

# Computer Methods in Applied Mechanics and Engineering

## Krylov Subspace Recycling for Evolving Structures

--Manuscript Draft--

<b>Manuscript Number:</b>	CMAME-D-20-01785R1
<b>Article Type:</b>	Research Paper
<b>Keywords:</b>	Krylov Subspace Recycling; Shape optimization; Changing Geometries and Structures; Recycling Conjugate Gradients; Approximate Invariant Subspaces
<b>Corresponding Author:</b>	Eric de Sturler, PhD Virginia Tech Blacksburg, Virginia UNITED STATES
<b>First Author:</b>	Matthias Bolten, PhD
<b>Order of Authors:</b>	Matthias Bolten, PhD Eric de Sturler, PhD Camilla Hahn Michael Lawrence Parks, PhD
<b>Abstract:</b>	<p>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.</p> <p>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 remeshing, even if the geometry changes only a little, the corresponding mesh might differ substantially from the previous one. Obviously, this prevents us from any straightforward mapping of the approximate invariant subspace of the linear system matrix (the focus of recycling in this paper) from one step to the next; similar problems arise for other selected subspaces.</p> <p>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.</p>

## Highlights

- New Krylov subspace recycling method for problems where matrix dimensions change
- New Krylov subspace recycling method for problems with changing meshes and remeshing
- Allows Krylov subspace recycling for problems where this is currently not possible
- Derivation and discussion of the efficient computation of a new recycle space for the recycling Conjugate Gradient algorithm.

# Krylov Subspace Recycling for Evolving Structures <sup>☆</sup>

M. Bolten

*Faculty of Mathematics und Natural Sciences, University of Wuppertal, 42119 Wuppertal,  
Germany*

E. de Sturler\*

*Department of Mathematics, Virginia Tech, Blacksburg, Virginia, 24061, USA*

C. Hahn

*Faculty of Mathematics und Natural Sciences, University of Wuppertal, 42119 Wuppertal,  
Germany*

M.L. Parks

*Center for Computing Research, Sandia National Laboratories, Albuquerque, New Mexico,  
87185, USA*

---

## 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

---

\*Corresponding Author

*Email 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)

1  
2  
3  
4  
5  
6  
7  
8  
9  
10  
11  
12  
13  
14  
15  
16  
17  
18  
19  
20  
21  
22  
23  
24  
25  
26  
27  
28  
29  
30  
31  
32  
33  
34  
35  
36  
37  
38  
39  
40  
41  
42  
43  
44  
45  
46  
47  
48  
49  
50  
51  
52  
53  
54  
55  
56  
57  
58  
59  
60  
61  
62  
63  
64  
65

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:* Moving Structures, Krylov Subspace Recycling, Recycling CG, Shape Optimization

*2010 MSC:* 00-01, 65-10

---

## 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) 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

1  
2  
3  
4  
5 sizes. This problem precludes a straightforward application of Krylov subspace  
6 recycling, since we cannot transfer the algebraic basis vectors spanning an ap-  
7 proximate invariant subspace (or spanning any another useful subspace to re-  
8 cycle) from one linear system to the next. However, we will show that through  
9 the sequence of finite element spaces from which the algebraic linear systems  
10 are derived, we can map approximate algebraic eigenvectors from one linear sys-  
11 tem to the next by mapping (through interpolation or other approximations)  
12 the corresponding approximate eigenvectors, or basis vectors of an approximate  
13 invariant subspace, from one finite element space to the next. In a straight-  
14 forward way, on meshes stemming from adaptive mesh refinement (AMR) and  
15 hence on nested meshes, this idea was explored in [2], however, with relatively  
16 modest improvements for convergence [3]. We consider more general mappings  
17 and corrections in this paper. We will introduce the mapping for a generic mesh  
18 in Section 3. For our numerical examples, we give an example of a mapping for  
19 a specific meshing technique in Section 4, the corresponding numerical examples  
20 are given in Section 5.  
21  
22  
23  
24  
25  
26  
27  
28  
29  
30

## 31 **2. Foundations**

32  
33  
34 In shape optimization, the aim is to optimize a given two- or three-dimensional  
35 shape without changes in its inherent topology in order to maximize or mini-  
36 mize a given objective, for example efficiency or reliability. Depending on the  
37 problem considered, certain material laws have to be fulfilled, which are usually  
38 modeled by PDEs, such as the elasticity equations in structural mechanics or  
39 the heat equation. This leads, in principle, to a PDE constrained optimization  
40 problem, where the variable over which one optimizes is neither a function nor  
41 an element of a finite dimensional space, but a domain, i.e., in most cases a sub-  
42 set of  $\mathbb{R}^2$  or  $\mathbb{R}^3$ . In this paper, we restrict ourselves to domains in  $\mathbb{R}^2$ . In a more  
43 formal way, let  $\Omega$  be a bounded domain with Lipschitz boundary  $\partial\Omega$  contained  
44 in a bounded set  $\tilde{\Omega} \subset \mathbb{R}^2$  such that each such  $\Omega \subseteq \tilde{\Omega}$  is admissible. This domain  
45  $\Omega$  is to be optimized subject to some functional  $J(\Omega, u)$  that depends on  $\Omega$  and  
46  
47  
48  
49  
50  
51  
52  
53  
54  
55  
56  
57  
58  
59  
60  
61  
62  
63  
64  
65

1  
2  
3  
4 the solution of a PDE  $u \in H^1(\Omega)$ , i.e.  
5

$$6 \quad \min_{\Omega} \quad J(\Omega, u) \quad (1)$$

$$7 \quad \text{s.t.} \quad Lu = f.$$

8  
9  
10 We restrict ourselves to changes in the geometry that can be described by a  
11 mapping  $F_t : \Omega \rightarrow \mathbb{R}^2$  that is a perturbation of identity, that is,  
12

$$13 \quad F_t = \text{id} + t\mathcal{V}, \quad t > 0,$$

14  
15 where  $\mathcal{V} \in (H^{1,\infty}(\Omega))^2$  is a velocity field.  
16

### 17 *2.1. Krylov Subspace Recycling*

18  
19 We briefly describe the main ideas behind Krylov subspace recycling and  
20 a recycling Conjugate Gradients algorithm (RCG) that combines a Deflated  
21 Conjugate Gradient algorithm with a very efficient method to compute, on  
22 the side, an approximate invariant subspace for the symmetric positive definite  
23 (SPD) system matrix  $\mathbf{K}$  that can be used for the next linear system. A Deflated  
24 Conjugate Gradient method was first proposed in [4]. The implementation here  
25 follows [5]. The method to compute an approximate invariant subspace on  
26 the side is adopted from [1], which proposes a recycling MINRES, adjusted  
27 for working in the  $\mathbf{K}$ -inner product. An alternative update for the recycle (or  
28 deflation) space is proposed in [5]; however, the update in [5] only uses the  
29 first  $\ell$  (chosen parameter) direction vectors from the new Krylov space, whereas  
30 RCG, following [1], uses the whole new Krylov space to update the recycle  
31 space, proceeding in cycles (see below) to keep memory usage modest. For a  
32 more efficient version of RMINRES see [6]; the first version of RMINRES was  
33 proposed in [7]. The CG and MINRES algorithms were proposed in [8] and [9],  
34 respectively.  
35

36  
37 The purpose of recycling is to reuse a judiciously selected subspace from  
38 the search space generated by earlier linear solves to speed-up the convergence  
39 of subsequent linear solves. Here, we focus on approximate invariant subspaces  
40 associated with small (in absolute value) eigenvalues. Depending on whether the  
41  
42  
43  
44  
45  
46  
47  
48  
49  
50  
51  
52  
53

right hand sides change substantially or not, recycling may also give good initial guesses [7]. For alternative Krylov subspace recycling methods, for a range of applications, and HPC implementations, see [10, 11, 12, 13, 14, 15, 16, 17]. A survey of Krylov subspace recycling methods is given in [18].

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 [5])

---

- 1: Input  $\mathbf{K}$ ,  $\mathbf{W} = [\mathbf{w}_1, \dots, \mathbf{w}_k]$  ( $\text{range}(\mathbf{W})$  is the recycle space), SPD preconditioner  $\mathbf{M}^{-1}$ , and initial guess  $\mathbf{u}_0$  satisfying  $\mathbf{r}_0 = \mathbf{f} - \mathbf{K}\mathbf{u}_0 \perp \mathbf{W}$ .
  - 2: Compute  $\mathbf{z}_0 = \mathbf{M}^{-1}\mathbf{r}_0$ ;  $\mathbf{p}_0 = \mathbf{z}_0 - \mathbf{W}(\mathbf{W}^T\mathbf{K}\mathbf{W})^{-1}\mathbf{W}^T\mathbf{K}\mathbf{z}_0$ .
  - 3: **for**  $i = 1, 2, \dots, \ell$  **do**
  - 4:  $\alpha_{i-1} = \mathbf{r}_{i-1}^T \mathbf{z}_{i-1} / \mathbf{p}_{i-1}^T \mathbf{K} \mathbf{p}_{i-1}$
  - 5:  $\mathbf{u}_i = \mathbf{u}_{i-1} + \alpha_{i-1} \mathbf{p}_{i-1}$
  - 6:  $\mathbf{r}_i = \mathbf{r}_{i-1} - \alpha_{i-1} \mathbf{K} \mathbf{p}_{i-1}$
  - 7:  $\mathbf{z}_i = \mathbf{M}^{-1} \mathbf{r}_i$
  - 8:  $\beta_{i-1} = \mathbf{r}_i^T \mathbf{z}_i / \mathbf{r}_{i-1}^T \mathbf{z}_{i-1}$
  - 9: Solve  $(\mathbf{W}^T \mathbf{K} \mathbf{W}) \mathbf{t}_i = (\mathbf{K} \mathbf{W})^T \mathbf{z}_i$
  - 10:  $\mathbf{p}_i = \beta_{i-1} \mathbf{p}_{i-1} + \mathbf{z}_i - \mathbf{W} \mathbf{t}_i$
  - 11: **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  $\text{range}(\mathbf{W})$  with  $\mathbf{W} \in \mathbb{R}^{n \times k}$  computed in the previous optimization step. We give the preconditioned deflated CG [5] 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) [19] 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 [5].

## 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,  $\mathbf{range}(\mathbf{W})$ . As RCG is optimal over  $\mathbf{range}(\mathbf{W}) \oplus \mathbf{range}([\mathbf{p}_0 \ \mathbf{p}_1 \ \dots \ \mathbf{p}_{\ell-1}])$  (after  $\ell$  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  $\mathbf{P}_j = [\mathbf{p}_{(j-1)m} \ \dots \ \mathbf{p}_{jm-1}] \in \mathbb{R}^{n \times m}$  and  $\bar{\mathbf{P}}_j = [\mathbf{p}_{(j-1)m-1} \ \dots \ \mathbf{p}_{jm}] = [\mathbf{P}_{j-1} \mathbf{e}_m \mid \mathbf{P}_j \mid \mathbf{P}_{j+1} \mathbf{e}_1] \in \mathbb{R}^{n \times (m+2)}$ , with standard basis vectors  $\mathbf{e}_1, \mathbf{e}_m \in \mathbb{R}^m$ ;  $\bar{\mathbf{P}}_1 = [\mathbf{0} \mid \mathbf{P}_1 \mid \mathbf{P}_2 \mathbf{e}_1]$ .

Here, we consider only approximate invariant subspaces using harmonic Ritz vectors [20] 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  $\theta$  if

$$\mathbf{M}^{-1} \mathbf{K} \boldsymbol{\xi} - \theta \boldsymbol{\xi} \perp_{\mathbf{M}} \mathbf{M}^{-1} \mathbf{K} \mathcal{S}. \quad (2)$$

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

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

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

$$\mathbf{G}_j \mathbf{x} = \theta \mathbf{F}_j \mathbf{x}, \quad \text{where} \quad (4)$$

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

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

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 [5] 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  $\mathbf{X}_j = [\mathbf{x}_1 \dots \mathbf{x}_k]$ , and

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

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  $\mathbf{Y}_{j-1}$

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

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

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

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

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

which together give

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

where  $\mathbf{T}_j = [\mathbf{t}_{(j-1)m} \ \mathbf{t}_{(j-1)m+1} \ \dots \ \mathbf{t}_{jm}]$ ,  $\Psi_j^{(m+1) \times m}$  is a lower bidiagonal matrix with the coefficients  $\pm\alpha_{(j-1)m}^{-1}$ ,  $\pm\alpha_{(j-1)m+1}^{-1}$ ,  $\dots$  from (7), and  $\Phi_j^{(m+2) \times (m+1)}$  is a lower bidiagonal matrix with the coefficients  $-\beta_{(j-1)m-1}$ ,  $-\beta_{(j-1)m}$ ,  $\dots$  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

$$\begin{aligned} (\mathbf{K}\mathbf{Y}_{j-1})^T \mathbf{M}^{-1} \mathbf{K}\mathbf{P}_j &= \mathbf{Y}_{j-1}^T \mathbf{K}(\mathbf{W}\mathbf{T}_j + \bar{\mathbf{P}}_j \Phi_j) \Psi_j \\ &= (\mathbf{Y}_{j-1}^T \mathbf{K}\mathbf{W}) \mathbf{T}_j \Psi_j + (\mathbf{Y}_{j-1}^T \mathbf{K}\bar{\mathbf{P}}_j) \Phi_j \Psi_j. \end{aligned} \quad (10)$$

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

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

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

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  $\mathbf{Y}_{j-1}^T \mathbf{K}\bar{\mathbf{P}}_j$ , we have, again using (6),

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

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),  $\text{range}(\mathbf{Y}_{j-2}) \subseteq \text{range}([\mathbf{Y}_{j-3} \ \mathbf{P}_{j-2}])$ , we have  $\mathbf{Y}_{j-2}^T \mathbf{K}\bar{\mathbf{P}}_j = \mathbf{O}$ . For the second block, we have  $\mathbf{P}_{j-1}^T \mathbf{K}\bar{\mathbf{P}}_j = \mathbf{P}_{j-1}^T \mathbf{K}[\mathbf{P}_{j-1} \mathbf{e}_m \mid \mathbf{P}_j \mid \mathbf{P}_{j+1} \mathbf{e}_1] = d_{(j-1)m-1} \mathbf{e}_m \mathbf{e}_1^T$ , 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,

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

$$\text{with } \mathbf{Y}_0^T \mathbf{K}\bar{\mathbf{P}}_1 = \mathbf{W}^T \mathbf{K}\bar{\mathbf{P}}_1 = \mathbf{O}. \quad (14)$$

1  
2  
3  
4 In addition  $(\mathbf{e}_1^{(m+2)})^T \Phi_j \Psi_j = -\frac{\beta_{(j-1)m-1}}{\alpha_{(j-1)m}} (\mathbf{e}_1^{(m)})^T$  with  $\mathbf{e}_1^{(m+2)} \in \mathbb{R}^{m+2}$  and  
5  $\mathbf{e}_1^{(m)} \in \mathbb{R}^m$ .

6  
7 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  
8  $m \times m$  tridiagonal matrix  
9

$$\begin{aligned} 10 \quad (\mathbf{K} \mathbf{P}_j)^T \mathbf{M}^{-1} (\mathbf{K} \mathbf{P}_j) &= (\mathbf{K} \mathbf{P}_j)^T (\mathbf{W} \mathbf{T}_j + \bar{\mathbf{P}}_j \Phi_j) \Psi_j = \mathbf{P}_j^T \mathbf{K} \bar{\mathbf{P}}_j \Phi_j \Psi_j \\ 11 &= [\mathbf{0} \mid \mathbf{D}_j \mid \mathbf{0}] \Phi_j \Psi_j, \end{aligned}$$

12 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  
13  $i = (j-1)m, (j-1)m+1, \dots, jm-1$ , on the diagonal. The coefficients  $d_i, \alpha_i, \beta_i$   
14 have all been computed in the past RCG cycle.  
15

16 Next, we consider the blocks of  $\mathbf{F}_j$ . Just as for the (1,1) block of  $\mathbf{G}_j$ , using  
17 (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  
18 (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  
19 of  $\mathbf{F}_j$  equals  $\mathbf{D}_j$ .  
20

21 Algorithm 2 outlines the (preconditioned) Recycling CG algorithm that in-  
22 cludes the recycle space into the search space. Algorithm 3 gives the computa-  
23 tion of  $\mathbf{G}_j$  and  $\mathbf{F}_j$ .  
24

25 While algorithm 2 allows any subspace to be recycled, we focus here on  
26 approximate invariant subspaces, in particular, those corresponding to small  
27 eigenvalues. Removing the small eigenvalues leads to substantially improved  
28 rates of convergence as the condition number is significantly improved [5]. See  
29 [21] for a general discussion of the link between the condition number and CG  
30 convergence for SPD systems.  
31  
32  
33  
34  
35  
36  
37  
38  
39  
40  
41  
42  
43  
44  
45  
46  
47  
48  
49  
50  
51  
52  
53  
54  
55  
56  
57  
58  
59  
60  
61  
62  
63  
64  
65

---

**Algorithm 2** Recycling CG

---

1: Input: matrix  $\mathbf{K}$ , preconditioner  $\mathbf{M}$ , (max) cycle length  $m$ , recycle space dimension  $k$ , recycle space basis  $\mathbf{W} = [\mathbf{w}_1, \dots, \mathbf{w}_k]$  (possibly empty),  $\mathbf{Y}_0 = \mathbf{W}$ , initial guess  $\mathbf{u}_0$ , and convergence tolerance  $\tau$ .

2:  $\mathbf{r}_0 = \mathbf{f} - \mathbf{K}\mathbf{u}_0$ .

3: **if**  $\mathbf{W}$  is defined (from solving a previous linear system) **then**

4:   Compute  $\widehat{\mathbf{W}} = \mathbf{K}\mathbf{W}$ ;    $\mathbf{L}\mathbf{L}^T = \mathbf{W}^T\widehat{\mathbf{W}}$  (Cholesky decomposition)

5:   Solve  $(\mathbf{L}\mathbf{L}^T)\mathbf{y} = \mathbf{W}^T\mathbf{r}_0$ ;    $\mathbf{u}_0 = \mathbf{u}_0 + \mathbf{W}\mathbf{y}$ ;    $\mathbf{r}_0 = \mathbf{r}_0 - \widehat{\mathbf{W}}\mathbf{y}$

6:    $\mathbf{z}_0 = \mathbf{M}^{-1}\mathbf{r}_0$ ;   Solve  $(\mathbf{L}\mathbf{L}^T)\mathbf{t}_0 = \widehat{\mathbf{W}}^T\mathbf{z}_0$ ;    $\mathbf{p}_0 = \mathbf{z}_0 - \mathbf{W}\mathbf{t}_0$

7: **else**

8:    $\mathbf{z}_0 = \mathbf{M}^{-1}\mathbf{r}_0$ ;    $\mathbf{p}_0 = \mathbf{z}_0$

9: **end if**

10:  $i = 0$  {iteration index};    $j = 1$  {cycle index}

11: **while**  $\|\mathbf{r}_i\|_2 > \tau$  **do**

12:    $i = i + 1$

13:   {Do one step of Preconditioned Deflated CG (Algorithm 1).}

14:    $\alpha_{i-1} = (\mathbf{r}_{i-1}^T\mathbf{z}_{i-1})/(\mathbf{p}_{i-1}^T\mathbf{K}\mathbf{p}_{i-1})$

15:    $\mathbf{u}_i = \mathbf{u}_{i-1} + \alpha_{i-1}\mathbf{p}_{i-1}$ ;    $\mathbf{r}_i = \mathbf{r}_{i-1} - \alpha_{i-1}\mathbf{K}\mathbf{p}_{i-1}$

16:    $\mathbf{z}_i = \mathbf{M}^{-1}\mathbf{r}_i$

17:    $\beta_{i-1} = \mathbf{r}_i^T\mathbf{z}_i/\mathbf{r}_{i-1}^T\mathbf{z}_{i-1}$

18:   Solve  $\mathbf{L}\mathbf{L}^T\mathbf{t}_i = \widehat{\mathbf{W}}^T\mathbf{z}_i$ ;    $\mathbf{p}_i = \beta_{i-1}\mathbf{p}_{i-1} + \mathbf{z}_i - \mathbf{W}\mathbf{t}_i$

19:   **if**  $\text{mod}(i, m) == 0$  **then**

20:     {Update recycle space  $\mathbf{Y}_j$  for use in next linear system.}

21:     Compute  $\mathbf{G}_j, \mathbf{F}_j$  from (4) ff. (Algorithm 3) and solve  $\mathbf{G}_j\mathbf{x} = \theta\mathbf{F}_j\mathbf{x}$ .

22:     Let  $\mathbf{X}_j$  contain  $k$  eigenvectors corresponding to  $k$  smallest magnitude eigenvalues (or an alternative selection, if desired).

23:      $\mathbf{Y}_j = [\mathbf{Y}_{j-1} \ \mathbf{P}_j]\mathbf{X}_j$ ;   { $\mathbf{Y}_0 = \mathbf{W}$  if  $\mathbf{W}$  exists, otherwise empty}

24:      $j = j + 1$

25:   **end if**

26: **end while**

27:  $\mathbf{W} = \mathbf{Y}_j$  {Assign recycle space for the next system.}

---

---

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

---

- 1: **if**  $j == 1$  **then**
- 2:    $\mathbf{F}_1(1, 1) = \mathbf{W}^T \mathbf{K} \mathbf{W}$    (available from RCG)
- 3:    $\mathbf{F}_1(1, 2) = \mathbf{O}^{k \times m}$ ;    $\mathbf{F}_1(2, 1) = \mathbf{O}^{m \times k}$
- 4:    $\mathbf{F}_1(2, 2) = \mathbf{D}_1$    (available from RCG)
- 5:    $\mathbf{G}_1(1, 1) = (\mathbf{K} \mathbf{W})^T \mathbf{M}^{-1} (\mathbf{K} \mathbf{W})$    ( $\mathbf{K} \mathbf{W}$  available from RCG)
- 6:    $\mathbf{G}_1(1, 2) = \mathbf{W}^T \mathbf{K} \mathbf{W} (\mathbf{T}_1 \Psi_1)$    ( $\mathbf{W}^T \mathbf{K} \mathbf{W}$  available from RCG)
- 7:    $\mathbf{G}_1(2, 1) = \mathbf{G}_1(1, 2)^T$
- 8:    $\mathbf{G}_1(2, 2) = [\mathbf{0} | \mathbf{D}_1 | \mathbf{0}] (\Phi_1 \Psi_1)$
- 9:   Solve for  $\mathbf{X}_1$  from  $\mathbf{G}_1 \mathbf{x} = \theta \mathbf{F}_1 \mathbf{x}$
- 10:   Compute  $\mathbf{Y}_1^T \mathbf{K} \mathbf{W}$  from (12);   Compute  $\mathbf{Y}_1^T \mathbf{K} \bar{\mathbf{P}}_2$  from (13)
- 11:    $\mathbf{G}_2(1, 1) = \mathbf{X}_1^T \mathbf{G}_1 \mathbf{X}_1$ ;    $\mathbf{F}_2(1, 1) = \mathbf{X}_1^T \mathbf{F}_1 \mathbf{X}_1$
- 12: **else**
- 13:    $\mathbf{F}_j(1, 2) = \mathbf{O}^{k \times m}$ ;    $\mathbf{F}_j(2, 1) = \mathbf{O}^{m \times k}$
- 14:    $\mathbf{F}_j(2, 2) = \mathbf{D}_j$    (available from RCG)
- 15:   Compute  $\mathbf{G}_j(1, 2)$  from (10)–(13)
- 16:    $\mathbf{G}_j(2, 1) = \mathbf{G}_j(1, 2)^T$
- 17:    $\mathbf{G}_j(2, 2) = [\mathbf{0} | \mathbf{D}_j | \mathbf{0}] (\Phi_j \Psi_j)$
- 18:   Solve for  $\mathbf{X}_j$  from  $\mathbf{G}_j \mathbf{x} = \theta \mathbf{F}_j \mathbf{x}$
- 19:   Compute  $\mathbf{Y}_j^T \mathbf{K} \mathbf{W}$  from (12);   Compute  $\mathbf{Y}_j^T \mathbf{K} \bar{\mathbf{P}}_{j+1}$  from (13)
- 20:    $\mathbf{G}_{j+1}(1, 1) = \mathbf{X}_j^T \mathbf{G}_j \mathbf{X}_j$ ;    $\mathbf{F}_{j+1}(1, 1) = \mathbf{X}_j^T \mathbf{F}_j \mathbf{X}_j$
- 21: **end if**

---

### 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.

#### 3.1. Mapping between successive meshes

Let  $\Omega_i$  and  $\Omega_{i+1}$  be two domains representing two immediately consecutive shapes stemming from an iterative shape optimization procedure, see Figure 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  $N_{i+1}$ , 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

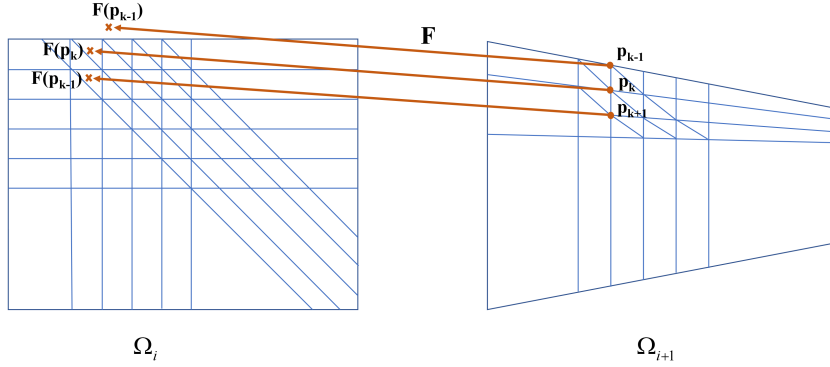


Figure 1: Mapping new mesh nodes to the domain of the previous optimization step.

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  $\Phi_1, \dots, \Phi_N$  of the subspace such that (16) is equivalent to

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

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

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

or in matrix notation  $\mathbf{K}\mathbf{u} = \mathbf{f}$ . Now, consider the approximate invariant subspace  $\text{range}(\mathbf{W}_k^{(i)})$  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

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

$\Phi_j^{(i)}(x)$  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  $F(x^{(i+1)}) : \Omega_{i+1} \rightarrow \Omega_i$ , that maps each node  $p_l$  in  $T_{i+1}$ ,  $l = 1, \dots, N_{i+1}$  to a corresponding point  $x_l$  in  $\Omega_i$ , 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:

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

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  $\Omega_i$  corresponding to the nodes in of  $T_{i+1}$ .

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** [22], 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)** [23], 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  $\Omega_i$  and  $\Omega_{i+1}$ ; see Figure 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  $\Omega_{i+1}$  to  $\Omega_i$  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  $\Omega_i$ , 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) [24]. Although in this case the two systems will certainly differ in dimension, the new nodes will definitely lie inside  $\Omega_i$ . 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, 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 [25]. 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.

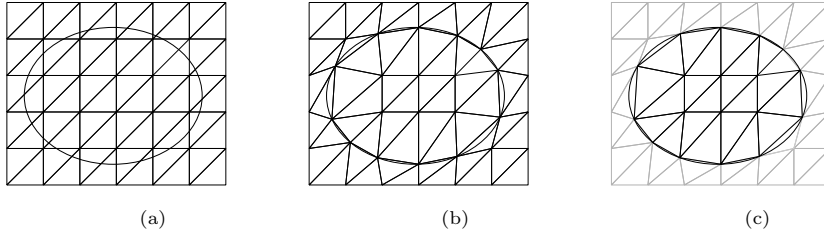


Figure 2: Grid generation: (a) the structured mesh  $\tilde{T}$  and the domain  $\Omega$ ; (b) the adapted mesh  $T^\infty$ ; (c) the active mesh  $T_a$ .

#### 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 [26, 27, 28].

##### 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 [29].

We assume the feasible area  $\tilde{\Omega}$  to be rectangular.  $\tilde{\Omega}$  is discretized by a regular triangular grid. The grid is denoted by  $\tilde{T}$ , the number of elements by  $\tilde{N}^{el}$  and the number of nodes by  $\tilde{N}^{no}$ . As in Figure 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  $\tilde{T}$  to the boundary of the shape by moving the closest vertex of a cut edge onto the boundary [29]. The adapted grid is denoted by  $T^\infty$ . 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  $T^\infty$ , which itself is a perturbation of  $\tilde{T}$ . An important advantage of this approach is the possibility to implicitly build the system of equations for all nodes in  $T^\infty$ , with

1  
2  
3  
4  
5 the rows and columns corresponding to the non-active nodes containing only  
6 zeros, and then perform the calculation only with the “active” sub-mesh that  
7 we denote by  $T_a$ .  
8

9 In this way, after updating the shape, the process starts again, each opti-  
10 mization step, with  $\tilde{T}$ , with the same node numbering and connectivity. The  
11 mesh is updated according to [29]. This gives a substantial speed-up in building  
12 the mesh in comparison with full re-meshing approaches, as only elements at  
13 the boundary have to be changed. Additionally, as most of the elements do  
14 not change, the system matrix has to be updated only for entries corresponding  
15 to nodes in elements that do change. Hence, the full assembly of the system  
16 matrix in each iteration is avoided. This provides an advantage over both the  
17 re-meshing and the mesh-morphing techniques, while obtaining an accuracy  
18 comparable to re-meshing approaches.  
19  
20  
21  
22  
23  
24  
25

#### 26 4.2. Mapping

27 The structured meshing technique described in the previous section is decid-  
28 edly well suited for designing a mapping of the type described in section 3. This  
29 is why, as a proof of concept, we introduce a quite simple mapping adjusted  
30 for this specific meshing technique, which still leads to a considerable speed up  
31 in many of our test cases. Consider the linear systems  $\mathbf{K}(\rho^{(i)})\mathbf{u}^{(i)} = \mathbf{f}^{(i)}$  and  
32  $\mathbf{K}(\rho^{(i+1)})\mathbf{u}^{(i+1)} = \mathbf{f}^{(i+1)}$  and  $\mathbf{W}^{(i)} \in \mathbb{R}^{N_a^{(i)} \times k}$ , with  $\mathbf{range}(\mathbf{W}^{(i)})$  approximating  
33 the invariant subspace corresponding to the  $k$  smallest eigenvalues of  $\mathbf{K}(\rho^{(i)})$ .  
34 As in this meshing approach the initial connectivity of the mesh is kept for  
35 all optimization steps, we can uniquely identify each node in iteration  $(i + 1)$   
36 with a node in iteration  $(i)$ . We distinguish between three cases to determine  
37 the matrix  $\widetilde{\mathbf{W}}^{(i)}$  representing the mapped approximate invariant subspace : (1)  
38 Matrix entries corresponding to inner nodes that stay inner nodes are kept; (2)  
39 matrix entries corresponding to nodes that change from inner node to bound-  
40 ary node or vice versa are recalculated according to (19); and (3) matrix entries  
41 corresponding to nodes that change from inactive to active are calculated in the  
42 following way. Assuming only small changes in the geometry, these nodes must  
43  
44  
45  
46  
47  
48  
49  
50  
51  
52  
53  
54  
55  
56  
57  
58  
59  
60  
61  
62  
63  
64  
65

1  
2  
3  
4  
5 be boundary nodes or close to boundary nodes. It is therefore likely that the  
6 values of former active nodes in their neighborhood provide a better approxima-  
7 tion than a value resulting from interpolating former active and inactive nodes.  
8 We therefore propose for these points the following extrapolation: Consider that  
9 the status of node  $\tilde{x}_\ell^{(i+1)}$  changes from inactive to active. From the set of nodes  
10 that share a finite element, we consider only the subset of nodes that were active  
11 in iteration  $(i)$ . For each active node,  $x_\ell^{(i+1)}$  we have such a set  $S_a^{(i,\ell)}$ . We set  
12 the entries of  $\widetilde{\mathbf{W}}^{(i)}$  corresponding to these nodes to the weighted mean of the  
13 values at  $S_a^{(i,\ell)}$ , given by

$$14 \quad (\widetilde{\mathbf{W}}^{(i)})_{(\ell,j)} = \frac{1}{|S_a^{(i,\ell)}| - 1} \sum_{s \in S_a^{(i,\ell)}} \frac{\sum_m \|s_m - x_\ell^{(i+1)}\|_2 - \|s - x_\ell^{(i+1)}\|_2}{\sum_m \|s_m - x_\ell^{(i+1)}\|_2} (\mathbf{W}^{\infty,(i)})_{(s,j)},$$

15  
16  
17  
18  
19  
20  
21  
22  
23 (21)

24 with weights corresponding to the distance between the neighbors' location on  
25 the old grid and  $x_\ell^{(i+1)}$ .  
26  
27  
28

### 29 4.3. Test of Mapping an Approximate Invariant Subspace

30 As our first example, we consider a model problem solving the linear elas-  
31 ticity equations on a bent rod with a  $181 \times 121$  nodes grid, resulting in 5507  
32 active nodes. To get a first impression of the quality of the approximation of  
33 the invariant subspace via the mapping, we perform deformation steps of the  
34 shape in a controlled way, see Figure 4, and calculate the principal angles of  
35 the resulting mapped approximate invariant subspaces and the true invariant  
36 subspace corresponding to the smallest eigenvalues of the new system matrix,  
37 which has been computed for the purpose of comparison only.  
38  
39  
40  
41  
42  
43

44 We perform three steps of deformation of the original shape, with recycle  
45 space size  $k = 15$ , and we calculate the principal angles between the mapped  
46 approximate invariant subspace and the invariant subspace corresponding to  
47 the 15 smallest eigenvalues according to [30, p. 604]. We can see that most of  
48 the angles are rather small, i.e., our approach approximates these spaces quite  
49 well. Note that we only consider small geometric deformations.  
50  
51  
52  
53  
54  
55  
56  
57  
58  
59  
60  
61  
62  
63  
64  
65

1  
2  
3  
4  
5  
6  
7  
8  
9  
10  
11  
12  
13  
14  
15  
16  
17  
18  
19  
20  
21  
22  
23  
24  
25  
26  
27  
28  
29  
30  
31  
32  
33  
34  
35  
36  
37  
38  
39  
40  
41  
42  
43  
44  
45  
46  
47  
48  
49  
50  
51  
52  
53  
54  
55  
56  
57  
58  
59  
60  
61  
62  
63  
64  
65

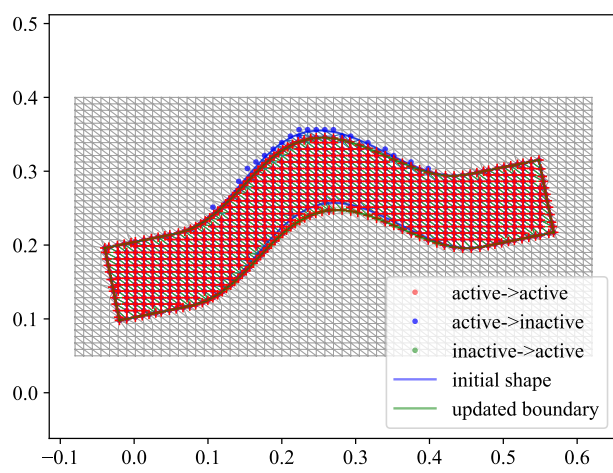


Figure 3: The status changes for nodes near the boundary for a small change in shape.

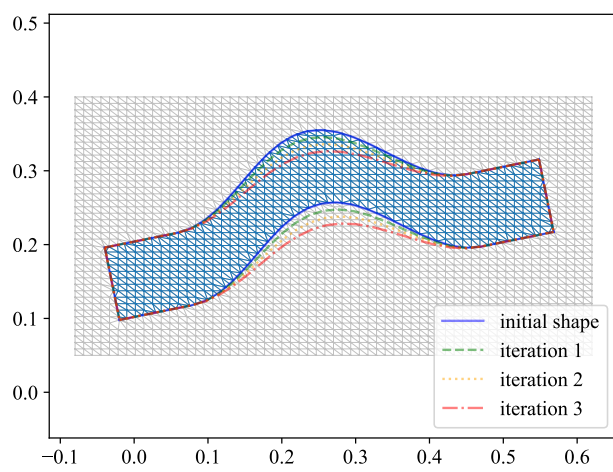


Figure 4: Sequence of shapes for a model problem.

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

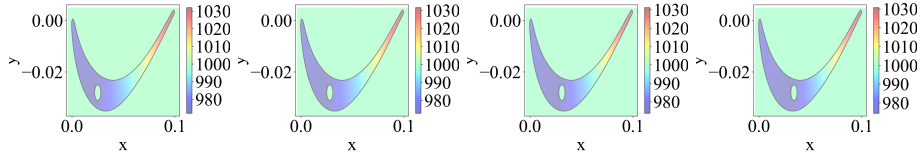
Table 1: Cosines of the principal angles ( $\cos(\theta_i)$ ) between the approximate invariant subspace of dimension  $k = 15$  and the true invariant subspace corresponding to the 15 smallest eigenvalues for three successive deformations (iteration 1, 2, and 3).

## 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,  $\tilde{T}$  is a 361 by 181 grid, which gives close to 12,000 active nodes on  $T_a$ , and, for the second,  $\tilde{T}$  is a 722 by 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 with  $k = 15$  for three consecutive changes of the domain; the geometries are visualized in Figure 5. An incomplete Cholesky factorization,  $IC(0)$ , preconditioner is used. For comparison, we also provide the convergence of preconditioned CG.



(a) Initial configura- (b) Configuration 1. (c) Configuration 2. (d) Configuration 3.  
tion.

Figure 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  $x = 0.034$ .

	$361 \times 181$			$722 \times 362$		
Opt. step	$N$	#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 (131)	269 (269)	46,669	267 (282)	996 (998)

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.

Figure 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 matrix-vector products required for the recycle space), independent from 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 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 state.

### 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

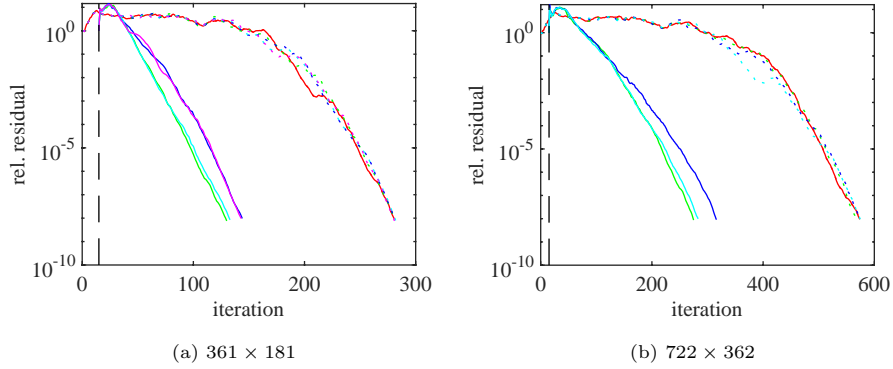


Figure 6: Convergence history of recycling CG with  $k = 15$  for solving the Poisson equation for four different positions of the hole in the blade using meshes of two different sizes. Dotted lines indicate the convergence of CG and solid lines the convergence of recycling CG. Red is used for the initial 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).

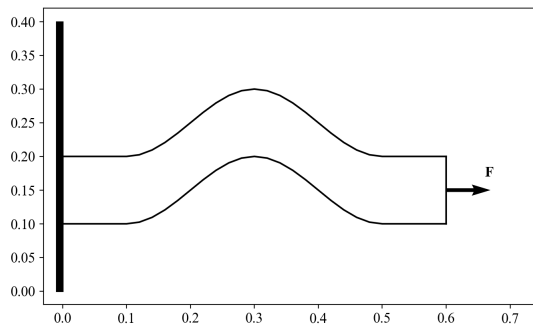


Figure 7: Test problem 2: Linear elasticity problem for ceramic object under tensile load.

1  
 2  
 3  
 4 applied on the right boundary, i.e., Neumann-boundary conditions are applied  
 5 on the right side; see Figure 7. In this example, the deformations of the shape  
 6 of the rod are not artificially generated but originate from an optimization  
 7 procedure using shape derivatives [31]. We assume that the rod is made from a  
 8 ceramic material, in this case  $\text{Al}_2\text{O}_3$ . Ceramic is a linear elastic material. The  
 9 shape of the rod is being optimized to maximize its reliability under the applied  
 10 tensile load. The reliability of the rod is measured by a functional giving the  
 11 probability of failure of the rod under a given tensile load,  
 12  
 13  
 14  
 15  
 16

$$17 \quad J(\Omega, u) := \frac{\Gamma(\frac{d}{2})}{2\pi^{\frac{d}{2}}} \int_{\Omega} \int_{S^{d-1}} \left( \frac{(\mathbf{n} \cdot \boldsymbol{\sigma}(u)\mathbf{n})^+}{\sigma_0} \right)^m d\mathbf{n} dx, \quad (22)$$

18 where  $\Omega \subseteq \mathbb{R}^d$  is the domain,  $u \in H^1(\Omega, \mathbb{R}^d)$  is the solution of the governing  
 19 linear elasticity equation,  $B(u, v) = L(v)$ ,  $\forall v \in H_0^1(\Omega, \mathbb{R}^d)$ ,  $\sigma(u)$  is the stress  
 20 tensor,  $m \geq 2$  is the Weibull modulus, and  $\sigma_0$  is some positive constant. For  
 21 simplicity,  $m = 2$  is assumed in this example. The differentiability of the func-  
 22 tional is shown in [32]. We use the discrete adjoint method to calculate the  
 23 shape derivative of the Lagrangian; for more details see [33].  $\tilde{T}$  is a 301 by  
 24 201 grid that leads to approximately 15,000 active nodes, and therefore about  
 25 30,000 unknowns in the linear elasticity equation. The calculation of the gra-  
 26 dient is based on the Steklov–Poincaré type metric introduced in [34]. The rod  
 27 is straightening during the iteration process. As the resulting linear systems are  
 28 ill-conditioned, we use IC(0)-preconditioning as in the previous example. Three  
 29 RCG solves are performed after the initial RCG solve (starting without a recycle  
 30 space) for four consecutive configurations in the optimization process.  
 31  
 32  
 33  
 34  
 35  
 36  
 37  
 38  
 39  
 40  
 41

42 The convergence is visualized in Figure 8, details can be found in Table 3.  
 43 Here we observe an average reduction of the necessary iterations by approxi-  
 44 mately 43% compared with preconditioned CG.  
 45  
 46  
 47  
 48  
 49  
 50  
 51  
 52  
 53  
 54  
 55  
 56  
 57  
 58  
 59  
 60  
 61  
 62  
 63  
 64  
 65

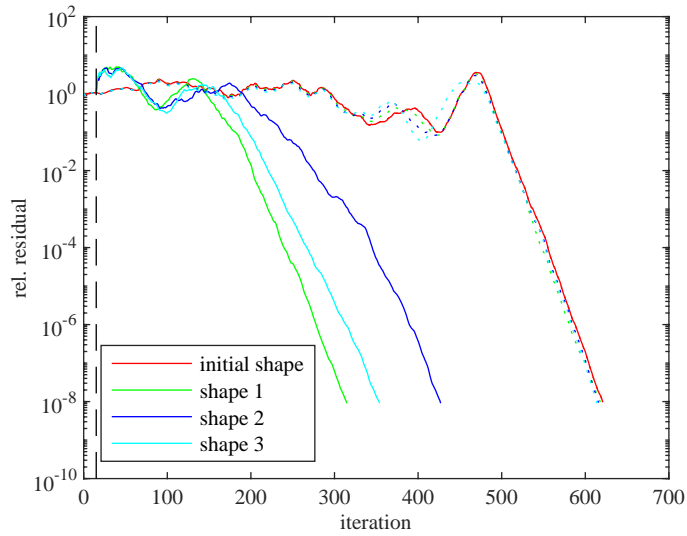


Figure 8: Convergence history of recycling CG for solving the elasticity equation as needed in four consecutive optimization steps using a mesh of size 301 by 201. Dotted lines indicate the convergence of CG and solid lines that of recycling CG. Red is used for the initial system, green for system 1, 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).

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)

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.

1  
2  
3  
4  
5 **6. Conclusion**  
6

7       In this paper, we have introduced a new approach to recycle information from  
8 the Krylov subspaces of previous systems in shape optimization. In contrast to  
9 other approaches previously considered in the literature, in shape optimization  
10 we often face a varying number of unknowns, and the mapping from the un-  
11 knowns in the algebraic systems to the meshes may not be consistent from one  
12 optimization step to the next. This makes it difficult to map subspaces from  
13 one optimization step to the next. We deal with these problems by considering  
14 the vectors spanning the recycle space as coefficient vectors of finite element  
15 functions on the underlying finite element mesh. The corresponding finite el-  
16 element functions are used to find the nodal values on the new mesh, possibly  
17 in combination with extrapolation, if a new active node is outside the previous  
18 active mesh. The resulting vectors span the approximate invariant subspace for  
19 the new system matrix.  
20  
21  
22  
23  
24  
25  
26

27       The numerical results demonstrate the efficacy of our finite element-based  
28 approach in two different cases: (1) When the number of unknowns does not  
29 change due to small changes in the domain, but the active and inactive nodes do,  
30 and (2) when the number of unknowns does change due to larger deformations  
31 of the domain under consideration. In both cases the number of iterations  
32 necessary to solve the sequence of systems has been substantially reduced.  
33  
34  
35  
36  
37

38 **Funding and Acknowledgements**  
39

40  
41       This work was supported by the federal ministry of research and education of  
42 Germany (BMBF, grant-no: 05M18PXA) as a part of the GIVEN consortium.  
43 This material is based upon work supported by the National Science Foundation  
44 under Grant No. 1720305. Sandia National Laboratories is a multimission labo-  
45 ratory managed and operated by National Technology & Engineering Solutions  
46 of Sandia, LLC, a wholly owned subsidiary of Honeywell International Inc., for  
47 the U.S. Department of Energy’s National Nuclear Security Administration un-  
48 der contract DE-NA0003525. This paper describes objective technical results  
49  
50  
51  
52  
53

1  
2  
3  
4 and analysis. Any subjective views or opinions that might be expressed in the  
5 paper do not necessarily represent the views of the U.S. Department of Energy  
6 or the United States Government.  
7

8  
9 The authors would like to thank the anonymous reviewers, whose suggestions  
10 helped to substantially improve this manuscript.  
11

## 12 13 14 **References**

- 15  
16 [1] S. Wang, E. de Sturler, G. H. Paulino, Large-scale topology optimization  
17 using preconditioned Krylov subspace methods with recycling, *Int. J. Numer. Meth. Engng.* 69 (2007) 2441–2468.  
18  
19 [2] S. Wang, Krylov subspace methods for topology optimization on adaptive  
20 meshes, Ph.D. thesis, University of Illinois at Urbana-Champaign, Department  
21 of Computer Science, Advisor: Eric de Sturler (2007).  
22  
23 [3] S. Wang, E. de Sturler, G. H. Paulino, Dynamic adaptive mesh  
24 refinement for topology optimization, Tech. rep., arxiv.org (2009).  
25 doi:arXiv:1009.4975.  
26 URL <https://arxiv.org/abs/1009.4975>  
27  
28 [4] R. A. Nicolaidis, Deflation of conjugate gradients with applications to  
29 boundary value problems, *SIAM J. Numer. Anal.* 24 (2) (1987) 355–365.  
30  
31 [5] Y. Saad, M. Yeung, J. Erhel, F. Guyomatc’h, A deflated version of the  
32 conjugate gradient algorithm, *SIAM J. Sci. Comput.* 21 (5) (2000) 1909–  
33 1926.  
34  
35 [6] L. Motta Mello, E. de Sturler, G. Paulino, E. C. Nelli Silva, Recycling  
36 Krylov subspaces for efficient large-scale electrical impedance tomography,  
37 *Comput. Methods Appl. Mech. Engrg.* 199 (2010) 3101–3110.  
38  
39 [7] M. E. Kilmer, E. de Sturler, Recycling subspace information for dif-  
40 fuse optical tomography, *SIAM J. Sci. Comput.* 27 (6) (2006) 2140–2166.  
41 doi:10.1137/040610271.  
42  
43  
44  
45  
46  
47  
48  
49  
50  
51  
52  
53

- 1  
2  
3  
4  
5 [8] M. R. Hestenes, E. Stiefel, Methods of conjugate gradients for solving linear  
6 systems, *Journal of Research of the National Bureau of Standards* 49 (1952)  
7 409–436 (1953).  
8  
9  
10 [9] C. C. Paige, M. A. Saunders, Solutions of sparse indefinite systems of linear  
11 equations, *SIAM Journal on Numerical Analysis* 12 (4) (1975) 617–629.  
12  
13 [10] P. Jolivet, P.-H. Tournier, Block iterative methods and recycling for im-  
14 proved scalability of linear solvers, in: *SC'16: Proceedings of the Interna-*  
15 *tional Conference for High Performance Computing, Networking, Storage*  
16 *and Analysis*, IEEE Press, 2016, p. 17. doi:10.1109/SC.2016.16.  
17  
18 [11] K. Ahuja, P. Benner, E. de Sturler, L. Feng, Recycling BiCGSTAB with  
19 an application to parametric model order reduction, *SIAM J. Sci. Comput.*  
20 37 (5) (2015) S429–S446. doi:10.1137/140972433.  
21  
22 [12] L. Feng, P. Benner, J. G. Korvink, Subspace recycling accelerates the  
23 parametric macro-modeling of MEMS, *International Journal for Numerical*  
24 *Methods in Engineering* 94 (1) (2013) 84–110.  
25  
26 [13] P. Gosselet, C. Rey, J. Pebrel, Total and selective reuse of Krylov subspaces  
27 for the resolution of sequences of nonlinear structural problems, *Internat.*  
28 *J. Numer. Methods Engrg.* 94 (1) (2013) 60–83. doi:10.1002/nme.4441.  
29  
30 [14] L. M. Carvalho, S. Gratton, R. Lago, X. Vasseur, A flexible general-  
31 ized conjugate residual method with inner orthogonalization and de-  
32 flated restarting, *SIAM J. Matrix Anal. Appl.* 32 (4) (2011) 1212–1235.  
33 doi:10.1137/100786253.  
34  
35 [15] K. Carlberg, V. Forstall, R. Tuminaro, Krylov-subspace recycling via the  
36 POD-augmented conjugate-gradient method, *SIAM J. Matrix Anal. Appl.*  
37 37 (3) (2016) 1304–1336.  
38  
39 [16] A. Amritkar, E. de Sturler, K. Świrydowicz, D. Tafti, K. Ahuja, Recycling  
40 Krylov subspaces for CFD applications and a new hybrid recycling solver,  
41 *J. Comput. Phys.* 303 (2015) 222–237. doi:10.1016/j.jcp.2015.09.040.  
42  
43  
44  
45  
46  
47  
48  
49  
50  
51  
52  
53  
54  
55  
56  
57  
58  
59  
60  
61  
62  
63  
64  
65

- 1  
2  
3  
4  
5 [17] H. Al Daas, L. Grigori, P. Hénon, P. Ricoux, Recycling Krylov subspaces  
6 and truncating deflation subspaces for solving sequences of linear systems,  
7 ACM Transactions on Mathematical Software 47 (2) (2021) 13:1 – 13:30.  
8  
9  
10 [18] K. M. Soodhalter, E. de Sturler, M. E. Kilmer, A survey of  
11 subspace recycling iterative methods, GAMM Mitteilungen 43 (4).  
12 doi:10.1002/gamm.202000016.  
13  
14  
15 [19] H. A. van der Vorst, Iterative Krylov methods for large linear systems,  
16 Vol. 13 of Cambridge Monographs on Applied and Computational Mathe-  
17 matics, Cambridge University Press, Cambridge, 2009, reprint of the 2003  
18 original.  
19  
20  
21  
22 [20] C. C. Paige, B. N. Parlett, H. A. van der Vorst, Approximate solutions and  
23 eigenvalue bounds from Krylov subspaces, Numerical Linear Algebra with  
24 Applications 2 (2) (1995) 115–134.  
25  
26  
27  
28 [21] Y. Saad, Iterative methods for sparse linear systems, Vol. 82, SIAM, 2003.  
29  
30 [22] M. Staten, S. Owen, S. Shontz, A. Salinger, T. Coffey, A comparison of  
31 mesh morphing methods for 3D shape optimization, Proceedings of the  
32 20th International Meshing Roundtable (2011) 293–311.  
33  
34  
35 [23] C. Hirt, A. Amsden, J. Cook†, An arbitrary Lagrangian-Eulerian comput-  
36 ing method for all flow speeds, J. Comput. Phys. 14 (1974) 227–253.  
37  
38  
39 [24] M. J. Berger, J. Olinger, Adaptive mesh refinement for hyperbolic partial  
40 differential equations, J. Comput. Phys. 53 (1984) 84–512.  
41  
42  
43 [25] M. L. Parks, E. de Sturler, G. Mackey, D. D. Johnson, S. Maiti, Recy-  
44 cling Krylov subspaces for sequences of linear systems, SIAM Journal on  
45 Scientific Computing 28 (5) (2006) 1651–1674. doi:10.1137/040607277.  
46  
47  
48 [26] W. Hackbusch, S. Sauter, Adaptive composite finite elements for the so-  
49 lution of PDEs containing nonuniformly distributed micro-scales, Matem.  
50 Mod. 8 (1996) 31–43.  
51  
52  
53  
54  
55  
56  
57  
58  
59  
60  
61  
62  
63  
64  
65

- 1  
2  
3  
4  
5 [27] W. Hackbusch, S. Sauter, Composite finite elements for problems contain-  
6 ing small geometric details, *Comput. Visual. Sci.* 1 (1997) 15–25.  
7  
8 [28] W. Hackbusch, S. Sauter, Composite finite elements for the approximation  
9 of PDEs on domains with complicated micro-structures, *Num. Math.* 75  
10 (1997) 447–472.  
11  
12 [29] M. Bolten, C. Hahn, Structured meshes for PDE constrained shape opti-  
13 mization, in preparation.  
14  
15 [30] G. H. Golub, C. F. van Loan, *Matrix Computations*, 3rd Edition, The  
16 Johns Hopkins University Press, Baltimore and London, 1996.  
17  
18 [31] J. Haslinger, R. A. E. Mäkinen, *Introduction to shape optimization: theory,*  
19 *approximation, and computation*, SIAM, Philadelphia, 2003.  
20  
21 [32] M. Bolten, H. Gottschalk, S. Schmitz, Minimal failure probability for ce-  
22 ramic design via shape control, *J. Optim. Theory Appl.* (2015) 983–1001.  
23  
24 [33] M. Bolten, H. Gottschalk, C. Hahn, M. Saadi, Numerical shape optimiza-  
25 tion to decrease failure probability of ceramic structures, *Comput. Visual*  
26 *Sci.* doi:10.1007/s00791-019-00315-z.  
27  
28 [34] V. Schulz, M. Siebenborn, K. Welker, Efficient PDE constrained shape  
29 optimization based on Steklov–Poincaré-type metrics, *SIAM J. Optim.* 26  
30 (2016) 2800–2819.  
31  
32  
33  
34  
35  
36  
37  
38  
39  
40  
41  
42  
43  
44  
45  
46  
47  
48  
49  
50  
51  
52  
53  
54  
55  
56  
57  
58  
59  
60  
61  
62  
63  
64  
65