

New flexible and inexact Golub-Kahan algorithms for inverse problems

Silvia Gazzola¹ and Malena Sabaté Landman²

¹Dipartimento di Matematica, Università di Pisa, *silvia.gazzola@unipi.it*.

²Mathematical Institute, Oxford, *malena.sabatelandman@maths.ox.ac.uk*

Abstract

This paper introduces a new class of algorithms for solving large-scale linear inverse problems based on new flexible and inexact Golub-Kahan factorizations. The proposed methods iteratively compute regularized solutions by approximating a solution to (re)weighted least squares problems via projection onto adaptively generated subspaces, where the constraint subspaces for the residuals are (formally) equipped with iteration-dependent preconditioners or inexactness. The new solvers offer a flexible and inexact Krylov subspace alternative to other existing Krylov-based approaches for handling general data fidelity functionals, e.g., those expressed in the p -norm. Numerical experiments in imaging applications, such as image deblurring and computed tomography, highlight the effectiveness and competitiveness of the proposed methods with respect to other popular methods.

Keywords. Flexible and inexact Krylov subspace methods, Golub-Kahan factorizations, covariance preconditioning, p -norm data fitting, image deblurring, computed tomography.

1 Introduction

Discrete linear inverse problems are formulated as the solution of large-scale linear systems of equations of the form

$$\mathbf{A}\mathbf{x}_{\text{true}} + \mathbf{n} = \mathbf{b}, \quad (1.1)$$

where the discretized forward operator $\mathbf{A} \in \mathbb{R}^{m \times n}$ is large-scale with ill-determined rank, $\mathbf{x}_{\text{true}} \in \mathbb{R}^n$ is the unknown quantity of interest, and $\mathbf{n} \in \mathbb{R}^m$ are some unknown perturbations (noise) affecting the available data $\mathbf{b} \in \mathbb{R}^m$. Such problems arise in many relevant applications in Science and Engineering, including image deblurring and computed tomography. Due to the ill-posedness of \mathbf{A} and the presence of noise in \mathbf{b} , one should regularize (1.1) in order to recover a meaningful approximation of $\mathbf{x}_{\text{true}} \in \mathbb{R}^n$, i.e., replace the ill-posed linear system by a nearby problem, whose solutions is more stable with respect to perturbations in the data. We refer to [19, 22, 25] for more details on discrete inverse problems and classical regularization methods.

Many iterative solvers that look for an approximation

$$\bar{\mathbf{x}}^* = \operatorname{argmin}_{\mathbf{x} \in \mathbb{R}^n} \frac{1}{2} \|\mathbf{A}\mathbf{x} - \mathbf{b}\|_2^2 \quad (1.2)$$

of \mathbf{x}_{true} in (1.1), including many Krylov subspace methods, are known to have regularizing properties achieved by early termination. Namely, the first iterations are unaffected by noise and the approximate solution approaches the true solution, but this stops once the noisy components and the unwanted solution of the unregularized least squares (LS) problem (1.2) start to be recovered. This type of behavior is usually referred to as semiconvergence; see [20, 22]. Among Krylov methods, the mathematically equivalent CGLS and LSQR can handle rectangular matrices in (1.2) and their

regularizing properties are well understood; see [23] and [21, Chapter 6]. Essentially, when applying such methods to (1.2), we are trying to ‘fit as much of the data, fitting the minimum amount of noise’. Even if such solvers can be efficiently employed with additional (often Tikhonov-like) regularization with the goal of mitigating semiconvergence, for simplicity this is not considered in this paper; see [8] for more details.

In this paper we consider iterative regularization methods that solve generalizations of problem (1.2), whereby some statistical information about the noise \mathbf{n} affecting the data in (1.1) is encoded into the problem formulation to enable more accurate recovery of \mathbf{x}_{true} . More specifically, if $\mathbf{R} \in \mathbb{R}^{m \times m}$ is the noise covariance matrix (symmetric positive definite (SPD) by construction), the well-known Gauss-Markov theorem prescribes to solve the weighted LS problem

$$\bar{\mathbf{x}}^* = \operatorname{argmin}_{\mathbf{x} \in \mathbb{R}^n} 1/2 \|\mathbf{Ax} - \mathbf{b}\|_{\mathbf{R}^{-1}}^2, \quad (1.3)$$

to find the best linear unbiased estimator (BLUE); see [33] and [18, Chapter 10]. Note that this reduces to (1.2) if the noise is Gaussian white since, in this case, $\mathbf{R} = \sigma^2 \mathbf{I}$, where σ is the standard deviation.

In many situations, the covariance matrix \mathbf{R} is not (fully) known but can be learned from the data. Indeed, often \mathbf{R} is assumed to have a parametric form whose parameters can be estimated by solving an optimization problem. However, this usually requires computing intermediate approximations of the solution $\bar{\mathbf{x}}^*$ for an intermediate set of parameters, so these solvers typically rely on nested cycles of iterations (inner for approximating $\bar{\mathbf{x}}^*$, outer for updating \mathbf{R}). Moreover, in some instances, \mathbf{R} might depend explicitly on the noiseless measurements, as in methods that approximate Poisson noise and other heteroskedastic distributions; see, e.g., [1] and [18, Chapter 10]. Similarly, the solution of (1.1), where the noise distribution has an underlying sparsity assumption, can be well recovered by considering a p -norm data fit, so that

$$\bar{\mathbf{x}}^* = \operatorname{argmin}_{\mathbf{x} \in \mathbb{R}^n} 1/p \|\mathbf{Ax} - \mathbf{b}\|_p^p, \quad (1.4)$$

with $0 < p \leq 1$. This is the case, for instance, of impulse or salt-and-pepper noise. More generally, the value of p is linked to the assumption of the noise following a given Generalized Error Distribution, so that, from a statistical viewpoint, solving (1.4) corresponds to computing the maximum likelihood estimator; see [2, 5], and the references therein. Although sophisticated non-smooth and possibly non-convex optimization methods exist to handle these cases (see, e.g., [11, 12, 30]), we consider an Iteratively Reweighted Least Squares (IRLS) approach; see [3, Section 4.5.2] and, specifically for regularized inverse problems, [28, 32].

IRLS methods can be regarded as specific instances of majorization-minimization (MM) methods, where a sequence of majorizing functions for (1.4) is constructed at successive approximations of $\bar{\mathbf{x}}^*$, each obtained by minimizing the corresponding majorant. More specifically, given an approximation $\tilde{\mathbf{x}}$ of a solution of (1.1), the objective function in (1.4) can be approximated by considering the objective function in (1.3) with weights

$$\mathbf{R}^{-1} = (\mathbf{W}(\tilde{\mathbf{x}}))^2 = \operatorname{diag} \left(((\mathbf{A}\tilde{\mathbf{x}} - \mathbf{b})^2 + \tau^2)^{\frac{p-2}{2}} \right). \quad (1.5)$$

Note that all the above operations on vectors are performed entry-wise and the smoothing parameter $\tau > 0$ is considered to avoid possible divisions by zero caused by the lack of smoothness of the objective function in (1.4) at zero. The choice of weights in (1.5) allows to interpret (1.3) as the tangent majorant of a smooth approximation of the original function at $\tilde{\mathbf{x}}$ (ignoring additive and multiplicative constants). Classical IRLS solvers consider a sequence of LS problems of the form (1.3), and iteratively update the matrix \mathbf{R}^{-1} by taking as $\tilde{\mathbf{x}}$ the solution of the previous LS problem (1.3) in the sequence, typically starting from $\mathbf{R}^{-1} = (\mathbf{W}(\mathbf{0}))^2$. Since, in a general large-scale setting, each problem in the sequence is solved by CGLS or LSQR, IRLS solvers are actually inner-outer

iterative schemes, whereby the weights are updated at each outer iteration. However, recently, more efficient IRLS formulations have been proposed, which enable to update the weights only after one iteration of the minimization algorithm has been performed (instead of a full solve). These solvers are typically based on ‘non-standard’ (such as flexible [7, 15] or generalized [26]) Krylov methods, and have been successfully considered to handle nonnegativity constraints in (1.2) [6] (possibly with the inclusion of covariance preconditioning, so to work with (1.3) [17]) and variational regularization methods with a p -norm fit-to-data term (like the one appearing in (1.4)) and a q -norm regularization term; see also [9, 27]. We emphasize that, among these ‘non-standard’ Krylov methods, only the ones based on generalized Krylov subspaces can handle fit-to-data terms expressed in the p -norm.

This paper proposes new solvers for the weighted least squares problem (1.3), with \mathbf{R} at least partially unknown and motivated by the statistics of the noise \mathbf{n} in (1.1). The new solvers are inspired by the IRLS approach to (1.3) and (1.4) described above, but aim to avoid inner-outer iterations by incorporating iteration-dependent approximations of \mathbf{R} on the fly. In order to achieve this, we introduce new instances of the flexible [7] and inexact [16] Golub-Kahan factorizations, linking them in specific cases and showing that, in the current setting, they are mathematically equivalent. Building on these new factorizations we introduce a range of flexible and inexact Krylov methods, based on the basic operations of approximation, projection and differentiation – whose details are explained later in this paper. We show that, in the current framework, it is possible to recover solvers that are mathematically equivalent to some solvers already available in the literature, as well as introduce some new ones. For most of the paper, the terms ‘flexible’ and ‘inexact’ are used interchangeably, and are mostly motivated by the solution of (1.3) and (1.4).

This paper is organized as follows. Section 2 introduces the new flexible and inexact Golub-Kahan factorizations. Section 3 develops the theoretical foundations for, and presents, the proposed methods. In particular, in Section 3.1, standard Krylov solvers for weighed LS problems are recalled, which serve as a baseline for the new solvers. Section 3.2 then presents and analyzes a new general framework for flexible and inexact solvers, providing a unified setting. Finally, Section 4 reports numerical experiments on inverse problems in imaging, including test problems in image deblurring and computed tomography, showing that the proposed solvers are competitive with other IRLS-based alternatives.

Notations

In the following, \mathbf{I}_d denotes the identity matrix of order d . Given a (bolded lower-case) vector $\mathbf{t} \in \mathbb{R}^d$, $[\mathbf{t}]_i$ denotes the i th entry of \mathbf{t} . Given a (bolded upper-case) matrix \mathbf{C} , its entries are denoted by $[\mathbf{C}]_{i,j}$, its columns are denoted by \mathbf{c}_j , and its range is denoted by $\mathcal{R}(\mathbf{C})$. We denote functionals (i.e., functions from \mathbb{R}^d to \mathbb{R}) with lower-case letters, e.g., $g(\mathbf{t})$. When such functionals are evaluated in a subspace of dimension k (of \mathbb{R}^d), and typically at the k th iteration of an iterative solver, we use the notation $g_k(\mathbf{t})$, i.e. these are functions from \mathbb{R}^k to \mathbb{R} for a given parametrization. When such functional is iteration-dependent, at the k th iteration we use the notation $g^{(k)}(\mathbf{t})$. The above notations can be combined, e.g., $g_k^{(k)}(\mathbf{t})$.

2 Flexible and inexact Golub-Kahan factorizations

Consider the matrix $\mathbf{A} \in \mathbb{R}^{m \times n}$ in (1.1), and an initial vector $\hat{\mathbf{u}}$, together with some SPD iteration-dependent preconditioning matrices $\mathbf{R}_1^{-1}, \dots, \mathbf{R}_{k+1}^{-1}$, with $k \ll \min\{m, n\}$ (ideally updated as the steps of the following factorizations proceed).

First we consider the new Flexible Golub-Kahan (FGK) factorization:

$$\mathbf{A}\mathbf{V}_k = \mathbf{U}_{k+1}\mathbf{M}_k, \quad \mathbf{A}^\top \mathbf{Y}_{k+1} = \mathbf{V}_{k+1}\mathbf{T}_{k+1}, \quad (2.1)$$

where $\mathbf{u}_1 = \hat{\mathbf{u}}/\beta$ and $\beta = \|\hat{\mathbf{u}}\|_2$, and

- $\mathbf{U}_{k+1} \in \mathbb{R}^{m \times (k+1)}$ and $\mathbf{V}_{k+1} = [\mathbf{V}_k, \mathbf{v}_{k+1}] \in \mathbb{R}^{n \times (k+1)}$ have orthonormal columns \mathbf{u}_i and \mathbf{v}_i ($i = 1, \dots, k+1$), respectively;
- $\mathbf{Y}_{k+1} = [\mathbf{R}_1^{-1}\mathbf{u}_1, \dots, \mathbf{R}_{k+1}^{-1}\mathbf{u}_{k+1}] \in \mathbb{R}^{m \times (k+1)}$;
- $\mathbf{M}_k \in \mathbb{R}^{(k+1) \times k}$ is upper Hessenberg;
- $\mathbf{T}_{k+1} \in \mathbb{R}^{(k+1) \times (k+1)}$ is upper triangular.

Note that, with respect to the flexible Golub-Kahan factorization introduced in [7], we allow variable preconditioning on the right of \mathbf{A}^\top , rather than on the right of \mathbf{A} .

Second, we consider a new FGK version, mathematically equivalent to (2.1):

$$\mathbf{A}\mathbf{V}_k = \mathbf{U}_{k+1}\mathbf{M}_k, \quad \mathbf{A}^\top \bar{\mathbf{R}}_{k+1} \mathbf{U}_{k+1} = \mathbf{V}_{k+1} \mathbf{T}_{k+1}, \quad (2.2)$$

where

$$\bar{\mathbf{R}}_{k+1} = \sum_{i=1}^{k+1} \mathbf{R}_i^{-1} \mathbf{u}_i \mathbf{u}_i^\top \quad (2.3)$$

and where the matrices \mathbf{V}_k , \mathbf{V}_{k+1} , \mathbf{U}_{k+1} , \mathbf{M}_k and \mathbf{T}_{k+1} are those appearing in (2.1). A model implementation of the new flexible Golub-Kahan factorization is given in Algorithm 1.

Algorithm 1 Flexible Golub-Kahan factorization

Require: $\mathbf{A} \in \mathbb{R}^{m \times n}$, initial vector $\hat{\mathbf{u}} \in \mathbb{R}^m$

- 1: $\mathbf{u}_1 = \hat{\mathbf{u}}/\beta$ with $\beta = \|\hat{\mathbf{u}}\|_2$
- 2: Initialize the preconditioner \mathbf{R}_1^{-1}
- 3: $\mathbf{y}_1 = \mathbf{R}_1^{-1}\mathbf{u}_1$
- 4: $\mathbf{v}_1 = \mathbf{A}^\top \mathbf{y}_1 / [\mathbf{T}_{k+1}]_{1,1}$ with $[\mathbf{T}_{k+1}]_{1,1} = \|\mathbf{A}^\top \mathbf{y}_1\|_2$
- 5: **for** $i = 1, \dots, k$ **do**
- 6: $\mathbf{u} = \mathbf{A}\mathbf{v}_i$
- 7: **for** $j = 1, \dots, i$ **do**
- 8: $\mathbf{u} = \mathbf{u} - [\mathbf{M}_k]_{j,i} \mathbf{u}_j$, with $[\mathbf{M}_k]_{j,i} = \mathbf{u}^\top \mathbf{u}_j$
- 9: **end for**
- 10: $\mathbf{u}_{i+1} = \mathbf{u} / [\mathbf{M}_k]_{i+1,i}$, with $[\mathbf{M}_k]_{i+1,i} = \|\mathbf{u}\|_2$
- 11: Update the preconditioner \mathbf{R}_{i+1}^{-1}
- 12: $\mathbf{y}_{i+1} = \mathbf{R}_{i+1}^{-1} \mathbf{u}_{i+1}$
- 13: $\mathbf{v} = \mathbf{A}^\top \mathbf{y}_{i+1}$
- 14: **for** $j = 1, \dots, i$ **do**
- 15: $\mathbf{v} = \mathbf{v} - [\mathbf{T}_{k+1}]_{j,i+1} \mathbf{v}_j$, with $[\mathbf{T}_{k+1}]_{j,i+1} = \mathbf{v}^\top \mathbf{v}_j$
- 16: **end for**
- 17: $\mathbf{v}_{i+1} = \mathbf{v} / [\mathbf{T}_{k+1}]_{i+1,i+1}$ with $[\mathbf{T}_{k+1}]_{i+1,i+1} = \|\mathbf{v}\|_2$
- 18: **end for**

Ensure: \mathbf{V}_{k+1} , \mathbf{U}_{k+1} , \mathbf{Y}_{k+1} , \mathbf{M}_k , \mathbf{T}_{k+1}

Finally, we consider the following inexact (iGK) version of (2.1), (2.2):

$$\mathbf{A}\mathbf{V}_k = \mathbf{U}_{k+1}\mathbf{M}_k, \quad (\mathbf{A} + \mathcal{E}_{k+1}^{(k)})^\top \mathbf{R}_{k+1}^{-1} \mathbf{U}_{k+1} = \mathbf{V}_{k+1} \mathbf{T}_{k+1}, \quad (2.4)$$

where $\mathcal{E}_{k+1}^{(k)}$ is a matrix incorporating some iteration-dependent inexactness in $\mathbf{A}^\top \mathbf{R}_{k+1}^{-1}$, defined as:

$$\mathcal{E}_{k+1}^{(k)} = \mathbf{R}_{k+1} \left(\sum_{i=1}^{k+1} \mathbf{u}_i \mathbf{u}_i^\top (\mathbf{E}_i^{(k)}) \right) \mathbf{A}, \quad \text{with} \quad \mathbf{E}_i^{(k)} := \mathbf{R}_i^{-1} - \mathbf{R}_{k+1}^{-1}. \quad (2.5)$$

Note that, in the above definition, we have used the facts that both $\mathbf{E}_i^{(k)}$ and \mathbf{R}_{k+1} are symmetric, and that the latter is positive definite. All the matrices $\mathbf{E}_i^{(k)}$, $i = 1, \dots, k+1$, change at each iteration k , leading to an iteration-dependent $\mathcal{E}_{k+1}^{(k)}$. The matrices \mathbf{V}_k , \mathbf{V}_{k+1} , \mathbf{U}_{k+1} , \mathbf{M}_k and \mathbf{T}_{k+1} are again those appearing in (2.1).

A fundamental conceptual difference between flexible factorizations of the kind (2.1), (2.2) and the inexact factorization (2.4) is that, in the former, the matrices \mathbf{R}_i^{-1} are just, formally, iteration-dependent preconditioners while, in the latter, one needs to define inexactness and errors with respect to an ideally exact matrix. Fixing some notion of inexactness is useful also to recast this problem in the framework of the inexact Krylov solvers presented in [16]. Coherently with [16], for reasons that will be clear later (and related to the IRLS framework introduced in Section 1), we regard the SPD matrices $\mathbf{R}_1^{-1}, \dots, \mathbf{R}_k^{-1}$ as inexact versions of \mathbf{R}_{k+1}^{-1} ; this is also reflected in the fact that, in (2.4), \mathbf{R}_{k+1}^{-1} appears as an exact right preconditioner for \mathbf{A}^\top . With respect to the inexact factorizations introduced in [16], (2.4) does not allow inexactness in the matrix-vector products with \mathbf{A} and includes the preconditioner \mathbf{R}_{k+1}^{-1} . However, factorizations (2.1), (2.2) and (2.4) (given the errors defined in (2.5)) are all mathematically equivalent and, in the following, we most often refer to the approximation subspace

$$\begin{aligned}\mathcal{R}(\mathbf{V}_k) &= \mathcal{K}_k(\mathbf{A}^\top \bar{\mathbf{R}}_{k+1} \mathbf{A}, \mathbf{A}^\top \bar{\mathbf{R}}_{k+1} \hat{\mathbf{u}}) \\ &= \mathcal{K}_k((\mathbf{A} + \mathcal{E}_{k+1}^{(k)})^\top \mathbf{R}_{k+1}^{-1} \mathbf{A}, (\mathbf{A} + \mathcal{E}_{k+1}^{(k)})^\top \mathbf{R}_{k+1}^{-1} \hat{\mathbf{u}})\end{aligned}\quad (2.6)$$

associated to any of them as flexible/inexact Krylov subspace, and the solvers defined by imposing that the k th approximate solution \mathbf{x}_k belongs to $\mathcal{R}(\mathbf{V}_k)$ and some constraints on the residual as flexible/inexact Krylov methods.

Finally, for both the factorizations (2.2) and (2.4) (but not (2.1)), we can derive the following flexible/inexact Lanczos-like relationships, respectively:

$$\begin{aligned}\mathbf{A}^\top \bar{\mathbf{R}}_{k+1} \mathbf{A} \mathbf{V}_k &= \mathbf{V}_{k+1} \hat{\mathbf{H}}_k \\ (\mathbf{A} + \mathcal{E}_{k+1}^{(k)})^\top \mathbf{R}_{k+1}^{-1} \mathbf{A} \mathbf{V}_k &= \mathbf{V}_{k+1} \hat{\mathbf{H}}_k\end{aligned}, \quad \text{where} \quad \hat{\mathbf{H}}_k := \mathbf{T}_{k+1} \mathbf{M}_k. \quad (2.7)$$

Recall that $\bar{\mathbf{R}}_{k+1}$ and $\mathcal{E}_{k+1}^{(k)}$ are two alternative ways of expressing the same error or inexactness, and are defined in (2.3) and (2.5), respectively.

Remark 1. Analogously to what already observed in [16], (2.7) is not equivalent to applying inexact Arnoldi to the iteration-dependent system matrix $\mathbf{A}^\top \mathbf{R}_i^{-1} \mathbf{A}$ approximating the exact system matrix $\mathbf{A}^\top \mathbf{R}_{k+1}^{-1} \mathbf{A}$. Indeed, the latter compactly reads:

$$\left(\mathbf{A}^\top \mathbf{R}_{k+1}^{-1} \mathbf{A} + \tilde{\mathcal{E}}_k^{(k)} \right) \mathbf{V}_k = \mathbf{V}_{k+1} \mathbf{H}_k, \quad \text{where} \quad \tilde{\mathcal{E}}_k^{(k)} = \mathbf{A}^\top \sum_{i=1}^k \mathbf{E}_i^{(k)} \mathbf{A} \mathbf{v}_i \mathbf{v}_i^\top,$$

here the matrix $\mathbf{H}_k \in \mathbb{R}^{(k+1) \times k}$ is upper Hessenberg and the matrices $\mathbf{E}_i^{(k)}$ are defined in (2.5). The above factorization can be easily derived starting from an inexact Arnoldi relationship of the kind:

$$[\mathbf{A}^\top \mathbf{R}_1^{-1} \mathbf{A} \mathbf{v}_1, \dots, \mathbf{A}^\top \mathbf{R}_k^{-1} \mathbf{A} \mathbf{v}_k] = \mathbf{V}_{k+1} \mathbf{H}_k,$$

where the matrix on the left can be spelled out as:

$$\mathbf{A}^\top \mathbf{R}_{k+1}^{-1} \mathbf{A} \mathbf{V}_k + \mathbf{A}^\top [\mathbf{E}_1^{(k)} \mathbf{A} \mathbf{v}_1, \dots, \mathbf{E}_k^{(k)} \mathbf{A} \mathbf{v}_k] = \mathbf{A}^\top \mathbf{R}_{k+1}^{-1} \mathbf{A} \mathbf{V}_k + \tilde{\mathcal{E}}_k^{(k)} \mathbf{V}_k.$$

We conclude this section by emphasizing that, even if the factorizations hereby introduced allow any initial vector $\hat{\mathbf{u}} \in \mathbb{R}^m$ to be used, in the following we always assume that $\hat{\mathbf{u}}$ is related to $\mathbf{r}_0 = \mathbf{b} - \mathbf{A} \mathbf{x}_0$, where \mathbf{x}_0 is an initial guess for the solution of (1.1). Moreover, although we refer to the matrices \mathbf{R}_i^{-1} , $i = 1, \dots, k+1$, as preconditioners, in the following they are not intended with

the classical purpose of accelerating the convergence of iterative solvers; they rather stem from the IRLS method applied to (1.3) or (1.4) and, as highlighted in Section 1, their purpose is to improve the quality of the approximations to the solution of (1.1). Still in this setting, the matrices \mathbf{R}_i^{-1} , $i = 1, \dots, k+1$, are updated using residual vectors, noting that, for the considered solvers, the solutions and residuals at the i th iteration, $i \leq k$, can be computed only after \mathbf{v}_{i+1} in line 17 of Algorithm 1 has been computed.

3 Standard, flexible and inexact solvers for general data-fitting problems

We slightly reformulate (1.3) as the problem of computing $\bar{\mathbf{x}}^* = \mathbf{x}_0 + \mathbf{x}^*$, where a correction

$$\mathbf{x}^* = \underset{\mathbf{x} \in \mathbb{R}^n}{\operatorname{argmin}} g(\mathbf{x}) \quad (3.1)$$

is computed to form an improved approximation $\bar{\mathbf{x}}^*$ to \mathbf{x}_{true} in (1.1), and where

$$\begin{aligned} g(\mathbf{x}) &= 1/2 (\mathbf{x}^\top \mathbf{A}^\top \mathbf{R}^{-1} \mathbf{A} \mathbf{x} - 2\mathbf{x}^\top \mathbf{A}^\top \mathbf{R}^{-1} \mathbf{r}_0 + \mathbf{r}_0^\top \mathbf{R}^{-1} \mathbf{r}_0) \\ &= 1/2 \|\mathbf{A} \mathbf{x} - \mathbf{r}_0\|_{\mathbf{R}^{-1}}^2. \end{aligned} \quad (3.2)$$

We will refer to $g(\mathbf{x})$ as *exact functional*. Problem (1.3) is obtained by setting $\mathbf{x}_0 = \mathbf{0}$ in (3.1).

If the matrix \mathbf{R} is given, then one can apply the mathematically equivalent preconditioned CGLS or LSQR to efficiently solve this problem. We will dwell on this case in Section 3.1. However, as explained in Section 1, there are relevant situations where \mathbf{R} is (at least partially) unknown, and we have to update \mathbf{R} based on current approximations to \mathbf{x}_{true} . We will dwell on this case in Section 3.2.

In describing all the solvers we rely on the overarching framework for Krylov solvers introduced in [13]. That is, we classify them as either orthogonal ‘residual’ (O-‘R’) or minimal ‘residual’ (M-‘R’) methods, with the term ‘residual’ to be broadly intended. More precisely, at the k th iteration of each solver, given an approximation subspace $\check{\mathcal{C}}_k$ for the correction \mathbf{x}^* , an approximation subspace $\check{\mathcal{W}}_k$ for the residual correction, and a possible subspace of constraints $\check{\mathcal{V}}_k$ (both dependent on $\check{\mathcal{C}}_k$), we determine a residual update $\check{\mathbf{w}}_k^{\text{O}}$ or $\check{\mathbf{w}}_k^{\text{M}}$ belonging to $\check{\mathcal{W}}_k$ such that

$$\begin{aligned} \check{\mathbf{r}}_0 - \check{\mathbf{w}}_k^{\text{O}} &\perp \check{\mathcal{V}}_k, & \text{for O-‘R’ methods,} \\ \|\check{\mathbf{r}}_0 - \check{\mathbf{w}}_k^{\text{M}}\| &= \min_{\check{\mathbf{w}} \in \check{\mathcal{W}}_k} \|\check{\mathbf{r}}_0 - \check{\mathbf{w}}\|_2, & \text{for M-‘R’ methods.} \end{aligned}$$

Equivalently, $\check{\mathbf{w}}_k^{\text{O}}$ is the oblique projection of $\check{\mathbf{r}}_0$ onto $\check{\mathcal{W}}_k$, orthogonal to $\check{\mathcal{V}}_k$ and $\check{\mathbf{w}}_k^{\text{M}}$ is the orthogonal projection of $\check{\mathbf{r}}_0$ onto $\check{\mathcal{W}}_k$ (so that $\check{\mathcal{V}}_k = \check{\mathcal{W}}_k$). In Table 1 we provide an upfront summary of all the methods considered in this section, specifying the spaces $\check{\mathcal{C}}_k$, $\check{\mathcal{W}}_k$, and $\check{\mathcal{V}}_k$, and the definition of the ‘residual’ $\check{\mathbf{r}}_0$. Note that the naming conventions used in the first column are motivated and detailed in the next subsections.

3.1 Exact Krylov solvers for the exact functional

Solving problem (3.1) for a fixed known \mathbf{R} can be done either using preconditioned CGLS, or preconditioned LSQR, which are mathematically equivalent; see [4, Chapter 4]. In this setting, the ‘preconditioner’ \mathbf{R}^{-1} is equivalently applied to the right of \mathbf{A}^\top or to the left of \mathbf{A} .

In particular, preconditioned CGLS solves the normal equations associated to (1.3) using conjugate gradient. We focus on the interpretation of CGLS as a ‘first differentiate, then project’ approach, i.e., the gradient of $g(\mathbf{x})$ in (3.2) or, equivalently, the normal equation residual, is projected onto a Krylov subspace. Specifically, at the k th iteration, CGLS computes an update for the

Table 1: Summary of all the methods considered in Section 3, using the compact notation introduced in the first row, with \simeq indicating that a stated property only approximately holds. Among the solvers, the following are mathematically equivalent: LSQR and CGLS; DAP, DPA and PDA; APD, PAD and ADP.

NOTATIONS: $\hat{\mathbf{A}}_{k+1}^\varepsilon = (\mathbf{A} + \mathcal{E}_{k+1}^{(k)})^\top \mathbf{R}_{k+1}^{-1} \mathbf{A}$, $(\mathbf{A}^\varepsilon)^\top = (\mathbf{A} + \mathcal{E}_{k+1}^{(k)})^\top \mathbf{R}_{k+1}^{-1}$, $(\mathbf{A}^\varepsilon) = \mathbf{A} + \mathcal{E}_{k+1}^{(k)}$					
solver	category	$\check{\mathcal{C}}_k$	$\check{\mathcal{W}}_k$	$\check{\mathcal{V}}_k$	$\check{\mathbf{r}}_0$
LSQR	M- g	$\mathcal{K}_k(\mathbf{A}^\top \mathbf{R}^{-1} \mathbf{A}, \mathbf{A}^\top \mathbf{R}^{-1} \mathbf{r}_0)$	$\mathbf{R}^{-1/2} \mathbf{A} \check{\mathcal{C}}_k$	$\check{\mathcal{W}}_k$	$\mathbf{R}^{-1/2} \mathbf{r}_0$
CGLS	O- ∇g	$\mathcal{K}_k(\mathbf{A}^\top \mathbf{R}^{-1} \mathbf{A}, \mathbf{A}^\top \mathbf{R}^{-1} \mathbf{r}_0)$	$\mathbf{A}^\top \mathbf{R}^{-1} \mathbf{A} \check{\mathcal{C}}_k$	$\check{\mathcal{C}}_k$	$\mathbf{A}^\top \mathbf{R}^{-1} \mathbf{r}_0$
DAP	O- $\widehat{\nabla g^{(k)}}$	$\mathcal{K}_k(\hat{\mathbf{A}}_{k+1}^\varepsilon, (\mathbf{A}^\varepsilon)^\top \mathbf{r}_0)$	$\hat{\mathbf{A}}_{k+1}^\varepsilon \check{\mathcal{C}}_k$	$\check{\mathcal{C}}_k$	$(\mathbf{A}^\varepsilon)^\top \mathbf{r}_0$
DPA	\simeq O- $\nabla g^{(k)}$	$\mathcal{K}_k(\hat{\mathbf{A}}_{k+1}^\varepsilon, (\mathbf{A}^\varepsilon)^\top \mathbf{r}_0)$	$\mathbf{A}^\top \mathbf{R}_{k+1}^{-1} \mathbf{A} \check{\mathcal{C}}_k$	$\check{\mathcal{C}}_k$	$\mathbf{A}^\top \mathbf{R}_{k+1}^{-1} \mathbf{r}_0$
PDA	\simeq M- $g^{(k)}$	$\mathcal{K}_k(\hat{\mathbf{A}}_{k+1}^\varepsilon, (\mathbf{A}^\varepsilon)^\top \mathbf{r}_0)$	$\mathbf{R}_{k+1}^{-1/2} \mathbf{A} \check{\mathcal{C}}_k$	$\mathbf{R}_{k+1}^{-1/2} (\mathbf{A}^\varepsilon) \check{\mathcal{C}}_k$	$\mathbf{R}_{k+1}^{-1/2} \mathbf{r}_0$
DAP-LSMR	M- $\widehat{\nabla g^{(k)}}$	$\mathcal{K}_k(\hat{\mathbf{A}}_{k+1}^\varepsilon, (\mathbf{A}^\varepsilon)^\top \mathbf{r}_0)$	$\hat{\mathbf{A}}_{k+1}^\varepsilon \check{\mathcal{C}}_k$	$\hat{\mathbf{A}}_{k+1}^\varepsilon \check{\mathcal{C}}_k$	$(\mathbf{A}^\varepsilon)^\top \mathbf{r}_0$
APD, PAD	M- $g^{(k)}$	$\mathcal{K}_k(\mathbf{A}^\top \bar{\mathbf{R}}_{k+1} \mathbf{A}, \mathbf{A}^\top \bar{\mathbf{R}}_{k+1} \mathbf{r}_0)$	$\mathbf{A}^\top \bar{\mathbf{R}}_{k+1}^{(S)} \mathbf{A} \check{\mathcal{C}}_k$	$\check{\mathcal{C}}_k$	$\mathbf{A}^\top \bar{\mathbf{R}}_{k+1}^{(S)} \mathbf{r}_0$
ADP	M- $\nabla \bar{g}^{(k)}$	$\mathcal{K}_k(\mathbf{A}^\top \bar{\mathbf{R}}_{k+1} \mathbf{A}, \mathbf{A}^\top \bar{\mathbf{R}}_{k+1} \mathbf{r}_0)$	$\mathbf{A}^\top \bar{\mathbf{R}}_{k+1}^{(S)} \mathbf{A} \check{\mathcal{C}}_k$	$\check{\mathcal{C}}_k$	$\mathbf{A}^\top \bar{\mathbf{R}}_{k+1}^{(S)} \mathbf{r}_0$

solution $\bar{\mathbf{x}}_k = \mathbf{x}_0 + \mathbf{x}_k$, where $\mathbf{x}_k \in \mathcal{R}(\mathbf{V}_k) = \mathcal{K}_k(\mathbf{A}^\top \mathbf{R}^{-1} \mathbf{A}, \mathbf{A}^\top \mathbf{R}^{-1} \mathbf{r}_0)$, i.e., $\mathbf{x}_k = \mathbf{V}_k \mathbf{s}_k$ and $\mathbf{s}_k \in \mathbb{R}^k$ satisfies

$$\nabla g(\mathbf{x}_k) = \mathbf{A}^\top \mathbf{R}^{-1} (\mathbf{A} \mathbf{x}_k - \mathbf{r}_0) \perp \mathcal{R}(\mathbf{V}_k) \iff \mathbf{V}_k^\top \mathbf{A}^\top \mathbf{R}^{-1} (\mathbf{A} \mathbf{V}_k \mathbf{s}_k - \mathbf{r}_0) = \mathbf{0}. \quad (3.3)$$

According to the framework in [13], preconditioned CGLS is an ‘orthogonal residual’ (OR) method applied to the normal equations of (1.3), so we more properly refer to it as an ‘orthogonal normal equation residual’ (O- ∇g) method.

As such, preconditioned CGLS performs a projection procedure, which determines $\mathbf{A}^\top \mathbf{R}^{-1} \mathbf{A} \mathbf{x}_k$ (i.e., the image of the correction \mathbf{x}_k under $\mathbf{A}^\top \mathbf{R}^{-1} \mathbf{A}$, belonging to $\mathbf{A}^\top \mathbf{R}^{-1} \mathbf{A} \mathcal{K}_k(\mathbf{A}^\top \mathbf{R}^{-1} \mathbf{A}, \mathbf{A}^\top \mathbf{R}^{-1} \mathbf{r}_0)$) by an oblique projection of $\mathbf{A}^\top \mathbf{R}^{-1} \mathbf{r}_0$ (orthogonal to $\mathcal{K}_k(\mathbf{A}^\top \mathbf{R}^{-1} \mathbf{A}, \mathbf{A}^\top \mathbf{R}^{-1} \mathbf{r}_0)$). In the framework presented in [4, 29], preconditioned CGLS is regarded as an orthogonal projection method for the preconditioned normal equations, since it uses $\mathcal{K}_k(\mathbf{A}^\top \mathbf{R}^{-1} \mathbf{A}, \mathbf{A}^\top \mathbf{R}^{-1} \mathbf{r}_0)$ as both approximation subspace for the solution \mathbf{x}_k and constraint subspace for the normal equation residual.

The mathematically equivalent preconditioned LSQR method can be interpreted as a ‘first project, then differentiate’ approach. In this case, we directly minimize $g(\mathbf{x})$ in (3.2) over $\mathcal{R}(\mathbf{V}_k) = \mathcal{K}_k(\mathbf{A}^\top \mathbf{R}^{-1} \mathbf{A}, \mathbf{A}^\top \mathbf{R}^{-1} \mathbf{r}_0)$, i.e., we consider the restriction $\mathbf{x} = \mathbf{V}_k \mathbf{s}$ and optimize over the coefficients $\mathbf{s} \in \mathbb{R}^k$. In formulas, we compute

$$\mathbf{s}_k = \underset{\mathbf{s} \in \mathbb{R}^k}{\operatorname{argmin}} \underbrace{1/2 \|\mathbf{A} \mathbf{V}_k \mathbf{s} - \mathbf{r}_0\|_{\mathbf{R}^{-1}}^2}_{=: g_k(\mathbf{s})} \iff \underbrace{\mathbf{V}_k^\top \mathbf{A}^\top \mathbf{R}^{-1} (\mathbf{A} \mathbf{V}_k \mathbf{s}_k - \mathbf{r}_0)}_{=: \nabla g_k(\mathbf{s}_k)} = \mathbf{0}, \quad (3.4)$$

and take $\mathbf{x}_k = \mathbf{V}_k \mathbf{s}_k$. According to the framework in [13], preconditioned LSQR is a ‘minimal residual’ (MR) method or, equivalently, it is based on a projection procedure that determines $\mathbf{R}^{-1/2} \mathbf{A} \mathbf{x}_k$ by performing an orthogonal projection of $\mathbf{R}^{-1/2} \mathbf{r}_0$ onto $\mathbf{R}^{-1/2} \mathbf{A} \mathcal{K}_k(\mathbf{A}^\top \mathbf{R}^{-1} \mathbf{A}, \mathbf{A}^\top \mathbf{R}^{-1} \mathbf{r}_0)$. Equivalently, $\mathbf{A} \mathbf{x}_k$ is the orthogonal projection of \mathbf{r}_0 onto $\mathbf{A} \mathcal{K}_k(\mathbf{A}^\top \mathbf{R}^{-1} \mathbf{A}, \mathbf{A}^\top \mathbf{R}^{-1} \mathbf{r}_0)$ in the \mathbf{R}^{-1} -weighted inner product. However, since preconditioned LSQR operates on the objective function $g(\mathbf{x})$ in (3.2) (i.e., a weighted residual norm) rather than on the residual norm, we more properly refer to it as a ‘minimal g ’ (M- g) method. In the framework presented in [4, 29], whereby preconditioned LSQR is regarded as an oblique projection method for the preconditioned system having $\mathcal{K}_k(\mathbf{A}^\top \mathbf{R}^{-1} \mathbf{A}, \mathbf{A}^\top \mathbf{R}^{-1} \mathbf{r}_0)$ as approximation subspace for the solution \mathbf{x}_k and $\mathbf{A} \mathcal{K}_k(\mathbf{A}^\top \mathbf{R}^{-1} \mathbf{A}, \mathbf{A}^\top \mathbf{R}^{-1} \mathbf{r}_0)$ as constraint subspace for the preconditioned residual $\mathbf{R}^{-1} (\mathbf{A} \mathbf{x}_k - \mathbf{r}_0)$. Following projection, differentiation is performed in (3.4) to impose the optimality conditions, leading to the solution of

a linear system with SPD coefficient matrix $\mathbf{V}_k^\top \mathbf{A}^\top \mathbf{R}^{-1} \mathbf{A} \mathbf{V}_k$. We refer to $g_k(\mathbf{s})$ in (3.4) as the $\mathcal{R}(\mathbf{V}_k)$ -restricted exact functional.

In summary, for both preconditioned CGLS and LSQR, at iteration k we obtain an approximate solution $\mathbf{x}_k = \mathbf{V}_k \mathbf{s}_k$ of (3.1) by computing the coefficients \mathbf{s}_k that equivalently solve (3.3) and (3.4), respectively. Let us consider the relations in (2.1), (2.2) for fixed $\mathbf{R}_i^{-1} = \mathbf{R}^{-1}$, which simplify to

$$\mathbf{A} \mathbf{V}_k = \mathbf{U}_{k+1} \mathbf{B}_{k+1,k}, \quad \mathbf{A}^\top \mathbf{R}^{-1} \mathbf{U}_{k+1} = \mathbf{V}_{k+1} \mathbf{B}_{k+1}^\top, \quad (3.5)$$

where \mathbf{V}_k , \mathbf{V}_{k+1} , \mathbf{U}_{k+1} are as in (2.1), (2.2), $\mathbf{B}_{k+1} \in \mathbb{R}^{(k+1) \times (k+1)}$ is lower bidiagonal and $\mathbf{B}_{k+1,k}$ is obtained by removing the last column of \mathbf{B}_{k+1} . Exploiting (3.5) and taking $\alpha_1 := [\mathbf{B}_{k+1}]_{1,1}$ and $\beta := \|\mathbf{r}_0\|_2$, both the linear systems on the left of (3.3) and (3.4) simplify to

$$\mathbf{B}_{k+1,k}^\top \mathbf{B}_{k+1,k} \mathbf{s}_k = \alpha_1 \beta \mathbf{e}_1.$$

Note, however, that common implementations of preconditioned CGLS and LSQR do not strictly follow the framework outlined above, with the former relying on three-term recurrences and the latter relying on smart updates of the QR factorization of $\mathbf{B}_{k+1,k}$; see [4, Chapter 4] for more details. In the following, we will refer to solvers that first differentiate and then project as ‘CGLS-like’; we will refer to solvers that first project and then differentiate as ‘LSQR-like’. Coherently with the terminology already adopted in this subsection, whenever we are concerned with the minimization of a functional $g(\mathbf{x})$, in the following we will often refer to $\nabla g(\mathbf{x})$ as ‘normal equations residual’.

3.2 Mixing and matching differentiation, projection, approximation

This section concerns the approximation of the solution of problem (3.1), with an (at least partially) unknown \mathbf{R} in (3.2) that is updated at each iteration of a solver for (3.1). To handle this situation, we present a new framework whereby flexible/inexact projection methods can be formulated: this encompasses some known solvers, and sets the stage for the introduction of new ones. Building on the basic operations of ‘projection’ (P) and ‘differentiation’ (D) already introduced in Section 3.1, we also consider the effect of approximation (A): flexible/inexact Krylov methods differ in the order in which such operations are performed, so that we name each method after permutations of the letters D, A, P. In particular, the relative order of A and D will give rise to different methods, while the relative order between D and P will characterize mathematically equivalent CGLS-like and LSQR-like methods.

3.2.1 DAP, DPA and PDA methods, i.e., ‘classical’ inexact Krylov solvers

As we consider \mathbf{R}_k^{-1} being iteration-dependent, at the k th iteration of a solver for (3.1) we may consider finding an approximate minimizer of the iteration-dependent objective function

$$\begin{aligned} g^{(k)}(\mathbf{x}) &= 1/2 (\mathbf{x}^\top \mathbf{A}^\top \mathbf{R}_{k+1}^{-1} \mathbf{A} \mathbf{x} - 2\mathbf{x}^\top \mathbf{A}^\top \mathbf{R}_{k+1}^{-1} \mathbf{r}_0 + \mathbf{r}_0^\top \mathbf{R}_{k+1}^{-1} \mathbf{r}_0) \\ &= 1/2 \|\mathbf{A} \mathbf{x} - \mathbf{r}_0\|_{\mathbf{R}_{k+1}^{-1}}^2, \end{aligned} \quad (3.6)$$

which we call the *exact functional at the k th iteration*. This choice also matches the IRLS setting mentioned in Section 1, as we would define as current reweighted least squares problem the one with weights computed with respect to the most recent approximation to \mathbf{x}^* in (3.1).

We start by considering a method that first differentiates (D) the objective function (3.6), then approximates (A) its gradient and eventually projects (P) it onto an approximation subspace, i.e., a DAP method. That is, we compute

$$\nabla g^{(k)}(\mathbf{x}) = \mathbf{A}^\top \mathbf{R}_{k+1}^{-1} (\mathbf{A} \mathbf{x} - \mathbf{r}_0) \approx \widehat{\nabla g^{(k)}}(\mathbf{x}) := (\mathbf{A} + \mathcal{E}_{k+1}^{(k)})^\top \mathbf{R}_{k+1}^{-1} (\mathbf{A} \mathbf{x} - \mathbf{r}_0) \quad (3.7)$$

and then apply a projection method involving the approximate gradient (or inexact normal equations residual) $\widehat{\nabla g^{(k)}}(\mathbf{x})$. Specifically, at the k th iteration of such solver, we compute a solution

$$\mathbf{x}_k \in \mathcal{R}(\mathbf{V}_k) = \mathcal{K}_k((\mathbf{A} + \mathcal{E}_{k+1}^{(k)})^\top \mathbf{R}_{k+1}^{-1} \mathbf{A}, (\mathbf{A} + \mathcal{E}_{k+1}^{(k)})^\top \mathbf{R}_{k+1}^{-1} \mathbf{r}_0), \quad (3.8)$$

such that

$$\widehat{\nabla g^{(k)}}(\mathbf{x}_k) = (\mathbf{A} + \mathcal{E}_{k+1}^{(k)})^\top \mathbf{R}_{k+1}^{-1} (\mathbf{A}\mathbf{x}_k - \mathbf{r}_0) \perp \mathcal{R}(\mathbf{V}_k), \quad (3.9)$$

or, equivalently, taking $\mathbf{x}_k = \mathbf{V}_k \mathbf{s}_k$,

$$\mathbf{V}_k^\top (\mathbf{A} + \mathcal{E}_{k+1}^{(k)})^\top \mathbf{R}_{k+1}^{-1} \mathbf{A} \mathbf{V}_k \mathbf{s}_k = \mathbf{V}_k^\top (\mathbf{A} + \mathcal{E}_{k+1}^{(k)})^\top \mathbf{R}_{k+1}^{-1} \mathbf{r}_0. \quad (3.10)$$

In practice, using the factorizations (2.4), the left-hand-side of the above equation reads

$$\mathbf{V}_k^\top (\mathbf{A} + \mathcal{E}_{k+1}^{(k)})^\top \mathbf{R}_{k+1}^{-1} \mathbf{U}_{k+1} \mathbf{M}_k \mathbf{s}_k = \mathbf{V}_k^\top \mathbf{V}_{k+1} \mathbf{T}_{k+1} \mathbf{M}_k \mathbf{s}_k = [\mathbf{I}_k, \mathbf{0}] \mathbf{T}_{k+1} \mathbf{M}_k \mathbf{s}_k,$$

while the right-hand-side reads

$$\mathbf{V}_k^\top (\mathbf{A} + \mathcal{E}_{k+1}^{(k)})^\top \mathbf{R}_{k+1}^{-1} \beta \mathbf{u}_1 = \beta \mathbf{V}_k^\top \mathbf{V}_{k+1} \mathbf{T}_{k+1} \mathbf{e}_1 = \beta t_{1,1} \mathbf{e}_1, \quad (3.11)$$

leading to the following projected linear system

$$\mathbf{T}_{k,k+1} \mathbf{M}_k \mathbf{s}_k = \beta t_{1,1} \mathbf{e}_1, \quad (3.12)$$

where $\mathbf{T}_{k,k+1}$ is \mathbf{T}_{k+1} without its last row, $\beta = \|\mathbf{r}_0\|_2$ and $t_{1,1} = [\mathbf{T}_{k+1}]_{1,1}$. The DAP method is a CGLS-like method and, indeed, the above formulation coincides with the inexact CGLS (iCGLS) method proposed in [16]. In the framework of [13], such DAP method may be regarded as an ‘orthogonal inexact normal equation residual’ ($\text{O-}\widehat{\nabla g^{(k)}}$) method that determines $(\mathbf{A} + \mathcal{E}_{k+1}^{(k)})^\top \mathbf{R}_{k+1}^{-1} \mathbf{A} \mathbf{x}_k$ by performing an oblique projection of $(\mathbf{A} + \mathcal{E}_{k+1}^{(k)})^\top \mathbf{R}_{k+1}^{-1} \mathbf{r}_0$ onto $(\mathbf{A} + \mathcal{E}_{k+1}^{(k)})^\top \mathbf{R}_{k+1}^{-1} \mathbf{A} \mathcal{R}(\mathbf{V}_k)$, orthogonal to $\mathcal{R}(\mathbf{V}_k)$. In the framework of [4, 29] such DAP method is an orthogonal projection method having $\mathcal{R}(\mathbf{V}_k) = \mathcal{K}_k((\mathbf{A} + \mathcal{E}_{k+1}^{(k)})^\top \mathbf{R}_{k+1}^{-1} \mathbf{A}, (\mathbf{A} + \mathcal{E}_{k+1}^{(k)})^\top \mathbf{R}_{k+1}^{-1} \mathbf{r}_0)$ as both approximation subspace for the solution \mathbf{x}_k and constraint subspace for the inexact normal equation residual $\widehat{\nabla g^{(k)}}(\mathbf{x}_k)$.

Equivalently, after differentiating the functional (3.6), we can compute an approximate projection of $\nabla g^{(k)}(\mathbf{x})$ onto the inexact Krylov subspace $\mathcal{R}(\mathbf{V}_k)$, i.e., we differentiate (D), project (P) and then approximate (A), to obtain a DPA method. In formulas, we still take $\mathbf{x}_k = \mathbf{V}_k \mathbf{s}_k \in \mathcal{R}(\mathbf{V}_k)$ as in (3.8) but, differently from (3.9), determine $\mathbf{s}_k \in \mathbb{R}^k$ by imposing the following orthogonality condition on the normal equation residual $\nabla g^{(k)}(\mathbf{x})$ (associated to the minimization of the iteration-dependent functional $g^{(k)}(\mathbf{x})$ in (3.6))

$$\underbrace{\mathbf{A}^\top \mathbf{R}_{k+1}^{-1} (\mathbf{A} \mathbf{x}_k - \mathbf{r}_0)}_{=\nabla g^{(k)}(\mathbf{x}_k)} = \mathbf{A}^\top \mathbf{R}_{k+1}^{-1} (\mathbf{A} \mathbf{V}_k \mathbf{s}_k - \mathbf{r}_0) \perp \mathcal{R}(\mathbf{V}_k). \quad (3.13)$$

Exploiting again the factorizations (2.4), with algebraic manipulations similar to the ones performed for the DAP method, we obtain

$$(\mathbf{T}_{k,k+1} \mathbf{M}_k - \underbrace{\mathbf{V}_k^\top (\mathcal{E}_{k+1}^{(k)})^\top \mathbf{R}_{k+1}^{-1} \mathbf{A} \mathbf{V}_k}_{=:\widehat{\mathbf{E}}_{k+1}}) \mathbf{s}_k = \beta t_{1,1} \mathbf{e}_1.$$

Getting rid of the term $\widehat{\mathbf{E}}_{k+1}$ in the above parenthesis (i.e., considering an inexact projection), we recover problem (3.12). Such DPA method is still a CGLS-like method that, in the framework of [13], determines $\mathbf{A}^\top \mathbf{R}_{k+1}^{-1} \mathbf{A} \mathbf{x}_k$ by performing an inexact oblique projection of $\mathbf{A}^\top \mathbf{R}_{k+1}^{-1} \mathbf{r}_0$ onto $\mathbf{A}^\top \mathbf{R}_{k+1}^{-1} \mathbf{A} \mathcal{R}(\mathbf{V}_k)$, orthogonal to $\mathcal{R}(\mathbf{V}_k)$, and therefore can be regarded as an ‘approximate orthogonal normal equation residual’ ($\simeq \text{O-}\nabla g^{(k)}$) method.

Finally, we may again take $\mathbf{x} = \mathbf{V}_k \mathbf{s}$, with $\mathbf{s} \in \mathbb{R}^k$ and \mathbf{V}_k as in (3.8), in the iteration-dependent exact functional $g^{(k)}(\mathbf{x})$ in (3.6), and consider a projection (P) analogous to the one underlying the M-g method in Section 3.1. That is, we take

$$\mathbf{s}_k = \underset{\mathbf{s} \in \mathbb{R}^k}{\operatorname{argmin}} g_k^{(k)}(\mathbf{s}), \quad \text{where} \quad g_k^{(k)}(\mathbf{s}) = 1/2 \|\mathbf{A} \mathbf{V}_k \mathbf{s} - \mathbf{r}_0\|_{\mathbf{R}_{k+1}^{-1}}^2,$$

which we call *restricted exact functional at the k th iteration*. We then differentiate (D) to impose the optimality condition, to compute $\mathbf{s}_k \in \mathbb{R}^k$ by solving

$$\mathbf{V}_k^\top \mathbf{A}^\top \mathbf{R}_{k+1}^{-1} \mathbf{A} \mathbf{V}_k \mathbf{s}_k - \mathbf{V}_k^\top \mathbf{A}^\top \mathbf{R}_{k+1}^{-1} \mathbf{r}_0 = \mathbf{0}. \quad (3.14)$$

Similarly to the DPA method, we rewrite

$$\underbrace{(\mathbf{V}_k^\top (\mathbf{A} + \mathcal{E}_{k+1}^{(k)})^\top \mathbf{R}_{k+1}^{-1} \mathbf{A} \mathbf{V}_k)}_{\mathbf{T}_{k,k+1} \mathbf{M}_k} - \underbrace{\mathbf{V}_k^\top (\mathcal{E}_{k+1}^{(k)})^\top \mathbf{R}_{k+1}^{-1} \mathbf{A} \mathbf{V}_k}_{=: \widehat{\mathbf{E}}_{k+1}} \mathbf{s}_k = \beta t_{1,1} \mathbf{e}_1$$

and approximate (A) the above equation by dropping the term $\widehat{\mathbf{E}}_{k+1}$, eventually recovering again the projected problem (3.12). Therefore the PDA method so defined is mathematically equivalent to the previous DAP and DPA methods, and can be regarded as the LSQR-like version of them, or as an ‘approximate minimal $g^{(k)}$ ’ (\simeq M- $g^{(k)}$) method. In the framework of [13], the quantity $\mathbf{R}_{k+1}^{-1/2} \mathbf{A} \mathbf{x}_k$ is now obtained as an oblique (rather than orthogonal) projection of $\mathbf{R}_{k+1}^{-1/2} \mathbf{r}_0$ onto $\mathbf{R}_{k+1}^{-1/2} \mathbf{A} \mathcal{R}(\mathbf{V}_k)$, orthogonal to $\mathbf{R}_{k+1}^{-1/2} (\mathbf{A} + \mathcal{E}_{k+1}^{(k)}) \mathcal{R}(\mathbf{V}_k)$. Equivalently, $\mathbf{A} \mathbf{x}_k$ is now obtained as an oblique (rather than orthogonal) projection of \mathbf{r}_0 onto $\mathbf{A} \mathcal{R}(\mathbf{V}_k)$, orthogonal to $(\mathbf{A} + \mathcal{E}_{k+1}^{(k)}) \mathcal{R}(\mathbf{V}_k)$, in the \mathbf{R}_{k+1}^{-1} -weighted inner product induced. These expressions clearly reflect that the obliqueness of the projections comes only from the inexactness (as they would be orthogonal if $\mathcal{E}_{k+1}^{(k)} = \mathbf{0}$, and would coincide with the ones associated to LSQR applied to the minimization of $g^{(k)}(\mathbf{x})$ in (3.6)).

A potential shortcoming of all these methods is that, further elaborating on (3.11), the right-hand-side vector in formulations (3.10) and (3.12), and all the mathematically equivalent versions of them, reads

$$\beta t_{1,1} \mathbf{e}_1 = \beta \mathbf{V}_k^\top (\mathbf{A} + \mathcal{E}_{k+1}^{(k)})^\top \mathbf{R}_{k+1}^{-1} \mathbf{u}_1 = \beta \mathbf{V}_k^\top \mathbf{A}^\top \mathbf{R}_1^{-1} \mathbf{u}_1 = \mathbf{V}_k^\top \mathbf{A}^\top \mathbf{R}_1^{-1} \mathbf{r}_0,$$

so that the right-hand-side vector appearing in the projected inexact normal equations (3.12) is effectively always computed using the initial \mathbf{R}_1^{-1} and may be quite different from the vector $\mathbf{V}_k^\top \mathbf{A}^\top \mathbf{R}_{k+1}^{-1} \mathbf{r}_0$ that appears in the projected normal equations (3.14) associated to the iteration-dependent exact functional $g^{(k)}(\mathbf{x})$.

Remark 2. Note that $\widehat{\nabla g^{(k)}}(\mathbf{x})$ cannot be regarded as the exact gradient of a certain functional $\widehat{g}^{(k)}(\mathbf{x})$ somewhat related to $g^{(k)}(\mathbf{x})$, as the matrix $(\mathbf{A} + \mathcal{E}_{k+1}^{(k)})^\top \mathbf{R}_{k+1}^{-1} \mathbf{A}$ appearing in its first term is not symmetric in general; this is the case only if $(\mathcal{E}_{k+1}^{(k)})^\top \mathbf{R}_{k+1}^{-1} \mathbf{A}$ is symmetric. For this fundamental reason, performing approximation before differentiation leads to a different class of flexible/inexact solver that is described in the next subsection.

Remark 3. The main focus of this paper is on minimal residual methods, where at least the classical methods described in Section 3.1 minimize the (preconditioned) residual norm when the solution is restricted to a given subspace. However, the DAP and the other mathematically equivalent methods derived so far in this subsection can also be used to define an inexact LSMR method where, at the k th iteration, one determines the solution \mathbf{x}_k by imposing the condition (3.8) (to restrict \mathbf{x}_k to the inexact Krylov approximation subspace) and by replacing the orthogonality condition (3.9) by

$$\widehat{\nabla g^{(k)}}(\mathbf{x}) = (\mathbf{A} + \mathcal{E}_{k+1}^{(k)})^\top \mathbf{R}_{k+1}^{-1} (\mathbf{A} \mathbf{x} - \mathbf{r}_0) \perp (\mathbf{A} + \mathcal{E}_{k+1}^{(k)})^\top \mathbf{R}_{k+1}^{-1} \mathbf{A} \mathcal{R}(\mathbf{V}_k). \quad (3.15)$$

Using the new inexact GK factorization (2.4), the above conditions lead to

$$\mathbf{x}_k = \mathbf{V}_k \mathbf{s}_k, \quad \text{where} \quad (\mathbf{T}_{k+1} \mathbf{M}_k)^\top (\mathbf{T}_{k+1} \mathbf{M}_k) \mathbf{s}_k = \beta t_{1,1} (\mathbf{T}_{k+1} \mathbf{M}_k)^\top \mathbf{e}_1.$$

We refer to the method so defined as DAP-LSMR. In the framework of [13], DAP-LSMR determines $(\mathbf{A} + \mathcal{E}_{k+1}^{(k)})^\top \mathbf{R}_{k+1}^{-1} \mathbf{A} \mathbf{x}_k \in (\mathbf{A} + \mathcal{E}_{k+1}^{(k)})^\top \mathbf{R}_{k+1}^{-1} \mathbf{A} \mathcal{R}(\mathbf{V}_k)$ as an orthogonal projection of $(\mathbf{A} + \mathcal{E}_{k+1}^{(k)})^\top \mathbf{R}_{k+1}^{-1} \mathbf{r}_0$ onto $(\mathbf{A} + \mathcal{E}_{k+1}^{(k)})^\top \mathbf{R}_{k+1}^{-1} \mathbf{A} \mathcal{R}(\mathbf{V}_k)$, and therefore can be regarded as a ‘minimal $\widehat{\nabla g^{(k)}}$ ’ (M- $\widehat{\nabla g^{(k)}}$) method.

Remark 4. The DAP, DPA and PDA methods do not enjoy any clear optimality property (i.e., in terms of minimization of inexact normal equation residuals $\widehat{\nabla g^{(k)}}(\mathbf{x})$ in the approximation subspace $\mathcal{R}(\mathbf{V}_k)$ for the solution). This was already observed in [16] for the iCGLS method, which is mathematically equivalent to the current DAP, DPA and PDA methods. Because of the oblique projection of $(\mathbf{A} + \mathcal{E}_{k+1}^{(k)})^\top \mathbf{R}_{k+1}^{-1} \mathbf{r}_0$ happening in (3.9), such methods can be regarded as inexact FOM versions, applied to the inexact normal equations. As explained in Remark 3, the DAP-LSMR method minimizes the norm of the inexact normal equation residuals $\widehat{\nabla g^{(k)}}(\mathbf{x})$ over the same subspace as the DAP, DPA and PDA methods and, therefore, may be regarded as an inexact GMRES applied to the inexact normal equations. For this reason, one may exploit known results about relations between the FOM and GMRES residual (norms) to assess how far the DAP, DPA and PDA inexact normal residual norms are from being optimal; see, e.g., [10].

Inexactness estimates We conclude this section by providing inexactness estimates for the inexact normal equation residuals computed by the DAP, DPA and PDA methods, i.e., given an approximation $\mathbf{x}_k \in \mathcal{R}(\mathbf{V}_k)$, we bound the difference between the exact $\nabla g^{(k)}(\mathbf{x}_k)$ and the inexact $\widehat{\nabla g^{(k)}}(\mathbf{x}_k)$, both appearing in (3.7). To do so, we adapt the bounds already provided in [16, 31]. Taking $\mathbf{E}_i^{(k)}$ as in (2.5),

$$\begin{aligned} \nabla g^{(k)}(\mathbf{x}_k) - \widehat{\nabla g^{(k)}}(\mathbf{x}_k) &= -(\mathcal{E}_{k+1}^{(k)})^\top \mathbf{R}_{k+1}^{-1} \mathbf{A} \mathbf{V}_k \mathbf{s}_k + (\mathcal{E}_{k+1}^{(k)})^\top \mathbf{R}_{k+1}^{-1} \mathbf{r}_0 \\ &= -\mathbf{A}^\top [\mathbf{E}_1^{(k)} \mathbf{u}_1, \dots, \mathbf{E}_{k+1}^{(k)} \mathbf{u}_{k+1}] \mathbf{M}_k \mathbf{s}_k + \mathbf{A}^\top \mathbf{E}_1^{(k)} \mathbf{r}_0 \end{aligned}$$

we straightforwardly derive the bound

$$\|\nabla g^{(k)}(\mathbf{x}_k) - \widehat{\nabla g^{(k)}}(\mathbf{x}_k)\| \leq \|\mathbf{A}^\top \mathbf{E}_1^{(k)} \mathbf{r}_0\| + \sum_{i=1}^{k+1} \|\mathbf{A}^\top \mathbf{E}_i^{(k)}\| \|\bar{\mathbf{s}}_k\|_i, \quad (3.16)$$

where $\bar{\mathbf{s}}_k = \mathbf{M}_k \mathbf{s}_k$. Note that the theoretical bound in (3.16) can be very expensive to compute: at the k th iteration it requires k additional matrix-vector products with \mathbf{A}^\top . Therefore, in practice, we can also consider computing the following cheaper but looser bound

$$\|\nabla g^{(k)}(\mathbf{x}_k) - \widehat{\nabla g^{(k)}}(\mathbf{x}_k)\| \leq \|\mathbf{A}^\top\| \|\mathbf{E}_1^{(k)} \mathbf{r}_0\| + \sum_{i=1}^{k+1} \|\mathbf{A}^\top\| \|\mathbf{E}_i^{(k)}\| \|\bar{\mathbf{s}}_k\|_i.$$

Here $\|\mathbf{A}^\top\|$ can be estimated by running only a few (standard) Golub-Kahan bidiagonalization iterations with \mathbf{A} , which could be also used to find an initial approximation \mathbf{x}_0 of $\bar{\mathbf{x}}^*$ in (1.3), (1.4).

Remark 5. Inexactness estimates are important for understanding how far the original problem is from the one being solved in practice. When the mismatch is too great, this can be rectified by means of a restart. In practice, this can have a great impact in the quality of the reconstructed solution. For this type of classical inexact methods, this is discussed in depth [16], where different restarting strategies are also specified. Note that the mismatch for the DAP, DPA and PDA methods is always measured in terms of the normal equations residual norm.

3.2.2 APD, ADP and PAD solvers, i.e., new flexible Krylov solvers

In this subsection, we propose a new flexible/inexact Krylov framework, where the exact objective function $g(\mathbf{x})$ in (3.2) (or the exact functional at the k th iteration $g^{(k)}(\mathbf{x})$ in (3.6)) is first approximated (A) by an iteration-dependent convex functional $\bar{g}^{(k)}(\mathbf{x})$, and then differentiated (D). As in the previous subsection, projection (P) can either happen before or after differentiation, leading to LSQR-like and CGLS-like solvers, respectively, which are again mathematically equivalent (in the current framework). With respect to the methods presented in the previous subsection, the

methods derived in the present subsection have the advantage of working with explicit optimality conditions for $\bar{g}^{(k)}(\mathbf{x})$, meaning that we can characterize the solution at each iteration as the minimizer of $\bar{g}^{(k)}(\mathbf{x})$ restricted to a solution space whose dimension increases with the iterations. For some constructions of $\bar{g}^{(k)}(\mathbf{x})$, this allows us to theoretically control the difference between the value of the iteration-dependent objective function at each iteration and a desired exact function $f(\mathbf{x})$; see Proposition 1. Note that, in contrast, the traditional inexact methods in Section 3.2.1 can only query inexact gradient norms, i.e., inexact normal equations residual norms.

We start by defining the following *inexact objective function at the k th iteration*

$$\begin{aligned}\bar{g}^{(k)}(\mathbf{x}) &:= 1/2 \left(\mathbf{x}^\top \mathbf{A}^\top \bar{\mathbf{R}}_{k+1} \mathbf{A} \mathbf{x} - \mathbf{x}^\top \mathbf{A}^\top \bar{\mathbf{R}}_{k+1} \mathbf{r}_0 - \mathbf{x}^\top \mathbf{A}^\top \bar{\mathbf{R}}_{k+1}^\top \mathbf{r}_0 + \mathbf{r}_0^\top \bar{\mathbf{R}}_{k+1} \mathbf{r}_0 \right) \\ &= 1/2 \left(\mathbf{x}^\top \mathbf{A}^\top \bar{\mathbf{R}}_{k+1}^{(S)} \mathbf{A} \mathbf{x} - 2\mathbf{x}^\top \mathbf{A}^\top \bar{\mathbf{R}}_{k+1}^{(S)} \mathbf{r}_0 + \mathbf{r}_0^\top \bar{\mathbf{R}}_{k+1}^{(S)} \mathbf{r}_0 \right) \\ &= 1/2 \left\| \mathbf{A} \mathbf{x} - \mathbf{r}_0 \right\|_{\bar{\mathbf{R}}_{k+1}^{(S)}}^2,\end{aligned}\tag{3.17}$$

where the matrix $\bar{\mathbf{R}}_{k+1}$ is defined as in (2.3) and $\bar{\mathbf{R}}_{k+1}^{(S)} = 1/2(\bar{\mathbf{R}}_{k+1} + \bar{\mathbf{R}}_{k+1}^\top)$ is its symmetric part.

Compared to the function $g^{(k)}(\mathbf{x})$ in (3.6), the weights in $\bar{g}^{(k)}(\mathbf{x})$ are defined with respect to the current and all the previous iteration-dependent weights \mathbf{R}_i^{-1} , $i = 1, \dots, k+1$. Moreover, the rank of $\bar{\mathbf{R}}_{k+1}^{(S)}$ is at most $2k+2$ so that, unless k is at least equal to the smallest integer greater or equal to $m-2/2$ (which is undesirable in large-scale settings), $\bar{\mathbf{R}}_{k+1}^{(S)}$ is singular and $\|\cdot\|_{\bar{\mathbf{R}}_{k+1}^{(S)}}$ is a semi-norm. However we will see that, for all the solvers in this subsection, $\bar{g}^{(k)}(\mathbf{x})$ is only evaluated for $\mathbf{x} \in \mathcal{R}(\mathbf{V}_k)$ (the approximation subspace for the solution, defined in (2.6) with $\hat{\mathbf{u}}_1 = \mathbf{r}_0$), implying that $\|\cdot\|_{\bar{\mathbf{R}}_{k+1}^{(S)}}$ is only applied to vectors belonging to $\mathcal{R}(\mathbf{U}_{k+1})$, and therefore it is a norm. Even if the approximation subspaces for the k th solution of all the flexible/inexact solvers considered in this and the previous sections coincide, the two classes of solvers are obviously different, as the initial problem formulations and the respective projected problems are different, leading to different solution coefficients (i.e., solutions of the projected problems) and, eventually, different solutions. This also agrees with Remark 2. In Section 4 we illustrate that, in many circumstances, the new solvers presented in this subsection deliver superior performance with respect to traditional solvers (considered in the previous subsection).

We first consider the approximation (A) in (3.17), and we then perform a projection (P), analogous to the one underlying the M- g method in Section 3.1: namely, we constrain the variable in (3.17) to belong to the space $\mathcal{R}(\mathbf{V}_k)$, i.e., we compute

$$\mathbf{x}_k = \mathbf{V}_k \mathbf{s}_k, \quad \text{with} \quad \mathcal{R}(\mathbf{V}_k) = \mathcal{K}_k(\mathbf{A}^\top \bar{\mathbf{R}}_{k+1} \mathbf{A}, \mathbf{A}^\top \bar{\mathbf{R}}_{k+1} \mathbf{r}_0)\tag{3.18}$$

by solving

$$\mathbf{s}_k = \underset{\mathbf{s} \in \mathbb{R}^k}{\operatorname{argmin}} \bar{g}_k^{(k)}(\mathbf{s}),\tag{3.19}$$

where, using the FGK relations (2.2),

$$\bar{g}_k^{(k)}(\mathbf{s}) = 1/2 \left(\mathbf{s}^\top \mathbf{V}_k^\top \mathbf{A}^\top \bar{\mathbf{R}}_{k+1}^{(S)} \mathbf{A} \mathbf{V}_k \mathbf{s} - 2\mathbf{s}^\top \mathbf{V}_k^\top \mathbf{A}^\top \bar{\mathbf{R}}_{k+1}^{(S)} \mathbf{r}_0 + \mathbf{r}_0^\top \bar{\mathbf{R}}_{k+1}^{(S)} \mathbf{r}_0 \right).\tag{3.20}$$

To solve the projected problem (3.19) we differentiate (D) $\bar{g}_k^{(k)}(\mathbf{s})$, getting the equation

$$\nabla \bar{g}_k^{(k)}(\mathbf{s}) = \mathbf{V}_k^\top \mathbf{A}^\top \bar{\mathbf{R}}_{k+1}^{(S)} \mathbf{A} \mathbf{V}_k \mathbf{s} - \mathbf{V}_k^\top \mathbf{A}^\top \bar{\mathbf{R}}_{k+1}^{(S)} \mathbf{r}_0 = \mathbf{0},$$

where $\bar{\mathbf{R}}_{k+1}^{(S)}$ is the symmetric part of $\bar{\mathbf{R}}_{k+1}$. Considering again the FGK relations in (2.2), and performing some straightforward algebraic manipulations (similar to the ones used to derive problem (3.12)), we eventually recover \mathbf{s}_k as the vector satisfying

$$(\mathbf{T}_{k,k+1} \mathbf{M}_k + (\mathbf{T}_{k,k+1} \mathbf{M}_k)^\top) \mathbf{s}_k = \beta t_{1,1} \mathbf{e}_1 + \mathbf{M}_k^\top \mathbf{Y}_{k+1}^\top \mathbf{r}_0,\tag{3.21}$$

where, like in the previous subsection, $\beta = \|\mathbf{r}_0\|_2$, $t_{1,1} = [\mathbf{T}_{k+1}]_{1,1}$, and $\mathbf{T}_{k,k+1}$ is \mathbf{T}_{k+1} without its last row. The APD method so derived is a LSQR-like solver. In the framework of [13], this method may be regarded as a ‘minimal inexact objective function’ (M- $\bar{g}^{(k)}$) method that determines $\mathbf{A}^\top \bar{\mathbf{R}}_{k+1}^{(S)} \mathbf{A} \mathbf{x}_k$ by performing an oblique projection of $\mathbf{A}^\top \bar{\mathbf{R}}_{k+1}^{(S)} \mathbf{r}_0$ onto $\mathbf{A}^\top \bar{\mathbf{R}}_{k+1}^{(S)} \mathbf{A} \mathcal{K}_k(\mathbf{A}^\top \bar{\mathbf{R}}_{k+1} \mathbf{A}, \mathbf{A}^\top \bar{\mathbf{R}}_{k+1} \mathbf{r}_0)$, orthonormal to $\mathcal{K}_k(\mathbf{A}^\top \bar{\mathbf{R}}_{k+1} \mathbf{A}, \mathbf{A}^\top \bar{\mathbf{R}}_{k+1} \mathbf{r}_0)$. Equivalently, it is based on a projection procedure that determines $(\bar{\mathbf{R}}_{k+1}^{(S)})^{1/2} \mathbf{A} \mathbf{x}_k$ by performing an orthogonal projection of $(\bar{\mathbf{R}}_{k+1}^{(S)})^{1/2} \mathbf{r}_0$ onto $(\bar{\mathbf{R}}_{k+1}^{(S)})^{1/2} \mathbf{A} \mathcal{K}_k(\mathbf{A}^\top \bar{\mathbf{R}}_{k+1} \mathbf{A}, \mathbf{A}^\top \bar{\mathbf{R}}_{k+1} \mathbf{r}_0)$.

Similarly, one can define an equivalent ADP method as follows. The starting point is the iteration-dependent approximate (A) functional $\bar{g}^{(k)}(\mathbf{x})$ in (3.17), which is then differentiated (D) to obtain

$$\nabla \bar{g}^{(k)}(\mathbf{x}) = \mathbf{A}^\top \bar{\mathbf{R}}_{k+1}^{(S)} \mathbf{A} \mathbf{x} - \mathbf{A}^\top \bar{\mathbf{R}}_{k+1}^{(S)} \mathbf{r}_0$$

and eventually projected (P) to get $\mathbf{x}_k = \mathbf{V}_k \mathbf{s}_k$, with $\mathbf{s}_k \in \mathbb{R}^k$, such that

$$\nabla \bar{g}^{(k)}(\mathbf{x}_k) \perp \mathcal{R}(\mathbf{V}_k) = \mathcal{K}_k(\mathbf{A}^\top \bar{\mathbf{R}}_{k+1} \mathbf{A}, \mathbf{A}^\top \bar{\mathbf{R}}_{k+1} \mathbf{r}_0) \iff \mathbf{s}_k \text{ solves (3.21).}$$

The ADP method so derived is a CGLS-like solver. In the framework of [13], this method may be regarded as an ‘orthogonal inexact normal equation residual’ (O- $\nabla \bar{g}^{(k)}$) that determines $\mathbf{A}^\top \bar{\mathbf{R}}_{k+1}^{(S)} \mathbf{A} \mathbf{x}_k$ by performing an oblique projection of $\mathbf{A}^\top \bar{\mathbf{R}}_{k+1}^{(S)} \mathbf{r}_0$ onto $\mathbf{A}^\top \bar{\mathbf{R}}_{k+1}^{(S)} \mathbf{A} \mathcal{R}(\mathbf{V}_k)$, orthogonal to $\mathcal{R}(\mathbf{V}_k)$, exactly as the LSQR-like APD solver. Remarkably, in the flexible/inexact setting, and after approximating the exact $g(\mathbf{x})$ in (3.2) (or $g^{(k)}(\mathbf{x})$ in (3.6)) by $\bar{g}^{(k)}(\mathbf{x})$ in (3.17), APD and ADP are the only LSQR-like and CGLS-like solvers that are mathematically equivalent when applied to minimize $\bar{g}^{(k)}(\mathbf{x})$ and solve $\nabla \bar{g}^{(k)}(\mathbf{x}) = \mathbf{0}$.

Finally, and again equivalently, one can start by projecting (P) the original $g(\mathbf{x})$ to get an approximation \mathbf{x}_k still belonging to the subspace $\mathcal{R}(\mathbf{V}_k)$ as in (3.18), i.e., one computes

$$\mathbf{x}_k = \mathbf{V}_k \mathbf{s}_k, \quad \text{with} \quad \mathbf{s}_k = \underset{\mathbf{s} \in \mathbb{R}^k}{\text{argmin}} g_k(\mathbf{s}) \quad \text{and} \quad g_k(\mathbf{s}) \text{ defined as in (3.4),}$$

and then takes $\bar{g}_k^{(k)}(\mathbf{s})$ in (3.20) to be an approximation (A) of $g_k(\mathbf{s})$, eventually obtaining the same formulation as (3.19). By applying differentiation (D) to impose the optimality conditions, one obtains again that \mathbf{s}_k solves problem (3.21), resulting in a LSQR-like PAD method equivalent to the first APD method of this class. Moreover, both PAD and APD share the same description in terms of the formalism in [13], including their characterization as M- $\bar{g}^{(k)}$ methods.

Inexactness estimates Similarly to what we did in Section 3.2.1, we provide some inexactness estimates, now concerning the gap between the exact functional $g^{(k)}(\mathbf{x})$ at iteration k , defined in (3.6), and the its inexact version $\bar{g}^{(k)}(\mathbf{x})$, defined in (3.17), both evaluated at $\mathbf{x}_k \in \mathcal{R}(\mathbf{V}_k)$.

Assume that, at iteration k of our new method, we have an approximate solution $\bar{\mathbf{x}}_k = \mathbf{x}_0 + \mathbf{V}_k \mathbf{s}_k$. First note that, similarly to what already stated for $\bar{g}^{(k)}$, when restricting the exact objective function at the k th iteration $g^{(k)}$ to $\mathcal{R}(\mathbf{V}_k)$, i.e., when computing

$$g_k^{(k)}(\mathbf{s}) = 1/2 \|\mathbf{A} \mathbf{V}_k \mathbf{s} - \mathbf{r}_0\|_{\mathbf{R}_{k+1}^{-1}}^2 = 1/2 \|\mathbf{U}_{k+1} \mathbf{M}_k \mathbf{s} - \mathbf{U}_{k+1} \beta \mathbf{e}_1\|_{\mathbf{R}_{k+1}^{-1}}^2 \quad (3.22)$$

on any possible coefficients $\mathbf{s} \in \mathbb{R}^k$, \mathbf{R}_{k+1}^{-1} is only applied to vectors in $\mathcal{R}(\mathbf{U}_{k+1}) = \text{span}\{\mathbf{u}_1, \dots, \mathbf{u}_{k+1}\}$. Now, let us define the following residual as

$$\mathbf{r}_k = \mathbf{r}_0 - \mathbf{A} \mathbf{V}_k \mathbf{s}_k = \mathbf{U}_{k+1} (\beta \mathbf{e}_1 - \mathbf{M}_k \mathbf{s}_k) \in \mathcal{R}(\mathbf{U}_{k+1}),$$

and note that, for any $\mathbf{u} = \sum_{i=1}^{k+1} (\mathbf{u}_i^\top \mathbf{u}) \mathbf{u}_i \in \mathcal{R}(\mathbf{U}_{k+1})$,

$$\mathbf{u}^\top (\bar{\mathbf{R}}_{k+1} - \mathbf{R}_{k+1}^{-1}) \mathbf{u} = \sum_{i=1}^{k+1} \mathbf{u}^\top (\mathbf{R}_i^{-1} - \mathbf{R}_{k+1}^{-1}) \mathbf{u}_i \mathbf{u}_i^\top \mathbf{u}.$$

Then, the magnitude of the difference between the exact and the inexact projected functionals on the solution at any iteration k can be bounded as follows:

$$\begin{aligned}
|\bar{g}_k^{(k)}(\mathbf{s}_k) - g_k^{(k)}(\mathbf{s}_k)| &= 1/2 \left| \sum_{i=1}^{k+1} \mathbf{r}_k^\top (\mathbf{R}_i^{-1} - \mathbf{R}_{k+1}^{-1}) \mathbf{u}_i \mathbf{u}_i^\top \mathbf{r}_k \right| \\
&\leq 1/2 \sum_{i=1}^{k+1} \left| \mathbf{r}_k^\top (\mathbf{R}_i^{-1} - \mathbf{R}_{k+1}^{-1}) \mathbf{u}_i \mathbf{u}_i^\top \mathbf{r}_k \right| \\
&\leq 1/2 \|\mathbf{r}_k\| \sum_{i=1}^{k+1} \|\mathbf{R}_i^{-1} - \mathbf{R}_{k+1}^{-1}\| |[\mathbf{M}_k \mathbf{s}_k - \beta \mathbf{e}_1]_i|
\end{aligned}$$

where we have used that $\mathbf{u}^\top \bar{\mathbf{R}}_{k+1}^{(S)} \mathbf{u} = \mathbf{u}^\top \bar{\mathbf{R}}_{k+1} \mathbf{u}$.

Remark 6. Similarly to the classical inexact methods, restarting when the original problem is too far from the problem that is solved in practice, is crucial to obtain good approximations of the solution.

Remark 7. As mentioned in the introduction in Section 1, a canonical example for our framework is the approximation of the following non-linear problem

$$f(\mathbf{x}) = 1/2 \|\mathbf{A}\mathbf{x} - \mathbf{b}\|_{\mathbf{R}^{-1}(\mathbf{x})}^2 \quad (3.23)$$

arising, for example, from considering the ℓ_p optimization in (1.4) where $\mathbf{R}^{-1}(\mathbf{x})$ is defined in (1.5). In this case, the inexactness estimates can be useful to understand the convergence of the flexible MM schemes, as detailed in the following proposition. Theoretically, one should restart once monotonicity of the objective function throughout the iterates cannot be guaranteed. Note that, even if the exact conditions are expensive to check in practice, this theory underpins the effect of restarting in the new flexible solvers.

Proposition 1. Monotonicity properties for flexible MM schemes. Consider the approximate solutions given the new flexible solvers to be $\{\mathbf{x}_k\}_{k=1,\dots}$, and assume that $\mathbf{R}_{k+1}^{-1} = \mathbf{R}^{-1}(\mathbf{x}_{k-1})$. Then, for $f(\mathbf{x})$ defined in (3.23), we can guarantee monotonicity in the objective value at each iteration k if sufficient progress is achieved in the corresponding inexact approximations $\bar{g}^{(k)}(\mathbf{x})$. Mathematically,

$$f(\mathbf{x}_k) < f(\mathbf{x}_{k-1}) \quad \Leftrightarrow \quad \bar{g}^{(k)}(\mathbf{x}_{k-1}) - \bar{g}^{(k)}(\mathbf{x}_k) > |\boldsymbol{\nu}^\top \sum_{i=1}^{k+1} (\mathbf{R}_i^{-1} - \mathbf{R}_{k+1}^{-1}) \mathbf{u}_i \mathbf{u}_i^\top \boldsymbol{\nu}|,$$

$$\text{for } \boldsymbol{\nu} = \mathbf{r}_k - \mathbf{r}_{k-1} = \mathbf{U}_k \begin{pmatrix} \mathbf{y}_k - \begin{bmatrix} \mathbf{y}_{k-1} \\ 0 \end{bmatrix} \end{pmatrix}.$$

Proof. Recall the definition of $g^{(k)}(\mathbf{x})$ for this specific case,

$$g^{(k)}(\mathbf{x}) = 1/2 \|\mathbf{A}\mathbf{x} - \mathbf{b}\|_{\mathbf{R}_{k+1}^{-1}}^2 = 1/2 \|\mathbf{A}\mathbf{x} - \mathbf{b}\|_{\mathbf{R}^{-1}(\mathbf{x}_{k-1})}^2,$$

and consider the following chain of in/equalities:

$$f(\mathbf{x}_k) < g^{(k)}(\mathbf{x}_k) \leq g^{(k)}(\mathbf{x}_{k-1}) = f(\mathbf{x}_{k-1}).$$

The first inequality is always true since $g^{(k)}(\mathbf{x})$ is a majorant of $f(\mathbf{x})$, and the last equality is always true since $g^{(k)}(\mathbf{x})$ is tangent to $f(\mathbf{x})$ at \mathbf{x}_{k-1} . Therefore, we only need to find explicit conditions for the central inequality to hold. We start by writing

$$g^{(k)}(\mathbf{x}) = \bar{g}^{(k)}(\mathbf{x}) - (\bar{g}^{(k)}(\mathbf{x}) - g^{(k)}(\mathbf{x})),$$

and we use similar derivations to the ones use for the inexactness estimates earlier in this section to express the second term.

In particular, we use that for any $\mathbf{u} \in \mathcal{R}(\mathbf{U}_{k+1})$

$$\mathbf{R}_{k+1}^{-1} \sum_{i=1}^{k+1} \mathbf{u}_i \mathbf{u}_i^\top \mathbf{u} = \mathbf{R}_{k+1}^{-1} \mathbf{u},$$

so that, since $\mathbf{r} \in \mathcal{R}(\mathbf{U}_{k+1})$,

$$\begin{aligned} g^{(k)}(\mathbf{x}_{k-1}) - g^{(k)}(\mathbf{x}_k) &= \\ &= \bar{g}^{(k)}(\mathbf{x}_{k-1}) - \bar{g}^{(k)}(\mathbf{x}_k) - (\mathbf{r}_{k-1} - \mathbf{r}_k)^\top \sum_{i=1}^{k+1} (\mathbf{R}_i^{-1} - \mathbf{R}_{k+1}^{-1}) \mathbf{u}_i \mathbf{u}_i^\top (\mathbf{r}_{k-1} - \mathbf{r}_k) \end{aligned}$$

where we know that

$$\bar{g}^{(k)}(\mathbf{x}_{k-1}) - \bar{g}^{(k)}(\mathbf{x}_k) = K > 0$$

since $\mathbf{x}_{k-1} \in \mathcal{R}(\mathbf{V}_{k-1}) \subset \mathcal{R}(\mathbf{V}_k)$ and \mathbf{x}_k minimizes $\bar{g}^{(k)}(\mathbf{x})$ in $\mathcal{R}(\mathbf{V}_k)$. Therefore, we can guarantee that the inequality will hold if the magnitude of the error term is smaller than K i.e.;

$$|(\mathbf{r}_k - \mathbf{r}_{k-1})^\top \sum_{i=1}^{k-1} \mathbf{E}_i^{-1} \mathbf{u}_i \mathbf{u}_i^\top (\mathbf{r}_k - \mathbf{r}_{k-1})| < K.$$

Interestingly, this is always true in the first iteration after each restart, since there isn't an error in the first term. \square

Remark 8. *Note that this analysis is only possible for the new APD, PAD and ADP solvers; because the inexactness estimates for classical solvers are only with respect to the gradient of the objective function and not the objective function itself.*

4 Numerical experiments

This section presents two numerical experiments designed to evaluate the performance of both the classical and the new inexact methods developed in this paper. In both problems, the measurements are corrupted with (sparse) salt-and-pepper noise, so it is reasonable to use an ℓ_1 data-fitting term. These examples highlight both the efficiency of the new methods with respect to other algorithms, as well as the improvement in reconstruction quality achieved by using the appropriate minimization problem.

Note that, as explained in Remark 5 and Remark 6, inexact methods greatly improve their performance if equipped with a restarting strategy. For these examples, we restart at the first iteration k such that

$$\exists i, j \in \{1, \dots, k+1\} : [\varepsilon_i]_j - [\varepsilon_i]_{j-1} > 0 \quad \text{for} \quad \varepsilon_i = \max(|\text{diag}(\mathbf{R}_i^{-1} - \mathbf{R}_{k+2}^{-1})|). \quad (4.1)$$

where $\mathbf{R}_{k+2}^{-1} = \mathbf{R}_{k+2}^{-1} = \mathbf{R}^{-1}(\mathbf{x}_k)$, or

$$\frac{\|\mathbf{A}\mathbf{x}_k - \mathbf{b}\|_2 - \|\mathbf{A}\mathbf{x}_0 - \mathbf{b}\|_2}{\|\mathbf{A}\mathbf{x}_k - \mathbf{b}\|_2} > \text{tol}. \quad (4.2)$$

Throughout the section, in order to evaluate the performance of the new algorithms and compare them to other standard methods, we report the relative error norm at iteration k , defined as $\|\mathbf{x}_k - \mathbf{x}_{\text{true}}\|_2 / \|\mathbf{x}_{\text{true}}\|_2$.

All computations were carried out in MATLAB R2023a, using both solvers and test problems available within the IR Tools [14] and AIR Tools II [24] MATLAB toolboxes.

Example 1: satellite image deblurring problem This example corresponds to an image deblurring problem where both the exact image and the available data (affected by Gaussian blur and salt-and-pepper noise) have 256×256 pixels and are displayed in Figure 1, together with the Gaussian point spread function defining the blur and noise array (also regarded as residual associated to the exact solution $\mathbf{b} - \mathbf{A}\mathbf{x}_{\text{true}} =: \mathbf{r}_{\text{true}}$). In particular, the salt-and-pepper noise is such that about 10% of the pixels are randomly set to values 0 or 1. Since the noise in the measurements is sparse, it is beneficial to use an ℓ_1 fit-to-data term to obtain a solution of better quality.

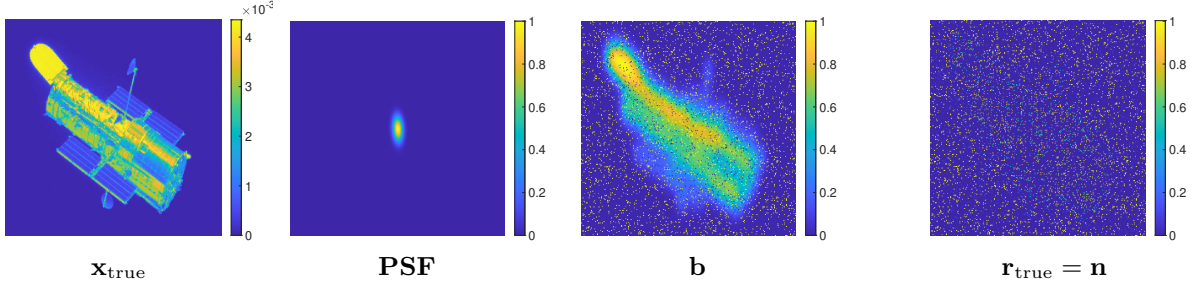


Figure 1: Data for the first example, representing an image deblurring test problem. From left to right: true solution, point spread function for the blur, noisy blurred image, and noise. The latter is displayed using the square root of the absolute values of each element.

For this example, we compare the performance of all the inexact/flexible methods considered in this paper. Namely, the new APD method, and the other methods operating in the same framework: DAP (corresponding to classical inexact CGLS and inexact LSQR) and DAP-LSMR (corresponding to classical inexact LSMR).

Since all of these are inexact methods, we compare their performance with and without restarts, where the restarts are triggered using the condition (4.1) and using $tol = 0.1$. Moreover, we also compare the new method to the classic LSQR, which considers a standard LS problem (with ℓ_2 -norm), and a standard IRLS solver for an ℓ_1 fit-to-data term, using LSQR as an inner solver. Last, we also compare preconditioned LSQR using the weights evaluated in the exact solution – this is of course not a valid method, since it requires the knowledge of the exact solution, and it is only displayed as a lower bound for the error.

The relative error histories for the different solvers are displayed in Figure 2. One can observe that the new method, jointly with the classic DAP and DAP-LSMR, produces the best reconstructions, as well as the fastest convergence behaviour, jointly with DAP.

The quality of the reconstructions is illustrated in the first row of Figure 3, which shows the best reconstruction obtained by each method (excluding LSQR with weights evaluated in the exact solution used as a preconditioner). The second row of Figure 3 presents the corresponding error images, obtained by reshaping the reconstruction errors. Figure 4 further displays the residuals and residual errors, also reshaped as images. For visualization purposes, the errors, residuals, and residual errors are shown as the absolute values of the square roots of the pixel values. Last, the basis vectors corresponding to the compared methods can be found in Figures 9 and 10 in Appendix A.

Example 2: Tomography problem This example corresponds to a (parallel beam) tomography test problem, where the solution has 256×256 pixels and the measurements, also known as sinogram, have 362×180 pixels (corresponding to the number of rays and the number of angles). The measurements have been normalized so that the maximum value is 1, and then ‘salt and pepper’ noise has been added, where 10% of the pixels are randomly taken to be 1 or 0. This test is created using IRtools [14], and the corresponding data can be found in Figure 5.

The relative error norm histories for different methods can be observed in Figure 6. For this example, the new inexact method (ADP), performs much better than any of the compared methods.

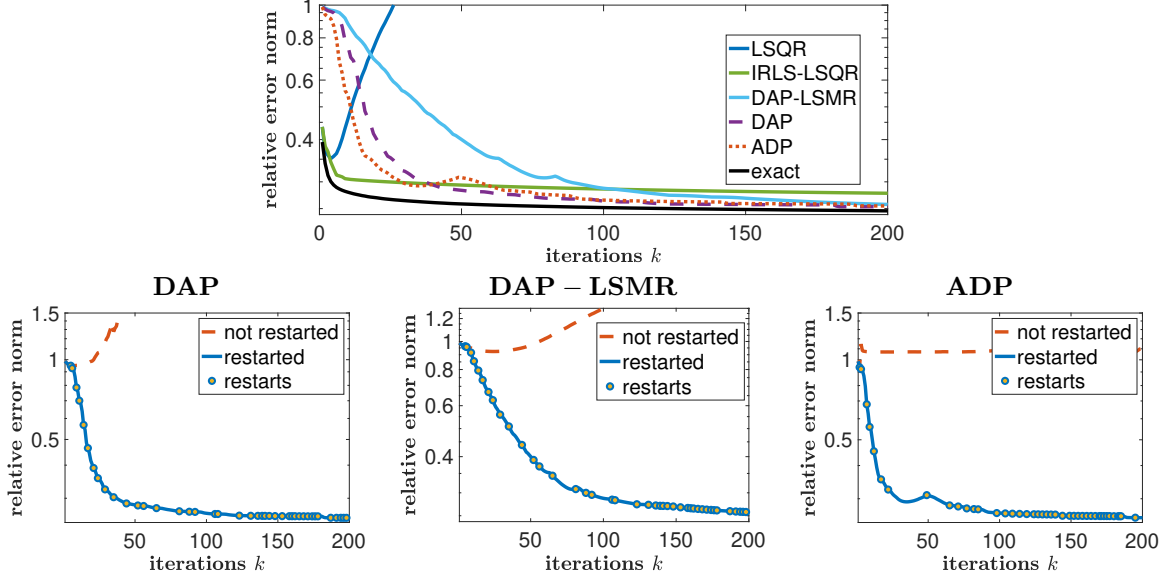


Figure 2: Example 1. Relative error norm values versus iteration number for a variety of solvers, with specific emphasis on the effect of restarting (second row). Note that the label ‘exact’ corresponds to preconditioned LSQR using the weights evaluated in the exact solution.

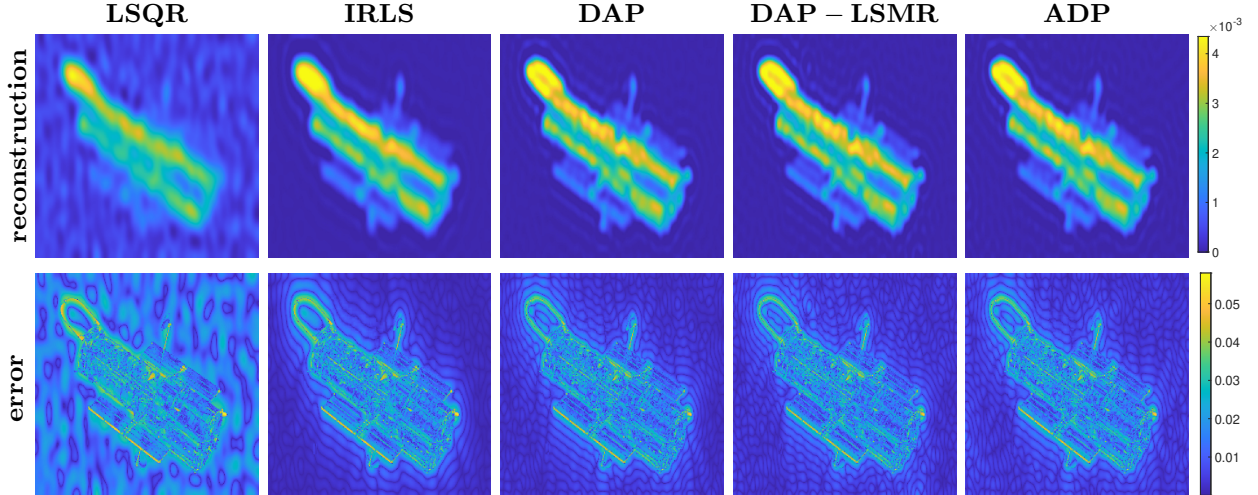


Figure 3: Example 1. Top row: Best reconstructions for a variety of algorithms. Bottom row: Square root of the absolute values of the error vectors (all operations performed entry-wise).

However, from the bottom row of Figure 6, it becomes evident again that restarting the iterations is necessary for the inexact methods to perform well. This matches the theory presented in Proposition 1. Note that, here, DAP and DAP-LSMR are restarted using (4.1) and ADP is restarted using (4.2), and in both cases $tol = 0.1$. The selection between these two criteria has been done to obtain the best results possible for each of the methods.

The reconstructions using different methods can be found in Figure 7, along with the errors (reshaped as an image). The latter have been displayed as the square root of the absolute value of each pixel, to help visualization, due to the big difference between their values. Here, one can observe again that the new ADP method produces the best quality reconstructions overall. It is also interesting to note that the error for the ADP method is less localized than that corresponding to the other methods. Moreover, the residuals, as well as the residual errors (defined as $\mathbf{Ax} - \mathbf{b} - \mathbf{r}_{\text{true}}$) can be observed in Figure 8, reshaped as images. Similarly to what we can observe in Figure 7, the

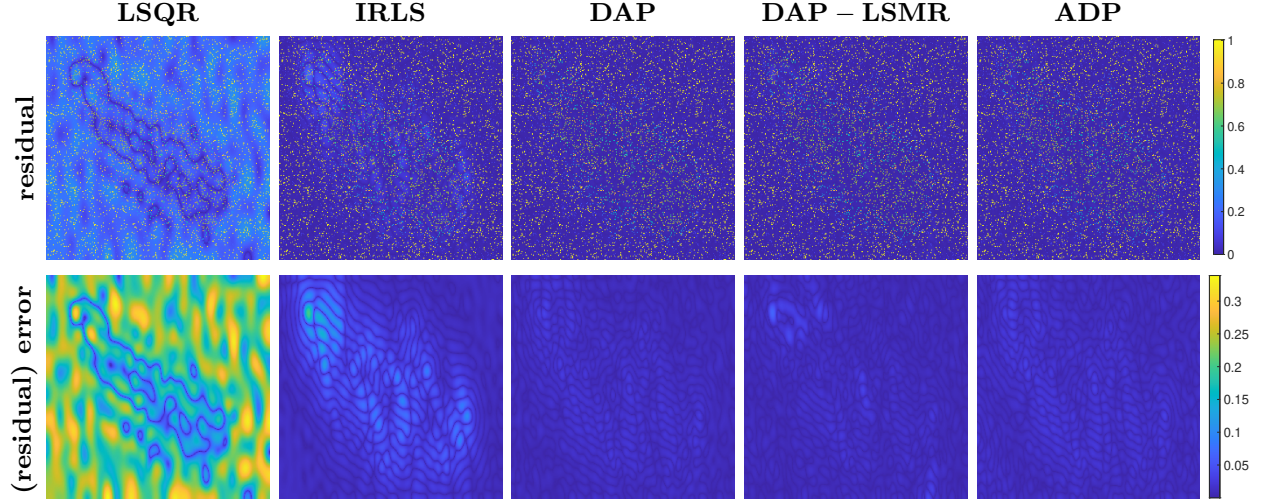


Figure 4: Example 1. Top row: Residuals associated to the best reconstructions for a variety of algorithms. Bottom row: residual error vectors, i.e. $\mathbf{Ax} - \mathbf{b} - \mathbf{r}_{\text{true}}$. All of the images are displaying the square root of the absolute values of each of the entries.

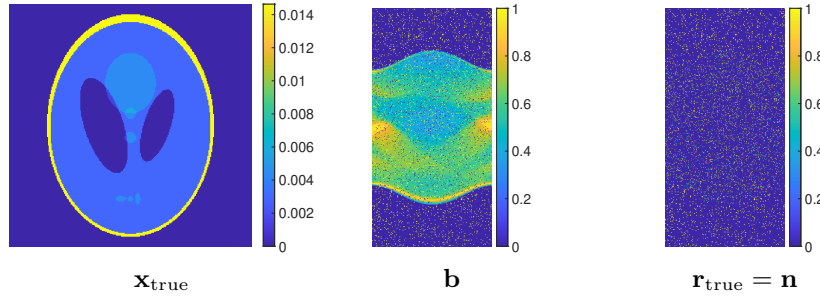


Figure 5: Data for the second example, representing a computed tomography test problem. Left: true solution. Center: noisy measurements (a.k.a. sinogram). Right: square root of the absolute value of the noise, or the residual of the true solution (all operations performed entry-wise).

new ADP method is the one that most accurately reconstructs the true residual, or, equivalently, the noise in the measurements. Finally, the basis vectors corresponding to the compared methods can be found in Figures 11 and 12 in Appendix A.

5 Conclusions and outlook

This paper introduces a generalized framework for inexact Krylov subspace methods, contributing an original insight on existing methods, as well as providing theoretical motivation for the development of new solvers. In particular, we present a novel inexact Golub-Kahan factorization associated to this new class of inexact Krylov methods, as well as an alternative residual constraint for the solution which can yield improved results with respect to their standard inexact counterparts. In this work, we also recall known bounds on the inexactness on traditional solvers, and we show that a more accurate measure can be given for the presented methods. This is based on the idea that the new methods are based on explicit optimality conditions of an approximated functional.

The main application focus of this paper is a new type of flexible Krylov subspace methods, which is particularly suited for problems involving a fit-to-data functional given in a ℓ_p norm. The starting point for this problem is to consider a sequence of weighted least-squares problems that can be partially solved by considering iteration-dependent left preconditioning which depends on

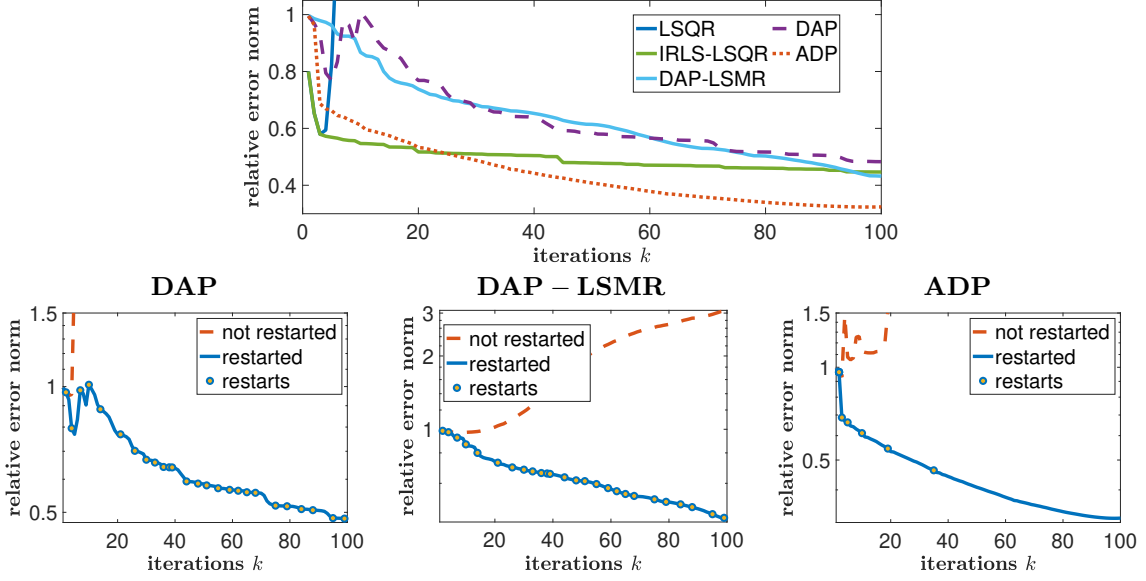


Figure 6: Example 2. Relative error norm values versus iteration number for a variety of solvers, with specific emphasis on the effect of restarting (bottom row).

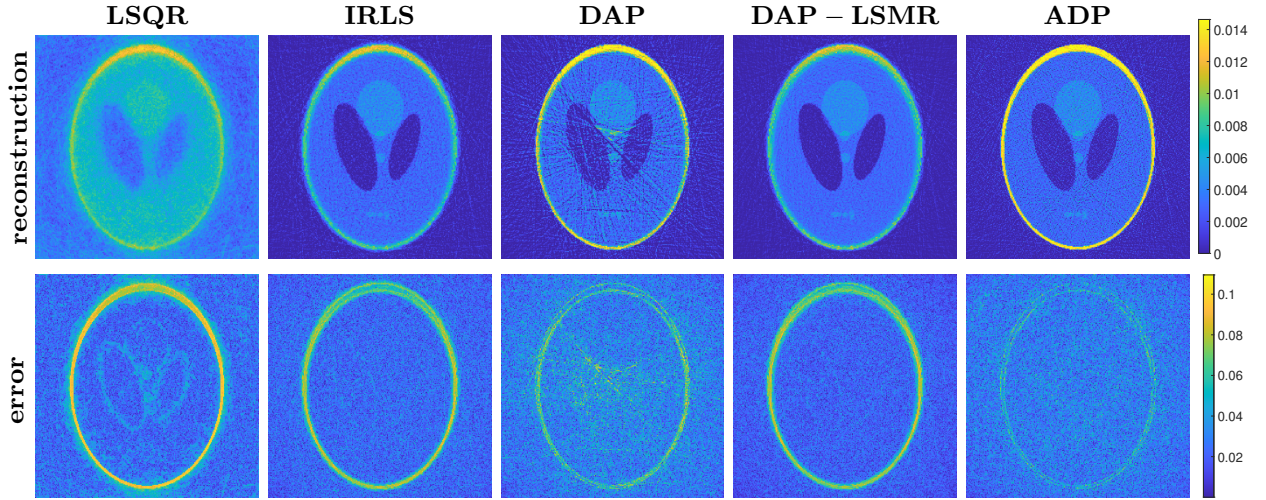


Figure 7: Example 2. Top row: Best reconstructions for a variety of algorithms. Bottom row: Square root of the absolute values of the error vectors (all operations performed entry-wise).

the last available approximation to the solution. The new inexact methods prove to be efficient alternatives to other methods. Moreover, for the solutions obtained with the new methods, we give conditions on the monotonicity of the original function with respect to the iterations, which motivate the use of restarts. However, these conditions are expensive to compute, so it is a topic of future work to investigate computationally cheap restarting conditions based on this theory.

The numerical experiments illustrate both the effectiveness and efficiency of the proposed methods compared to other standard solvers for an ℓ_p fit-to-data term.

Last, note that both the theory as well as the new solvers can be generalized to include explicit regularization in the projection functionals. This can be either Tikhonov regularization, to obtain a so called hybrid method, or more advanced regularization functionals such as those appearing in $\ell_p - \ell_q$ regularization. However, we leave this as a topic of further research.

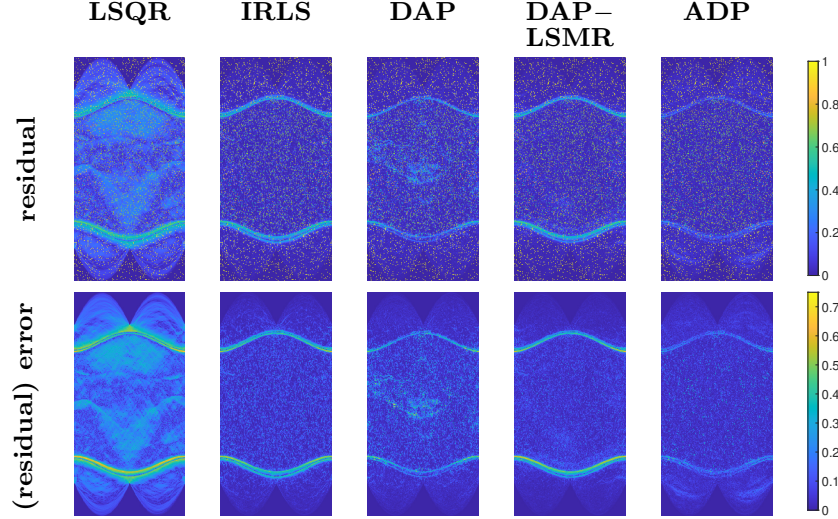


Figure 8: Example 2. Top row: Residuals associated to the best reconstructions for a variety of algorithms. Bottom row: residual error vectors, i.e. $\mathbf{Ax} - \mathbf{b} - \mathbf{r}_{\text{true}}$. All of the images are displaying the square root of the absolute values of each of the entries.

References

- [1] J. M. Bardsley and J. G. Nagy. Covariance-preconditioned iterative methods for nonnegatively constrained astronomical imaging. *SIAM J. Matrix Anal. Appl.*, 27(4):1184–1197, 2006.
- [2] M. Benning and M. Burger. Error estimates for general fidelities. *Electron. Trans. Numer. Anal.*, 38(44-68):77, 2011.
- [3] Å. Björck. *Numerical Methods for Least Squares Problems*. SIAM, Philadelphia, PA, 1996.
- [4] Å. Björck. *Numerical Methods in Matrix Computations*. Springer Nature, Switzerland AG, 2015.
- [5] A. Buccini, O. De la Cruz Cabrera, M. Donatelli, A. Martinelli, and L. Reichel. Large-scale regression with non-convex loss and penalty. *Appl. Numer. Math.*, 157:590–601, 2020.
- [6] A. Buccini, M. Pasha, and L. Reichel. Modulus-based iterative methods for constrained ℓ_p - ℓ_q minimization. *Inverse Probl.*, 36(8):084001, 2020.
- [7] J. Chung and S. Gazzola. Flexible Krylov methods for ℓ_p regularization. *SIAM J. Sci. Comput.*, 41(5):S149–S171, 2019.
- [8] J. Chung and S. Gazzola. Computational methods for large-scale inverse problems: A survey on hybrid projection methods. *SIAM Rev.*, 66(2):205–284, 2024.
- [9] J. Cornelis and W. Vanroose. Projected Newton method for noise constrained ℓ_p regularization. *Inverse Probl.*, 36:125004, 2024.
- [10] J. Cullum and A. Greenbaum. Relations between Galerkin and norm-minimizing iterative methods for solving linear systems. *SIAM J. Matrix Anal. Appl.*, (17):223–247, 1996.
- [11] Y. Dong, M. Hintermuller, and M. Neri. An efficient primal-dual method for l1tv image restoration. *SIAM J. Imaging Sci.*, 2(4):1168–1189, 2009.
- [12] Y. Dong and T. Zeng. A convex variational model for restoring blurred images with multiplicative noise. *SIAM J. Imaging Sci.*, 6(3):1598–162, 2013.

- [13] M. Eiermann and O. G. Ernst. Geometric aspects of the theory of Krylov subspace methods. *Acta Numer.*, 10:251–312, 2001.
- [14] S. Gazzola, P. C. Hansen, and J. G. Nagy. IR Tools: a MATLAB package of iterative regularization methods and large-scale test problems. *Numer. Algorithms*, 81(3):773–811, 2019. Software available at <https://github.com/jnagy1/IRtools>.
- [15] S. Gazzola, J. G. Nagy, and M. Sabaté Landman. Iteratively reweighted FGMRES and FLSQR for sparse reconstruction. *SIAM J. Sci. Comput.*, 43(5), 2021.
- [16] S. Gazzola and M. Sabaté Landman. Regularization by inexact Krylov methods with applications to blind deblurring. *SIAM J. Matrix Anal. Appl.*, 42(4):1528–1552, 2021.
- [17] S. Gazzola and Y. Wiaux. Fast nonnegative least squares through flexible Krylov subspaces. *SIAM J. Sci. Comput.*, 39(2), 2017.
- [18] W. H. Greene. Econometric analysis. *Pretence Hall*, 2003.
- [19] M. Hanke and P. C. Hansen. Regularization methods for large-scale problems. *Surveys Math. Indust.*, 3(4):253–315, 1993.
- [20] P. C. Hansen. The discrete Picard condition for discrete ill-posed problems. *BIT*, 30:658–672, 1990.
- [21] P. C. Hansen. *Rank-Deficient and Discrete Ill-Posed Problems*. SIAM, Philadelphia, PA, 1998.
- [22] P. C. Hansen. *Discrete Inverse Problems: Insight and Algorithms*. SIAM, 2010.
- [23] P. C. Hansen. Insight into semi-convergence of iterative regularization methods. *Linear Algebra Appl.*, In Press, 2025.
- [24] P. C. Hansen and J. S. Jørgensen. AIR Tools II: Algebraic Iterative Reconstruction Methods, Improved Implementation. *Numer. Algorithms*, 2018. Software available at <https://github.com/jakobsj/AIRToolsII/>.
- [25] P. C. Hansen, J. G. Nagy, and D. P. O’Leary. *Deblurring Images: Matrices, Spectra and Filtering*. SIAM, Philadelphia, PA, 2006.
- [26] G. Huang, A. Lanza, S. Morigi, L. Reichel, and F. Sgallari. Majorization–minimization generalized Krylov subspace methods for $\ell_p - \ell_q$ optimization applied to image restoration. *BIT*, 57(2):351–378, 2017.
- [27] M. Sabate Landman and J. Chung. Flexible Krylov Methods for Group Sparsity Regularization. *Phys. Scr.*, 99(12), 2024.
- [28] P. Rodriguez and B. Wohlberg. An efficient algorithm for sparse representations with ℓ^p data fidelity term. In *Proceedings of 4th IEEE Andean Technical Conference (ANDESCON)*, 2008.
- [29] Y. Saad. *Iterative Methods for Sparse Linear Systems*. SIAM, Philadelphia, 2003.
- [30] F. Sciacchitano, Y. Dong, and T. Zeng. Variational approach for restoring blurred images with Cauchy noise. *SIAM J. Imaging Sci.*, 8(3):1894–1922, 2015.
- [31] V. Simoncini and D. Szyld. Theory of inexact Krylov subspace methods and applications to scientific computing. *SIAM J. Sci. Comput.*, 25(2), 2003.
- [32] B. Wohlberg and P. Rodriguez. An iteratively reweighted norm algorithm for minimization of total variation functionals. *IEEE Signal Processing Letters*, 14(12):948–951, 2007.

- [33] G. Zyskind and F. B. Martin. On best linear estimation and general Gauss-Markov Theorem in linear models with arbitrary nonnegative covariance structure. *SIAM J. Appl. Math.*, 17(6):1190–1202, 1969.

A Basis vectors

In this Appendix, we show a selection of basis vectors constructed by the different methods compared in Section 4. Note that, even if some of these processes share the same flexible Golub-Kahan factorization described in Algorithm 1, the iteration-dependent preconditioning depends on the approximations of the solution at each iteration and, therefore, the solution subspace depends indirectly on the optimality conditions that differentiate the methods. Note that, particularly for the computed tomography example, one can clearly observe the difference between the two sets of basis vectors. Moreover, it is interesting to note that the basis vectors look very different for the different methods. Understanding how the properties of the basis vectors constructed by the ADP method relate to the improved quality of their results is left as future work.

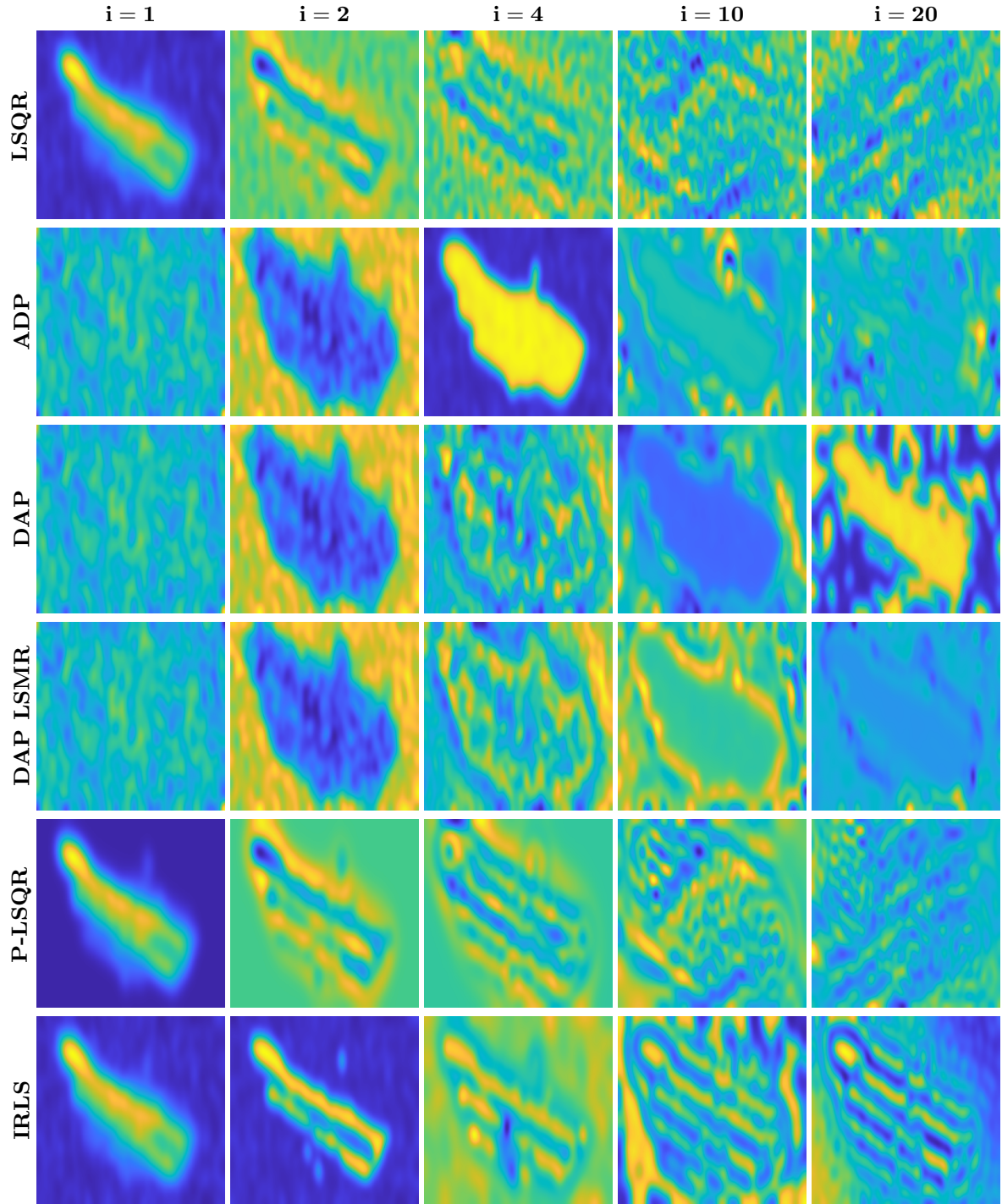


Figure 9: Example 1. Basis vectors for the solution space, i.e. \mathbf{V} columns, for the different compared methods.

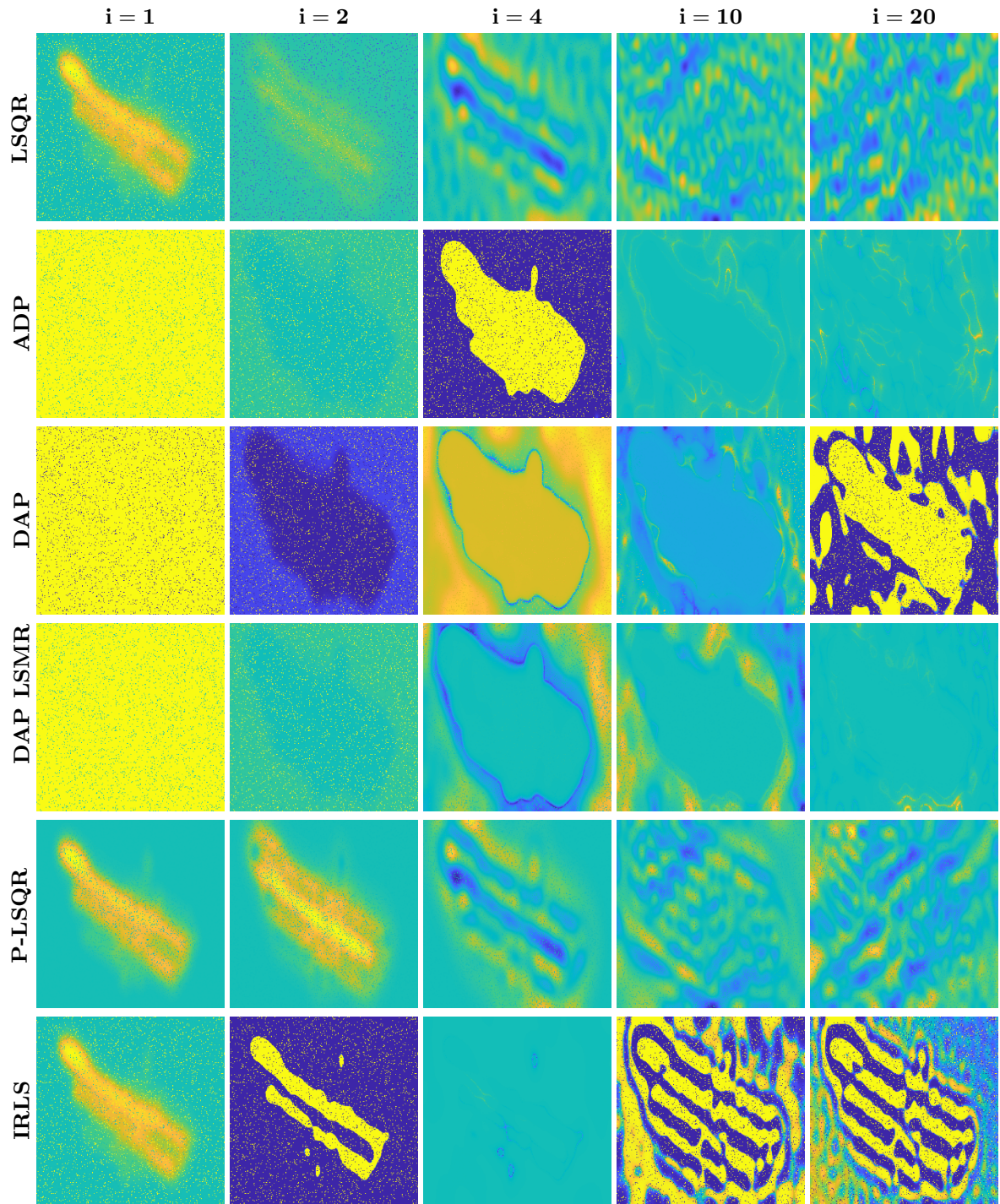


Figure 10: Example 1. Basis vectors for the residual space, i.e. \mathbf{Z} columns, for the different compared methods.

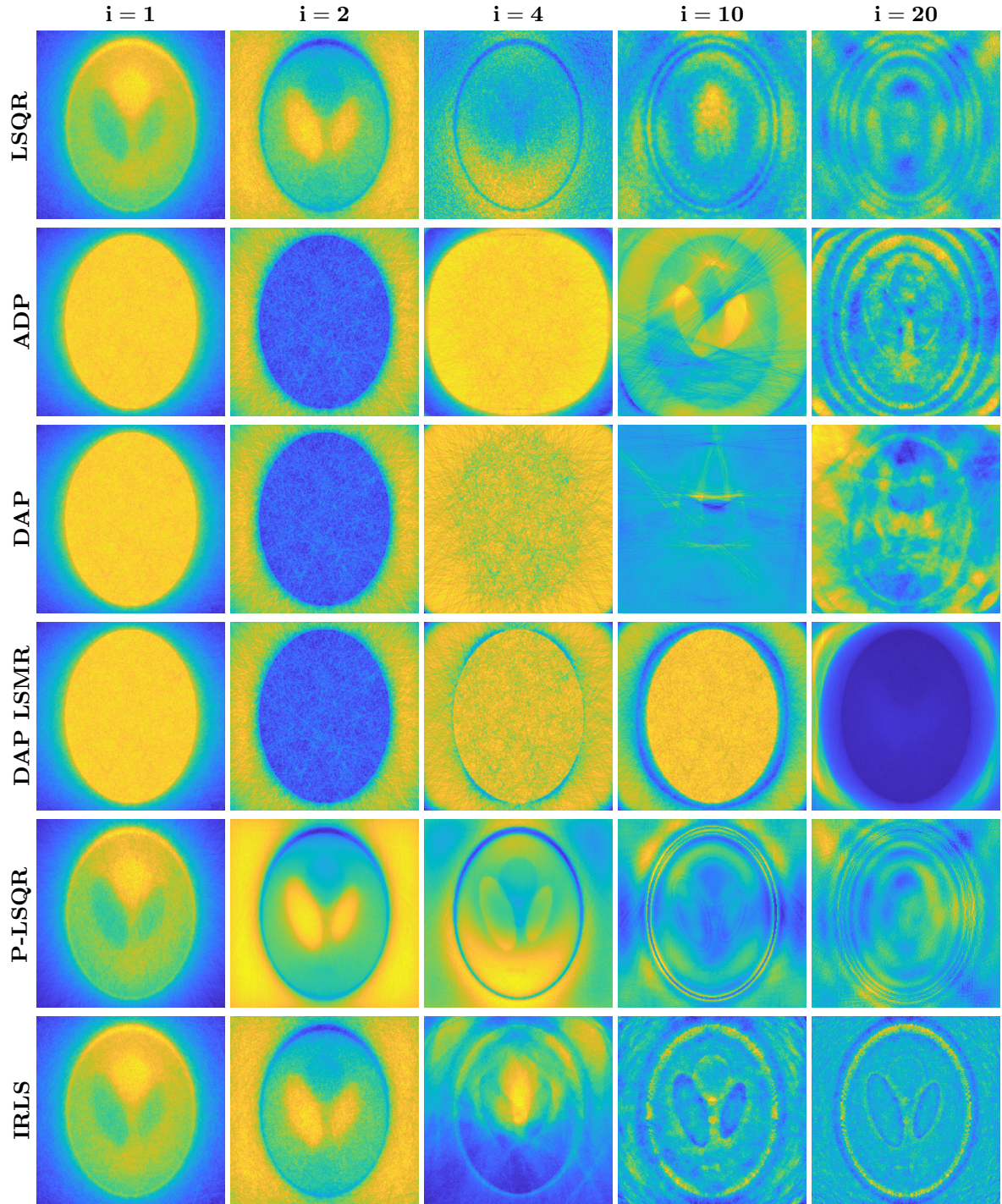


Figure 11: Example 2. Basis vectors for the solution space, i.e. \mathbf{V} columns, for the different compared methods.

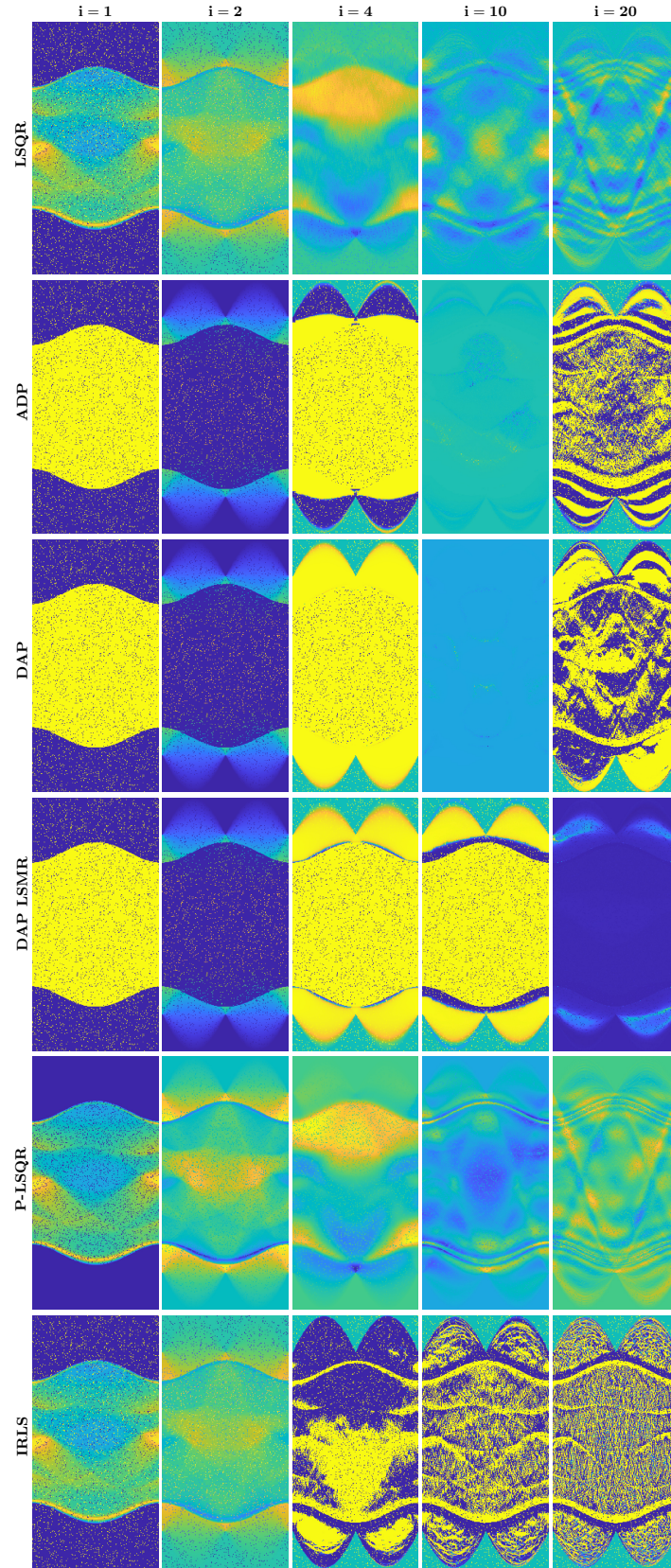


Figure 12: Example 2. Basis vectors for the residual space, i.e. \mathbf{Z} columns, for the different compared methods.