} \quad \mathbf{Y}_0^T\mathbf{K}\underline{\overline{\mathbf{P}}}_1 = \mathbf{W}^T\mathbf{K}\underline{\overline{\mathbf{P}}}_1 = \mathbf{O}. \tag{14}$$

In addition $(\mathbf{e}_1^{(m+2)})^T\mathbf{\Phi}_j\mathbf{\Psi}_j = -\frac{\beta_{(j-1)m-1}}{\alpha_{(j-1)m}}(\mathbf{e}_1^{(m)})^T$ with $\mathbf{e}_1^{(m+2)} \in \mathbb{R}^{m+2}$ and $\mathbf{e}_1^{(m)} \in \mathbb{R}^m$.

Finally, for the (2,2) block of $\mathbf{G}_j$ we get, using (9) and $\mathbf{P}_j^T\mathbf{K}\mathbf{W} = \mathbf{O}$, the $m \times m$ tridiagonal matrix

$$\begin{aligned}(\mathbf{K}\mathbf{P}_j)^T\mathbf{M}^{-1}(\mathbf{K}\mathbf{P}_j) &= (\mathbf{K}\mathbf{P}_j)^T(\mathbf{W}\mathbf{T}_j + \underline{\overline{\mathbf{P}}}_j\mathbf{\Phi}_j)\mathbf{\Psi}_j = \mathbf{P}_j^T\mathbf{K}\underline{\overline{\mathbf{P}}}_j\mathbf{\Phi}_j\mathbf{\Psi}_j \\ &= [\mathbf{0} \mid \mathbf{D}_j \mid \mathbf{0}]\mathbf{\Phi}_j\mathbf{\Psi}_j,\end{aligned}$$

where $\mathbf{D}_j = \mathbf{P}_j^T\mathbf{K}\mathbf{P}_j$ is a diagonal matrix with coefficients $d_i = \mathbf{p}_i^T\mathbf{K}\mathbf{p}_i$, for $i = (j-1)m, (j-1)m+1, \ldots, jm-1$, on the diagonal. The coefficients d_i, α_i, β_i have all been computed in the past RCG cycle.

Next, we consider the blocks of $\mathbf{F}_j$. Just as for the (1,1) block of $\mathbf{G}_j$, using (6), we get for the (1,1) block of $\mathbf{F}_j$ that $\mathbf{Y}_{j-1}^T\mathbf{K}\mathbf{Y}_{j-1} = \mathbf{X}_{j-1}^T\mathbf{F}_{j-1}\mathbf{X}_{j-1}$. The (1,2) block and (2,1) block of $\mathbf{F}_j$ satisfy $\mathbf{Y}_{j-1}^T\mathbf{K}\mathbf{P}_j = \mathbf{O}$. Finally, the (2,2) block of $\mathbf{F}_j$ equals $\mathbf{D}_j$.

Algorithm 2 outlines the (preconditioned) Recycling CG algorithm that includes the recycle space into the search space. Algorithm 3 gives the computation of $\mathbf{G}_j$ and $\mathbf{F}_j$. The invariant subspace update in [6] corresponds to Algorithm 3 for $j == 1$ (with some minor changes). Extending this update to proceed in cycles (with $j > 1$) requires the additional recurrences derived above to update $\mathbf{G}_j$ and $\mathbf{F}_j$ without any additional $O(n)$ computations and storing at most m direction vectors.

While algorithm 2 allows any subspace to be recycled, we focus here on approximate invariant subspaces, in particular, those corresponding to small eigenvalues. Removing the small eigenvalues leads to substantially improved rates of convergence as the condition number is significantly improved [6]. See [26] for a general discussion of the link between the condition number and CG convergence for SPD systems.

3. Recycling CG for evolving geometries

In shape optimization, the changes in geometry in each optimization step and thus in the underlying mesh prevent a straightforward application of the described Krylov subspace recycling. Depending on the meshing technique, a mapping of the matrix representing the subspace in one optimization step to the next might be necessary.

Algorithm 2 Recycling CG

Input: matrix $\mathbf{K}$, right hand side $\mathbf{f}$, preconditioner $\mathbf{M}$, (max) cycle length m, recycle space dimension k, recycle space basis $\mathbf{W} = [\mathbf{w}_1, \ldots, \mathbf{w}_k]$ (possibly empty), $\mathbf{Y}_0 = \mathbf{W}$, initial guess $\mathbf{u}_0$, and convergence tolerance τ.
$\mathbf{r}_0 = \mathbf{f} - \mathbf{K}\mathbf{u}_0$.
if $\mathbf{W}$ is defined (from solving a previous linear system) **then**
  Compute $\widehat{\mathbf{W}} = \mathbf{K}\mathbf{W}$; $\mathbf{L}\mathbf{L}^T = \mathbf{W}^T\widehat{\mathbf{W}}$ (Cholesky decomposition)
  Solve $(\mathbf{L}\mathbf{L}^T)\mathbf{y} = \mathbf{W}^T\mathbf{r}_0$; $\mathbf{u}_0 = \mathbf{u}_0 + \mathbf{W}\mathbf{y}$; $\mathbf{r}_0 = \mathbf{r}_0 - \widehat{\mathbf{W}}\mathbf{y}$
  $\mathbf{z}_0 = \mathbf{M}^{-1}\mathbf{r}_0$; Solve $(\mathbf{L}\mathbf{L}^T)\mathbf{t}_0 = \widehat{\mathbf{W}}^T\mathbf{z}_0$; $\mathbf{p}_0 = \mathbf{z}_0 - \mathbf{W}\mathbf{t}_0$
else
  $\mathbf{z}_0 = \mathbf{M}^{-1}\mathbf{r}_0$; $\mathbf{p}_0 = \mathbf{z}_0$
end if
$i = 0$ {iteration index}; $j = 1$ {cycle index}
while $\|r_i\|_2 > \tau$ **do**
  $i = i + 1$
  {Do one step of Preconditioned Deflated CG (Algorithm 1).}
  $\alpha_{i-1} = (\mathbf{r}_{i-1}^T\mathbf{z}_{i-1})/(\mathbf{p}_{i-1}^T\mathbf{K}\mathbf{p}_{i-1})$
  $\mathbf{u}_i = \mathbf{u}_{i-1} + \alpha_{i-1}\mathbf{p}_{i-1}$; $\mathbf{r}_i = \mathbf{r}_{i-1} - \alpha_{i-1}\mathbf{K}\mathbf{p}_{i-1}$
  $\mathbf{z}_i = \mathbf{M}^{-1}\mathbf{r}_i$
  $\beta_{i-1} = \mathbf{r}_i^T\mathbf{z}_i/\mathbf{r}_{i-1}^T\mathbf{z}_{i-1}$
  Solve $\mathbf{L}\mathbf{L}^T\mathbf{t}_i = \widehat{\mathbf{W}}^T\mathbf{z}_i$; $\mathbf{p}_i = \beta_{i-1}\mathbf{p}_{i-1} + \mathbf{z}_i - \mathbf{W}\mathbf{t}_i$
  if mod $(i, m) == 0$ **then**
    {Update recycle space $\mathbf{Y}_j$ for use in next linear system.}
    Compute $\mathbf{G}_j, \mathbf{F}_j$ from (4) ff. (Algorithm 3) and solve $\mathbf{G}_j\mathbf{x} = \theta\mathbf{F}_j\mathbf{x}$.
    Let $\mathbf{X}_j$ contain k eigenvectors corresponding to k smallest magnitude eigenvalues (or an alternative selection, if desired).
    $\mathbf{Y}_j = [\mathbf{Y}_{j-1}\ \ \mathbf{P}_j]\mathbf{X}_j$; {$\mathbf{Y}_0 = \mathbf{W}$ if $\mathbf{W}$ exists, otherwise empty}
    $j = j + 1$
  end if
end while
$\mathbf{W} = \mathbf{Y}_j$ {Assign recycle space for the next system.}

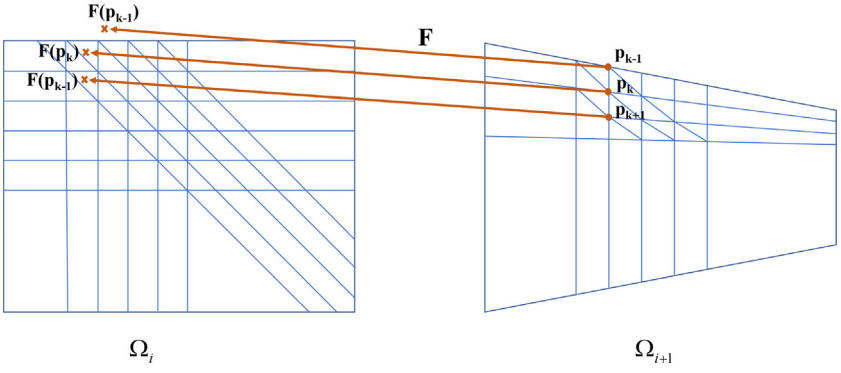


Fig. 1. Mapping new mesh nodes to the domain of the previous optimization step.

3.1. Mapping between successive meshes

Let Ω_i and Ω_{i+1} be two domains representing two immediately consecutive shapes stemming from an iterative shape optimization procedure, see Fig. 1. These domains are discretized by finite element meshes T_i and T_{i+1}, respectively. In general, the meshes feature different connectivities and different numbers of nodes, N_i and N_{i+1},

Algorithm 3 Computing $\mathbf{G}_j$ and $\mathbf{F}_j$

if j == 1 **then**
  $\mathbf{F}_1(1,1) = \mathbf{W}^T\mathbf{K}\mathbf{W}$ (available from RCG)
  $\mathbf{F}_1(1,2) = \mathbf{O}^{k\times m}$; $\mathbf{F}_1(2,1) = \mathbf{O}^{m\times k}$
  $\mathbf{F}_1(2,2) = \mathbf{D}_1$ (available from RCG)
  $\mathbf{G}_1(1,1) = (\mathbf{K}\mathbf{W})^T\mathbf{M}^{-1}(\mathbf{K}\mathbf{W})$ ($\mathbf{K}\mathbf{W}$ available from RCG)
  $\mathbf{G}_1(1,2) = \mathbf{W}^T\mathbf{K}\mathbf{W}(\mathbf{T}_1\Psi_1)$ ($\mathbf{W}^T\mathbf{K}\mathbf{W}$ available from RCG)
  $\mathbf{G}_1(2,1) = \mathbf{G}_1(1,2)^T$
  $\mathbf{G}_1(2,2) = [\mathbf{0}|\mathbf{D}_1|\mathbf{0}](\Phi_1\Psi_1)$
  Solve for $\mathbf{X}_1$ from $\mathbf{G}_1\mathbf{x} = \theta\mathbf{F}_1\mathbf{x}$
  Compute $\mathbf{Y}_1^T\mathbf{K}\mathbf{W}$ from (12); Compute $\mathbf{Y}_1^T\mathbf{K}\overline{\mathbf{P}}_2$ from (13)
  $\mathbf{G}_2(1,1) = \mathbf{X}_1^T\mathbf{G}_1\mathbf{X}_1$; $\mathbf{F}_2(1,1) = \mathbf{X}_1^T\mathbf{F}_1\mathbf{X}_1$
else
  $\mathbf{F}_j(1,2) = \mathbf{O}^{k\times m}$; $\mathbf{F}_j(2,1) = \mathbf{O}^{m\times k}$
  $\mathbf{F}_j(2,2) = \mathbf{D}_j$ (available from RCG)
  Compute $\mathbf{G}_j(1,2)$ from (10)–(13)
  $\mathbf{G}_j(2,1) = \mathbf{G}_j(1,2)^T$
  $\mathbf{G}_j(2,2) = [\mathbf{0}|\mathbf{D}_j|\mathbf{0}](\Phi_j\Psi_j)$
  Solve for $\mathbf{X}_j$ from $\mathbf{G}_j\mathbf{x} = \theta\mathbf{F}_j\mathbf{x}$
  Compute $\mathbf{Y}_j^T\mathbf{K}\mathbf{W}$ from (11) and (12); Compute $\mathbf{Y}_j^T\mathbf{K}\overline{\mathbf{P}}_{j+1}$ from (13)
  $\mathbf{G}_{j+1}(1,1) = \mathbf{X}_j^T\mathbf{G}_j\mathbf{X}_j$; $\mathbf{F}_{j+1}(1,1) = \mathbf{X}_j^T\mathbf{F}_j\mathbf{X}_j$
end if

respectively. Additionally, from optimization step i, an approximate invariant subspace is given by $\mathbf{range}(\mathbf{W}_k^{(i)})$, with $\mathbf{W}_k^{(i)} \in \mathbb{R}^{N_i\times k}$, which is supposed to be recycled in optimization step $i+1$. However, the system in step $i+1$ is of dimension N_{i+1}. Therefore, we are in need of a function that maps the $N_i \times k$ matrix representing the approximate invariant subspace of the system in optimization step i to a $N_{i+1} \times k$ matrix, which is meant to represent a good approximate invariant subspace of the linear system in optimization step $i+1$.

The mapping we propose exploits the fact that the linear systems we consider are closely linked to the continuous finite element spaces $\mathcal{V}_h(\Omega_i, T_i)$ and $\mathcal{V}_h(\Omega_{i+1}, T_{i+1})$, determined by T_i and T_{i+1}, respectively. They represent the bilinear form in the weak formulation and each matrix entry is computed by evaluating this bilinear form using the geometric information from the underlying mesh. As a reminder, we give a recap to the Galerkin method: In general, to solve a PDE, it is considered in its weak formulation defined on a Sobolev space $H^m(\Omega)$ or $H_0^m(\Omega)$. The solution is approximated in a finite dimensional subspace of this Sobolev space, the finite element space $\mathcal{V}_h(\Omega)$. Consider for example the elliptic zero-boundary value problem

$$\begin{aligned} -\sum_{i,k=1}^{n} \partial_i(a_{ik}\partial_k u) + a_0 u &= f \quad \text{in } \Omega \\ u &= 0 \quad \text{on } \partial\Omega, \end{aligned} \tag{15}$$

with $u \in H_0^1(\Omega)$. The function u is a solution to (15) if

$$a(u,v) = \langle f, v\rangle, \quad \text{for all } v \in H_0^1(\Omega). \tag{16}$$

On the finite dimensional subspace $V_h(\Omega)$ we can find a basis of nodal basis functions $\Phi_1, \ldots, \Phi_N$ of the subspace such that (16) is equivalent to

$$a(u_h, \Phi_j) = \langle f, \Phi_j\rangle, \quad j = 1, \ldots, N, \tag{17}$$

with $u_h \in V_h(\Omega)$. Therefore, u_h can be formulated in terms of the basis functions, $u_h = \sum_{j=1}^N u_j \Phi_j$. This approach leads to the system of equations

$$\sum_{k=1}^{N} a(\Phi_k, \Phi_j)u_k = \langle f, \Phi_j\rangle, \quad j = 1, \ldots, N, \tag{18}$$

or in matrix notation $\mathbf{Ku} = \mathbf{f}$, where

$$(\mathbf{K})_{j,k} = a(\Phi_k, \Phi_j) \quad \text{and} \quad (\mathbf{f})_j = \langle f, \Phi_j \rangle.$$

Now, consider the approximate invariant subspace $\mathbf{range}(\mathbf{W}_k^{(i)})$ with $\mathbf{W}_k^{(i)} \in \mathbb{R}^{N_i \times k}$. Instead of considering the coefficients of the matrix in the algebraic sense, we can see the columns of the matrix as vectors containing the coefficients of continuous functions defined on the finite element space $\mathcal{V}_h(\Omega_i, T_i)$. These functions are defined as

$$w_m^{(i)}(x) := \sum_{j=1}^{N_i} (\mathbf{W}_k^{(i)})_{(j,m)} \Phi_j^{(i)}(x), \quad m = 1, \ldots, k, \quad x \in \Omega_i, \tag{19}$$

$\Phi_j^{(i)}(x)$ being the nodal basis functions of the finite element space $\mathcal{V}_h(\Omega_i, T_i)$. If we are able to find a transformation $F(x^{(i+1)}) : \Omega_{i+1} \to \Omega_i$, that maps each node p_l in T_{i+1}, $l = 1, \ldots, N_{i+1}$ to a corresponding point x_l in Ω_i, we can build the matrix representing the mapped approximate invariant subspace, $\widetilde{\mathbf{W}}_k^{(i)} \in \mathbb{R}^{N_{i+1} \times k}$, in the following way:

$$(\widetilde{\mathbf{W}}_k^{(i)})_{(l,m)} := w_m^{(i)}(F(p_l)). \tag{20}$$

In summary, this method consists of two steps: First, we define a map from the nodes of the new mesh to points in the old domain (see Fig. 1); and second, we interpolate the values of the matrix $\mathbf{W}_k^{(i)}$ via the functions defined in (19) in the finite element space and evaluate these functions at the points in Ω_i corresponding to the nodes in T_{i+1}.

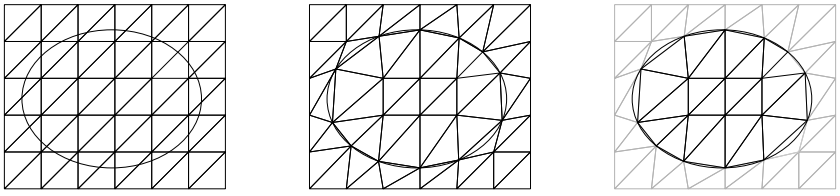
Although the second step is straightforward, the first step can be challenging, depending on the meshing technique that is used. Therefore, we briefly discuss general concepts to realize such a mapping for different meshing techniques. If the update of the shape in the optimization procedure is performed via **mesh morphing** [27], each node in T_{i+1} can be mapped uniquely to the corresponding node in T_i, hence the approximate invariant subspace can be recycled without being transformed. Techniques that work with a reference frame, like **Arbitrary Lagrangian Eulerian (ALE)** [28], come with an inherent mapping from the reference frame to the domain. Through this reference frame, we can map each node in T_{i+1} to a node in T_i and thus recycle the approximate invariant subspace as in the mesh morphing case. For meshing techniques that do not have such an inherent mapping it gets more difficult.

If we assume, for example, a general re-meshing scheme without a prescribed number of nodes or restrictions on the connectivity, then the optimization procedure does not provide any information about the relation between the two domains Ω_i and Ω_{i+1}; see Fig. 1. Additionally, the two meshes generally may differ in the number of nodes and thus basis functions. Therefore, a more sophisticated mapping of the approximate invariant subspace of the linear system matrix derived from mesh T_i to an approximate invariant subspace for the system matrix derived from mesh T_{i+1} becomes inevitable. If no other information is available, the simplest map from Ω_{i+1} to Ω_i is $F(x) := x$. This choice, however, does not guarantee that $F(x) \in \Omega_i$. Therefore, we suggest to choose $F(x)$ as the minimizer, in a suitable norm, of the distance between the given point x and points in Ω_i, i.e., $F(x) := \arg\min_{\tilde{x}_i \in \Omega_i} \|\tilde{x}_i - x\|$, which implies the identity for $x \in \Omega_i$. Another approach would be to extrapolate the $w_m^{(i)}(\cdot)$ in a simple way. We discuss this idea in more detail in Section 4.

A special case of re-meshing is mesh refinement (and derefinement), especially **adaptive mesh refinement** (AMR) [29]. Although in this case the two systems will certainly differ in dimension, the new nodes will definitely lie inside Ω_i. Additionally, for the new nodes the global evaluation in (19) reduces to a local evaluation on the respective refined element; a mapping of approximate invariant subspaces between AMR meshes is described in [2].

3.2. Improving the approximate invariant subspace

While our approach to map an approximate invariant subspace from one optimization step to the next, discussed in the previous subsection, usually provides good approximate invariant subspaces, occasionally the error from the approximation procedure is large enough that recycling the approximate invariant subspace is not effective. In that case, we suggest to use an eigensolver to improve the approximation. For the purpose of effective recycling in the linear solver, we generally need only a modest improvement of the accuracy of the desired recycle space as an approximate invariant subspace [30]. In our experience, a few cycles of subspace iteration or Arnoldi is typically enough. Of course, the cost of the eigensolver must be offset by cost reductions in the linear solver.



(a) (b) (c)

Fig. 2. Grid generation: (a) the structured mesh $\widetilde{T}$ and the domain Ω; (b) the adapted mesh T^{∞}; (c) the active mesh T_a.

4. Mapping on structured meshes

To solve the optimization problem given in Section 2, it is first discretized via the finite element method. Here, the meshing is realized by an approach similar to Composite Finite Elements, first developed in [31–33].

4.1. Structured meshes for evolving geometries

The structured meshing approach that is used in this paper combines some of the best features of re-meshing and mesh morphing techniques. For demonstration purposes, the technique is described in two dimensions but easily extends to three dimensions. For more details see [34].

We assume the feasible area $\widetilde{\Omega}$ to be rectangular. $\widetilde{\Omega}$ is discretized by a regular triangular grid. The grid is denoted by $\widetilde{T}$, the number of elements by $\widetilde{N}^{el}$ and the number of nodes by $\widetilde{N}^{no}$. As in Fig. 2a, the boundary $\delta\Omega_0$ of the shape to be optimized is superimposed onto the grid, represented, for example, by a set of splines. In a second step, we adapt the grid $\widetilde{T}$ to the boundary of the shape by moving the closest vertex of a cut edge onto the boundary [34]. The adapted grid is denoted by T^{∞}. The computations are only performed on the elements inside the domain. We call these elements active elements, making up the active grid T_a, and the corresponding nodes are called active nodes (P_a, with $|P_a| =: N_a$). The active elements are a subset of the elements of T^{∞}, which itself is a perturbation of $\widetilde{T}$. An important advantage of this approach is the possibility to implicitly build the system of equations for all nodes in T^{∞}, with the rows and columns corresponding to the non-active nodes containing only zeros, and then perform the calculation only with the "active" sub-mesh that we denote by T_a.

In this way, after updating the shape, the process starts again, each optimization step, with $\widetilde{T}$, with the same node numbering and connectivity. The mesh is updated according to [34]. This gives a substantial speed-up in building the mesh in comparison with full re-meshing approaches, as only elements at the boundary have to be changed. Additionally, as most of the elements do not change, the system matrix has to be updated only for entries corresponding to nodes in elements that do change; see Fig. 3. Hence, the full assembly of the system matrix in each iteration is avoided. This provides an advantage over both the re-meshing and the mesh-morphing techniques, while obtaining an accuracy comparable to re-meshing approaches.

4.2. Mapping

The structured meshing technique described in the previous section is decidedly well suited for designing a mapping of the type described in Section 3. This is why, as a proof of concept, we introduce a quite simple mapping adjusted for this specific meshing technique, which still leads to a considerable speed up in many of our test cases. Consider the linear systems $\mathbf{K}(\rho^{(i)})\mathbf{u}^{(i)} = \mathbf{f}^{(i)}$ and $\mathbf{K}(\rho^{(i+1)})\mathbf{u}^{(i+1)} = \mathbf{f}^{(i+1)}$ and $\mathbf{W}^{(i)} \in \mathbb{R}^{N_a^{(i)} \times k}$, with $\mathbf{range}(\mathbf{W}^{(i)})$ approximating the invariant subspace corresponding to the k smallest eigenvalues of $\mathbf{K}(\rho^{(i)})$. As in this meshing approach the initial connectivity of the mesh is kept for all optimization steps, we can uniquely identify each node in iteration $(i+1)$ with a node in iteration (i). We distinguish between three cases to determine the matrix $\widetilde{\mathbf{W}}^{(i)}$ representing the mapped approximate invariant subspace, see Fig. 3: (1) Matrix entries corresponding to inner nodes that stay inner nodes are kept; (2) matrix entries corresponding to nodes that change from inner node to boundary node or vice versa are recalculated according to (19); and (3) matrix entries corresponding to nodes that change from inactive to active are calculated in the following way. Assuming only small changes in the geometry, these nodes must be boundary nodes or close to boundary nodes. It is therefore likely that the values of former active

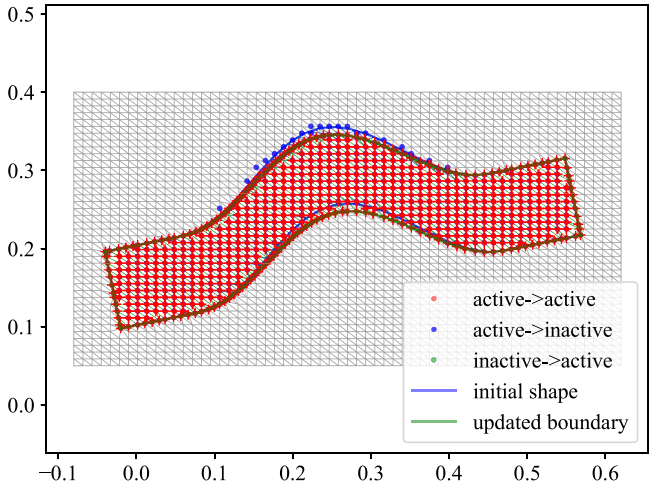


Fig. 3. The status changes for nodes near the boundary for a small change in shape.

Table 1
Cosines of the principal angles ($\cos(\theta_i)$) between the approximate invariant subspace of dimension $k = 15$ and the true invariant subspace corresponding to the 15 smallest eigenvalues for three successive deformations (iteration 1, 2, and 3).

Angle	1	···	8	9	10	11	12	13	14	15
iter 1	0.992	···	0.945	0.939	0.935	0.935	0.913	0.872	0.806	0.150
iter 2	0.983	···	0.904	0.843	0.785	0.722	0.551	0.331	0.153	0.102
iter 3	0.987	···	0.933	0.908	0.899	0.844	0.781	0.700	0.476	0.217

nodes in their neighborhood provide a better approximation than a value resulting from interpolating former active and inactive nodes. We therefore propose for these points the following extrapolation: Consider that the status of node $\tilde{x}_\ell^{(i+1)}$ changes from inactive to active. From the set of nodes that share a finite element, we consider only the subset of nodes that were active in iteration (i). For each active node, $x_\ell^{(i+1)}$ we have such a set $S_a^{(i,\ell)}$. We set the entries of $\widetilde{\mathbf{W}}^{(i)}$ corresponding to these nodes to the weighted mean of the values at $S_a^{(i,\ell)}$, given by

$$(\widetilde{\mathbf{W}}^{(i)})_{(\ell,j)} = \frac{1}{|S_a^{(i,\ell)}| - 1} \sum_{s \in S_a^{(i,\ell)}} \frac{\sum_{\bar{s} \in S_a^{(i,\ell)}} \|\bar{s} - x_\ell^{(i+1)}\|_2 - \|s - x_\ell^{(i+1)}\|_2}{\sum_{\bar{s} \in S_a^{(i,\ell)}} \|\bar{s} - x_\ell^{(i+1)}\|_2} (\mathbf{W}^{(i)})_{(s,j)}, \tag{21}$$

with weights corresponding to the distance between the neighbors' location on the old grid and $x_\ell^{(i+1)}$.

4.3. Test of mapping an approximate invariant subspace

As our first example, we consider a model problem solving the linear elasticity equations on a bent rod with a 181×121 nodes grid, resulting in 5507 active nodes. To get a first impression of the quality of the approximation of the invariant subspace via the mapping, we perform deformation steps of the shape in a controlled way, see Fig. 4, and calculate the principal angles between the resulting mapped approximate invariant subspaces and the true invariant subspaces corresponding to the smallest eigenvalues of the new system matrix, which have been computed for the purpose of comparison only.

We perform three steps of deformation of the original shape, with recycle space dimension $k = 15$, and we calculate the principal angles between the mapped approximate invariant subspace and the invariant subspace corresponding to the 15 smallest eigenvalues according to [35, p. 604]; see Table 1. We see that most of the angles are small, i.e., our approach approximates these spaces quite well. Note that we only consider small geometric deformations.

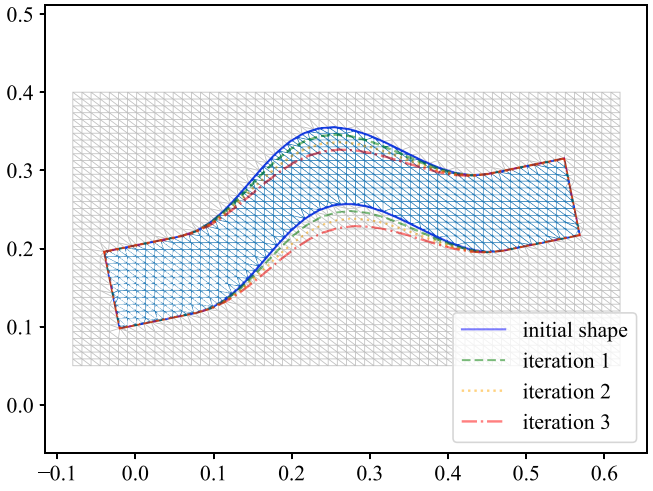
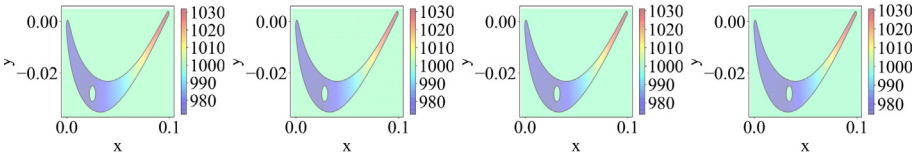


Fig. 4. Sequence of shapes for a model problem.



(a) Initial configuration. (b) Configuration 1. (c) Configuration 2. (d) Configuration 3.

Fig. 5. Test problem 1: Poisson equation on a turbine blade. Four consecutive positions of the hole at $x = 0.025$, $x = 0.028$, $x = 0.031$ and $x = 0.034$.

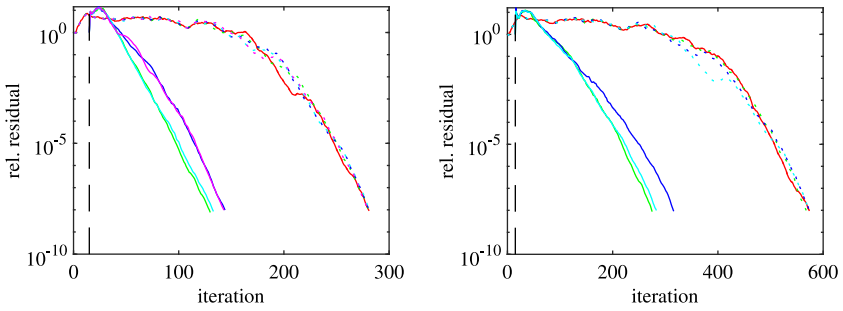
5. Numerical results

To demonstrate the efficacy of the described methods, we consider two two-dimensional examples. The first one concerns a turbine blade on which the Poisson equation is solved. For the second one, we solve the linear elasticity equation on a bent rod.

5.1. Poisson equation on a turbine blade

As a first example, we solve the Poisson equation on a turbine blade, where the change of geometry is caused by a hole, representing a cooling channel, as its position inside the turbine blade is being optimized. In this example, the movement of the hole is artificial and not driven by an optimization. We provide results for two mesh sizes. For the first, $\widetilde{T}$ is a 361×181 grid, which gives close to 12,000 active nodes on T_a, and, for the second, $\widetilde{T}$ is a 722×362 grid yielding a little less than 47,000 active nodes. On the boundary of the blade, Robin-boundary conditions hold with constant heat coefficients, and the temperature on the cooling channel boundary is chosen to be two times lower than the one on the outer boundary. We perform an initial solve using RCG without a recycle space (but computing one for the next solve), and hence the convergence will be the same as for CG, and then three solves with RCG and a recycle space dimension of $k = 15$ for three consecutive changes of the domain; the geometries are visualized in Fig. 5. An incomplete Cholesky factorization, IC(0), preconditioner is used. For comparison, we also provide the convergence of preconditioned CG.

Fig. 6 and Table 2 demonstrate that a speed-up of around 55% in terms of the number of matrix–vector products is obtained (including the initial 15 matrix–vector products required for the recycle space), independent of the problem size. In the smaller example, the number of active nodes does not change due to the fact that the hole is moved in very controlled way and does not change in size. Nevertheless, the rows and columns of the matrix do



(a) 361 × 181 (b) 722 × 362

Fig. 6. Convergence history of recycling CG with recycle space dimension $k = 15$ for solving the Poisson equation for four different positions of the hole in the blade using meshes of two different sizes. Dotted lines indicate the convergence of CG and solid lines the convergence of recycling CG. Red is used for the initial system, green for system 1, blue for system 2 and cyan for system 3. The dashed vertical line indicates the $k = 15$ matrix–vector multiplications that have to be invested before the recycling CG method can start (if there is a recycle space). (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

Table 2
Number of unknowns, number of iterations and number of nodes changing status from active to inactive and vice versa for solving the Poisson equation for four different positions of the hole in the blade using underlying meshes of two different sizes. The dimension of the recycle space is $k = 15$ for both the small problem and the larger problem.

Opt. step	361 × 181			722 × 362		
	N	#its (#matvecs)	Active (inactive)	N	#its (#matvecs)	Active (inactive)
0	11,893	279 (279)		46,668	572 (572)	
1	11,893	114 (129)	269 (269)	46,671	259 (274)	999 (996)
2	11,893	128 (143)	269 (269)	46,671	300 (315)	997 (997)
3	11,893	117 (132)	269 (269)	46,669	267 (282)	996 (998)

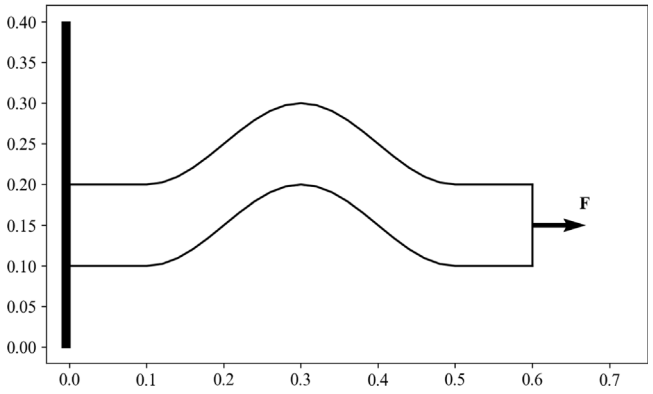


Fig. 7. Test problem 2: Linear elasticity problem for ceramic object under tensile load.

not represent the same nodes in the region of the hole, as indicated by the number of nodes in the regular grid changing between active and inactive states.

5.2. Gradient based shape optimization with linear elasticity as governing PDE

In our second example, we consider a bent rod that is clamped at the left, i.e., with zero-boundary conditions on the left boundary, and a tensile load is applied on the right boundary, i.e., Neumann-boundary conditions are applied on the right side; see Fig. 7. In this example, the deformations of the shape of the rod are not artificially generated but originate from an optimization procedure using shape derivatives [36]. We assume that the rod is made from a

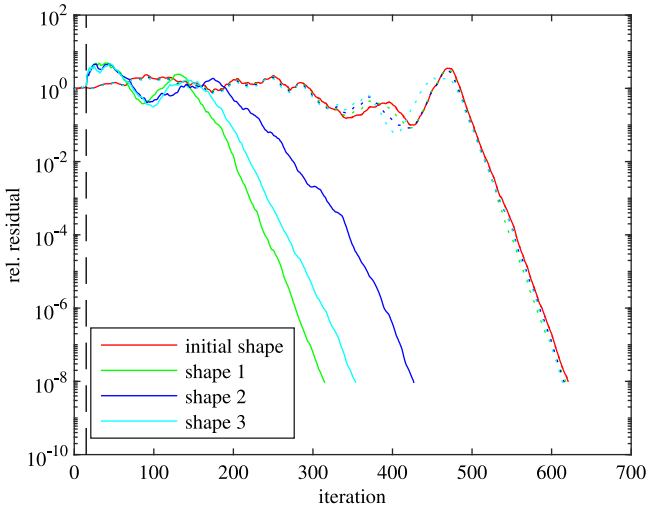


Fig. 8. Convergence history of recycling CG for solving the elasticity equation as needed in four consecutive optimization steps using a mesh of size 301 × 201. Dotted lines indicate the convergence of CG and solid lines that of recycling CG. Red is used for the initial system, green for system 1, blue for system 2 and cyan for system 3. The dashed vertical line indicates the $k = 15$ matrix–vector multiplications that have to be invested before the recycling CG method can start (if there is a recycle space). (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

Table 3
Number of unknowns, number of iterations and number of nodes changing status from active to inactive and vice versa for solving the elasticity equation for four consecutive optimization steps using underlying meshes of two different sizes.

Opt. step	N	#its (#matvecs)	Active (inactive)
0	30,108	619 (619)	
1	30,062	299 (314)	629 (644)
2	30,030	411 (426)	641 (641)
3	29,984	338 (353)	624 (644)

ceramic material, in this case Al_2O_3. Ceramic is a linear elastic material. The shape of the rod is being optimized to maximize its reliability under the applied tensile load. The reliability of the rod is measured by a functional giving the probability of failure of the rod under a given tensile load,

$$J(\Omega, u) := \frac{\Gamma(\frac{d}{2})}{2\pi^{\frac{d}{2}}} \int_{\Omega} \int_{S^{d-1}} \left(\frac{(\mathbf{n} \cdot \boldsymbol{\sigma}(u)\mathbf{n})^+}{\sigma_0} \right)^m d\mathbf{n} dx, \tag{22}$$

where $\Omega \subseteq \mathbb{R}^d$ is the domain, $u \in H^1(\Omega, \mathbb{R}^d)$ is the solution of the governing linear elasticity equation, $B(u, v) = L(v)$, $\forall v \in H_0^1(\Omega, \mathbb{R}^d)$, $\sigma(u)$ is the stress tensor, $m \geq 2$ is the Weibull modulus, and σ_0 is some positive constant. For simplicity, $m = 2$ is assumed in this example. The differentiability of the functional is shown in [37]. We use the discrete adjoint method to calculate the shape derivative of the Lagrangian; for more details see [38]. $\widetilde{T}$ is a 301 × 201 grid that leads to approximately 15,000 active nodes, and therefore about 30,000 unknowns in the linear elasticity equation. The calculation of the gradient is based on the Steklov–Poincaré type metric introduced in [39]. The rod is straightening during the iteration process. As the resulting linear systems are ill-conditioned, we use IC(0)-preconditioning as in the previous example. Three RCG solves are performed after the initial RCG solve (starting without a recycle space) for four consecutive configurations in the optimization process. The dimension of the recycle space is $k = 15$.

The convergence is visualized in Fig. 8, and details can be found in Table 3. Here we observe an average reduction of the necessary iterations by approximately 43% compared with preconditioned CG.

6. Conclusion

In this paper, we have introduced a new approach to recycle information from the Krylov subspaces of previous systems in shape optimization. In contrast to other approaches previously considered in the literature, in shape optimization we often face a varying number of unknowns, and the mapping from the unknowns in the algebraic systems to the meshes may not be consistent from one optimization step to the next. This makes it difficult to map subspaces from one optimization step to the next. We deal with these problems by considering the vectors spanning the recycle space as coefficient vectors of finite element functions on the underlying finite element mesh. The corresponding finite element functions are used to find the nodal values on the new mesh, possibly in combination with extrapolation, if a new active node is outside the previous active mesh. The resulting vectors span the approximate invariant subspace for the new system matrix.

The numerical results demonstrate the efficacy of our finite element-based approach in two different cases: (1) When the number of unknowns does not change due to small changes in the domain, but the active and inactive nodes do, and (2) when the number of unknowns does change due to larger deformations of the domain under consideration. In both cases the number of iterations necessary to solve the sequence of systems has been substantially reduced.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Funding and acknowledgments

This work was supported by the federal ministry of research and education of Germany (BMBF, grant-no: 05M18PXA) as a part of the GIVEN consortium. This material is based upon work supported by the National Science Foundation under Grant No. 1720305. Sandia National Laboratories is a multimission laboratory managed and operated by National Technology & Engineering Solutions of Sandia, LLC, a wholly owned subsidiary of Honeywell International Inc., for the U.S. Department of Energy's National Nuclear Security Administration under contract DE-NA0003525. This paper describes objective technical results and analysis. Any subjective views or opinions that might be expressed in the paper do not necessarily represent the views of the U.S. Department of Energy or the United States Government.

The authors would like to thank the anonymous reviewers, whose suggestions helped to substantially improve this manuscript.

References

[1] S. Wang, E. de Sturler, G.H. Paulino, Large-scale topology optimization using preconditioned krylov subspace methods with recycling, Internat. J. Numer. Methods Engrg. 69 (2007) 2441–2468.

[2] S. Wang, Krylov Subspace Methods for Topology Optimization on Adaptive Meshes (Ph.D. thesis), University of Illinois at Urbana-Champaign, Department of Computer Science, Advisor: Eric de Sturler, 2007.

[3] S. Wang, E. de Sturler, G.H. Paulino, Dynamic Adaptive Mesh Refinement for Topology Optimization, Tech. rep., 2009, arxiv.org ar Xiv:1009.4975 URL https://arxiv.org/abs/1009.4975.

[4] R.A. Nicolaides, Deflation of conjugate gradients with applications to boundary value problems, SIAM J. Numer. Anal. 24 (2) (1987) 355–365.

[5] Z. Dostál, Conjugate gradient method with preconditioning by projector, Int. J. Comput. Math. 23 (3–4) (1988) 315–323, doi: 10.1080/00207168808803625.

[6] Y. Saad, M. Yeung, J. Erhel, F. Guyomarc'h, A deflated version of the conjugate gradient algorithm, SIAM J. Sci. Comput. 21 (5) (2000) 1909–1926.

[7] L. Motta Mello, E. de Sturler, G. Paulino, E.C. Nelli Silva, Recycling krylov subspaces for efficient large-scale electrical impedance tomography, Comput. Methods Appl. Mech. Engrg. 199 (2010) 3101–3110.

[8] M.E. Kilmer, E. de Sturler, Recycling subspace information for diffuse optical tomography, SIAM J. Sci. Comput. 27 (6) (2006) 2140–2166, http://dx.doi.org/10.1137/040610271.

[9] M.R. Hestenes, E. Stiefel, Methods of conjugate gradients for solving linear systems, J. Res. Natl. Bureau Stand. 49 (1952) 409–436, (1953).

[10] C.C. Paige, M.A. Saunders, Solutions of sparse indefinite systems of linear equations, SIAM J. Numer. Anal. 12 (4) (1975) 617–629.

[11] A. Stathopoulos, K. Orginos, Computing and deflating eigenvalues while solving multiple right-hand side linear systems with an application to quantum chromodynamics, SIAM J. Sci. Comput. 32 (1) (2010) 439–462, URL https://www.siam.org/journals/sisc/32-1/72553.html.

[12] D. Darnell, R.B. Morgan, W. Wilcox, Deflated GMRES for systems with multiple shifts and multiple right hand sides, Linear Algebra Appl. 429 (2008) 2415–2434, http://dx.doi.org/10.1016/j.laa.2008.04.019.

[13] L. Feng, P. Benner, J.G. Korvink, Subspace recycling accelerates the parametric macro-modeling of MEMS, Internat. J. Numer. Methods Engrg. 94 (1) (2013) 84–110.

[14] P. Gosselet, C. Rey, J. Pebrel, Total and selective reuse of Krylov subspaces for the resolution of sequences of nonlinear structural problems, Internat. J. Numer. Methods Engrg. 94 (1) (2013) 60–83, http://dx.doi.org/10.1002/nme.4441.

[15] A. Amritkar, E. de Sturler, K. Świrydowicz, D. Tafti, K. Ahuja, Recycling krylov subspaces for CFD applications and a new hybrid recycling solver, J. Comput. Phys. 303 (2015) 222–237, http://dx.doi.org/10.1016/j.jcp.2015.09.040.

[16] K. Ahuja, P. Benner, E. de Sturler, L. Feng, Recycling BiCGSTAB with an application to parametric model order reduction, SIAM J. Sci. Comput. 37 (5) (2015) S429–S446, http://dx.doi.org/10.1137/140972433.

[17] L.M. Carvalho, S. Gratton, R. Lago, X. Vasseur, A flexible generalized conjugate residual method with inner orthogonalization and deflated restarting, SIAM J. Matrix Anal. Appl. 32 (4) (2011) 1212–1235, http://dx.doi.org/10.1137/100786253.

[18] K. Carlberg, V. Forstall, R. Tuminaro, Krylov-subspace recycling via the POD-augmented conjugate-gradient method, SIAM J. Matrix Anal. Appl. 37 (3) (2016) 1304–1336.

[19] G.B.D. Cortes, C. elis Vuik, J.-D. Jansen, Accelerating the solution of linear systems appearing in two-phase reservoir simulation by the use of POD-based deflation methods, Comput. Geosci., 0000. http://dx.doi.org/10.1007/s10596-021-10041-6.

[20] P. Jolivet, P.-H. Tournier, Block iterative methods and recycling for improved scalability of linear solvers, in: SC16 - Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis, IEEE Press, 2016, http://dx.doi.org/10.1109/SC.2016.16.

[21] F.-X. Roux, A. Barka, Block Krylov recycling algorithms for FETI-2LM applied to 3-d electromagnetic wave scattering and radiation, IEEE Trans. Antennas and Propagation 65 (4) (2017) 1886–1895, http://dx.doi.org/10.1109/TAP.2017.2670541.

[22] H. Al Daas, L. Grigori, P. Hénon, P. Ricoux, Recycling Krylov subspaces and truncating deflation subspaces for solving sequences of linear systems, ACM Trans. Math. Software 47 (2) (2021) 13:1–13:30.

[23] K.M. Soodhalter, E. de Sturler, M.E. Kilmer, A survey of subspace recycling iterative methods, GAMM Mitteilungen, 43, (4), 0000. http://dx.doi.org/10.1002/gamm.202000016.

[24] H.A. van der Vorst, Iterative Krylov Methods for Large Linear Systems, in: Cambridge Monographs on Applied and Computational Mathematics, vol. 13, Cambridge University Press, Cambridge, 2009.

[25] C.C. Paige, B.N. Parlett, H.A. van der Vorst, Approximate solutions and eigenvalue bounds from Krylov subspaces, Numer. Linear Algebra Appl. 2 (2) (1995) 115–134.

[26] Y. Saad, Iterative Methods for Sparse Linear Systems, Vol. 82, SIAM, 2003.

[27] M. Staten, S. Owen, S. Shontz, A. Salinger, T. Coffey, A comparison of mesh morphing methods for 3D shape optimization, in: Proceedings of the 20th International Meshing Roundtable, 2011, pp. 293–311.

[28] C. Hirt, A. Amsden, J. Cook, An arbitrary Lagrangian-Eulerian computing method for all flow speeds, J. Comput. Phys. 14 (1974) 227–253.

[29] M.J. Berger, J. Oliger, Adaptive mesh refinement for hyperbolic partial differential equations, J. Comput. Phys. 53 (1984) 84–512.

[30] M.L. Parks, E. de Sturler, G. Mackey, D.D. Johnson, S. Maiti, Recycling Krylov subspaces for sequences of linear systems, SIAM J. Sci. Comput. 28 (5) (2006) 1651–1674, http://dx.doi.org/10.1137/040607277.

[31] W. Hackbusch, S. Sauter, Adaptive composite finite elements for the solution of PDEs containing nonuniformely distributed micro-scales, Matem. Mod. 8 (1996) 31–43.

[32] W. Hackbusch, S. Sauter, Composite finite elements for problems containing small geometric details, Comput. Vis. Sci. 1 (1997) 15–25.

[33] W. Hackbusch, S. Sauter, Composite finite elements for the approximation of PDEs on domains with complicated micro-structures, Numer. Math. 75 (1997) 447–472.

[34] M. Bolten, C. Hahn, Structured meshes for PDE constrained shape optimization, in preparation.

[35] G.H. Golub, C.F. van Loan, Matrix Computations, third ed., The Johns Hopkins University Press, Baltimore and London, 1996.

[36] J. Haslinger, R.A.E. Mäkinen, Introduction to Shape Optimization: Theory, Approximation, and Computation, SIAM, Philadelphia, 2003.

[37] M. Bolten, H. Gottschalk, S. Schmitz, Minimal failure probability for ceramic design via shape control, J. Optim. Theory Appl. (2015) 983–1001.

[38] M. Bolten, H. Gottschalk, C. Hahn, M. Saadi, Numerical shape optimization to decrease failure probability of ceramic structures, Comput. Visual Sci., 0000 http://dx.doi.org/10.1007/s00791-019-00315-z.

[39] V. Schulz, M. Siebenborn, K. Welker, Efficient PDE constrained shape optimization based on Steklov–Poincaré-type metrics, SIAM J. Optim. 26 (2016) 2800–2819.