# Accelerating linear system solvers for time-domain component separation of cosmic microwave background data

Jan Papež, Laura Grigori, Radek Stompor

A&A 638, A73 (2020)
https://doi.org/10.1051/0004-6361/202037687


Astronomy & Astrophysics

# Accelerating linear system solvers for time-domain component separation of cosmic microwave background data

J. Papež[1,2], L. Grigori[1], and R. Stompor[3,4]

[1] INRIA Paris, Sorbonne Université, Université Paris-Diderot SPC, CNRS, Laboratoire Jacques-Louis Lions, ALPINES Team, France
e-mail: jan@papez.org
[2] Currently at Institute of Mathematics, Czech Academy of Sciences, Prague, Czech Republic
[3] Université de Paris, CNRS, AstroParticule et Cosmologie, 75013 Paris, France
[4] CNRS-UCB International Research Laboratory, "Centre Pierre Binétruy", UMI2007, CPB-IN2P3, France

Received 7 February 2020 / Accepted 12 April 2020

**ABSTRACT**

Component separation is one of the key stages of any modern cosmic microwave background data analysis pipeline. It is an inherently nonlinear procedure and typically involves a series of sequential solutions of linear systems with similar but not identical system matrices, derived for different data models of the same data set. Sequences of this type arise, for instance, in the maximization of the data likelihood with respect to foreground parameters or sampling of their posterior distribution. However, they are also common in many other contexts. In this work we consider solving the component separation problem directly in the measurement (time-) domain. This can have a number of important benefits over the more standard pixel-based methods, in particular if non-negligible time-domain noise correlations are present, as is commonly the case. The approach based on the time-domain, however, implies significant computational effort because the full volume of the time-domain data set needs to be manipulated. To address this challenge, we propose and study efficient solvers adapted to solving time-domain-based component separation systems and their sequences, and which are capable of capitalizing on information derived from the previous solutions. This is achieved either by adapting the initial guess of the subsequent system or through a so-called subspace recycling, which allows constructing progressively more efficient two-level preconditioners. We report an overall speed-up over solving the systems independently of a factor of nearly 7, or 5, in our numerical experiments, which are inspired by the likelihood maximization and likelihood sampling procedures, respectively.

**Key words.** cosmic background radiation – methods: numerical

## 1. Context and motivation

Measurements registered by cosmic microwave background (CMB) experiments contain, in addition to the sought-after signal of cosmological origin, contributions from astrophysical sources. These are generically called foregrounds and can be of either galactic or extragalactic origins and be either diffuse or point-source-like morphologically. A separation of the foreground signals from each other and, specifically, from the CMB signal is therefore an essential step of any modern CMB data analysis. This step is referred to as component separation. It is performed by capitalizing on either different electromagnetic frequency dependence and/or statistical properties of different signals (e.g., Planck Collaboration X 2016, and references therein). In polarization the foreground signals tend to dominate the CMB signal over a broad range of angular scales and observational frequencies. The next generation of CMB observatories will therefore be only capable of delivering its science in full if high-precision statistically sound and reliable component separation techniques and their numerically efficient implementations are available.

Component separation is a nonlinear operation. Based on data measured at multiple different frequency bands, it aims to simultaneously recover the frequency dependence of the foregrounds as well as their spatial morphology. It is commonly performed in a pixel domain and uses maps of the sky estimated for each frequency band and their statistical uncertainties as inputs. These objects are assumed to have been obtained in a preceding step of the data analysis that is called map-making.

For concreteness, in this work we focus on the so-called parametric component separation approach (e.g., Brandt et al. 1994; Eriksen et al. 2008; Stompor et al. 2009), where the frequency-scaling relations for each considered sky component are assumed to be given up to a limited number of unknown parameters, called foreground spectral parameters. However, the numerical techniques discussed here are more general and should be found useful also in other component separation methods.

The component separation is typically performed in two steps. In the first step, the spectral parameters, or more generally, the mixing matrix elements, are estimated from the data, and in the second step, they are used to recover maps of sky components from the frequency maps. This approach is conceptually simple and potentially very efficient computationally. The input frequency maps preserve essentially all the information present in a typically much larger initial raw data set, and their smaller sizes make them easier to store and operate on. For the next generation of CMB experiments, we expect to have as many as $n_{\rm t} \sim O(10^{13} - 10^{15})$ raw measurements, but only $n_{\rm pix} \sim O(10^5 - 10^8)$ sky pixels.

The pixel-domain component separation approaches can ensure satisfactory performance but require a sufficiently precise statistical description of the frequency maps. This has to

be derived from the raw measurements, which we refer to hereafter as time-domain data. In practice, this is often difficult because storage and computational cycles are limited. A general full covariance matrix of a single frequency map contains $n_{\rm pix}^2 \sim O(10^{10} - 10^{16})$ elements, which would need to be stored in memory. Computing these elements costs at least $O(\lambda)\,O(n_{\rm t})$ floating point operations (flops). (Here $\lambda$ is a time-domain noise correlation length and can reach many thousands of samples.) The full computations therefore quickly become prohibitively expensive. This is the case even if the explicit inversion of the covariance matrix is replaced by some iterative procedure, which typically requires $O(n_{\rm iter}\, n_{\rm pix}^2)$ flops, where the number of iterations, $n_{\rm iter}$ is usually on the order of $10^2$. Consequently, the best way forward in practice may be to invoke some approximations. This is often problematic as well, however, because a successful approximation needs to ensure sufficient accuracy to avoid introducing systematic biases in the estimated foreground parameters and later also in the component maps.

A general solution to the problem would be to avoid relying on the frequency maps at all and to perform all the calculation directly on the time-domain data. This would typically require memory on the order of $O(n_{\rm t})$ and $O(p\, n_{\rm iter}\, n_{\rm t}\, \ln\, \lambda)$ flops. The prefactor $p$ is on the order of unity for a typical map-making run, but in our case, it can vary widely between a few tens and many thousands. This highlights the challenge faced by the proposed approach. We note that while this is certainly very demanding, it is not necessarily prohibitive. Some of the best optimized existing map-making codes can already perform many hundreds of runs, for instance, as required in massive Monte Carlo simulations. The proposed approach may not only be more robust, but may be the only way forward if significant time-domain noise correlations are present, $\lambda \gg 1$. This is commonly the case in the CMB experiments, in particular, those operating from the ground.

In this work, we explore some of the avenues that might render this approach tractable. We first identify the main computation-heavy step that unavoidably appears in any implementation of this technique. We then investigate how it might be accelerated by employing better and more advanced methods and their implementations.

The plan of this paper is as follows. In Sect. 2 we present the component separation problem and the numerical challenges it poses. In Sect. 3 we describe the proposed solution and in Sect. 4 the results of the numerical tests. Section 5 provides a brief summary and outlines prospects. Material that is more technical in nature or that is added for completeness is as usual deferred to the appendices.

## 2. Problem description and setting

### 2.1. Preliminaries

Hereafter, we consider polarized signals and assume, for simplicity and definiteness, that in every sky pixel the signal is characterized by two Stokes parameters, $Q$ and $U$. Extensions to include total intensity, $I$, are straightforward. Consequently, hereafter, every considered sky map consists of two maps corresponding to the two Stokes parameters. They are concatenated in a single map vector,

$$\mathsf{v} = \left\lceil v_q, v_u \right\rfloor \equiv \begin{bmatrix} v_q \\ v_u \end{bmatrix}, \quad v_q, v_u \in \mathbb{R}^{n_{\rm pix}}. \tag{1}$$

Hereafter, partial brackets, $\lceil \ldots \rfloor$, denote a vertical object. Examples of the sky maps as discussed in the following are single-frequency maps storing information about the sky signal as observed at a given frequency, or single-component maps containing information about a sky signal of some specific physical origin. We refer to the ordering defined above as Stokes-wise because a complete sky map of one Stokes parameter is followed up by another. In addition, we also consider a pixel-wise ordering, which for single maps reads

$$\mathsf{v} \equiv \left\lceil v_q(1), v_u(1), \ldots, v_q(n_{\rm pix}), v_u(n_{\rm pix}) \right\rfloor, \tag{2}$$

where the $Q$ and $U$ parameters of a signal in one pixel are stored consecutively and are followed by those in another.

The goal of the component separation procedure is to estimate all assumed sky component signals given multiple frequency data. Therefore, we commonly deal with multiple maps of the same type, such as multiple single-frequency maps or multiple single-component maps. We concatenate them in a single multifrequency or multicomponent vector. For definiteness, in this work we fix the number of components to $n_{\rm comp} = 3$ and consider three different sky components: CMB, dust, and synchrotron. A multicomponent vector, $\mathtt{s}$, therefore contains information about the $Q$ and $U$ Stokes parameters of all three components. Such a vector can be ordered in multiple ways. Most commonly, we assume that it is ordered either in a component-wise way, when

$$\begin{aligned} \mathtt{s} &\equiv \left\lceil \mathsf{s}_{\rm cmb}, \mathsf{s}_{\rm dust}, \mathsf{s}_{\rm sync} \right\rfloor \\ &= \left\lceil s_{{\rm cmb},q}, s_{{\rm cmb},u}, s_{{\rm dust},q}, s_{{\rm dust},u}, s_{{\rm sync},q}, s_{{\rm sync},u} \right\rfloor \in \mathbb{R}^{6n_{\rm pix}}, \end{aligned} \tag{3}$$

or in a pixel-wise way, where for each pixel all Stokes parameters follow consecutively for all considered components, that is,

$$\begin{aligned} \mathtt{s} \equiv \lceil & s_{{\rm cmb},q}(1), s_{{\rm cmb},u}(1), \ldots, s_{{\rm sync},q}(1), s_{{\rm sync},u}(1), \ldots \\ & s_{{\rm cmb},q}(n_{\rm pix}), s_{{\rm cmb},u}(n_{\rm pix}), \ldots, s_{{\rm sync},q}(n_{\rm pix}), s_{{\rm sync},u}(n_{\rm pix}) \rfloor. \end{aligned} \tag{4}$$

Multifrequency vectors can be ordered in analogous manners.

The choice of the ordering in general depends on the specific context and is obviously of key importance for the numerical implementation of the map-making or component separation procedures. Nonetheless, mathematically, switching the ordering from one to another is described by a linear, orthonormal, full-rank operator, $U$. This operator is conceptually trivial to apply, and its application commutes with other matrix operations such as a matrix inversion because

$$(U\, M\, U^t)^{-1} = U\, M^{-1}\, U^t, \tag{5}$$

for any invertible matrix $M$. Consequently, a matrix can be inverted using one ordering, for instance, computing $M^{-1}$, and the result can later be reordered to obtain the inverse in the other ordering scheme, that is, $(U\, M\, U^t)^{-1}$. For this reason, we freely switch between the different orderings depending on the context in the following in order to highlight specific structures of the matrices, which may be more apparent for one choice than the other.

### 2.2. Data model

As mentioned earlier, we consider a component separation procedure performed directly on the time-domain data as measured by the instrument. Thus we do not invoke any prior explicit map-making procedure. We therefore need to relate the time-domain measurements directly to the component maps because these maps are the intended outcome of the component separation procedure. We assume that for each frequency the time-domain data

are made of sequences of consecutive observations registered by all detectors operating at this frequency and concatenated, we can write

$$d_\mathrm{f} = P_{\beta^\star,\mathrm{f}}\, \mathtt{s}_\star + n_\mathrm{f}, \quad d_\mathrm{f}, n_\mathrm{f} \in \mathbb{R}^{n_t}, \quad f = 1, \ldots, n_\mathrm{freq}. \tag{6}$$

Here $\mathtt{s}_\star$ is the unknown vector of the component amplitudes, and the star indicates that those are their actual values. $n_\mathrm{f}$ denotes an (unknown) noise vector. The number of the frequency channels, $n_\mathrm{freq}$, is assumed to be larger than that of the components, $n_\mathrm{comp}$, set to 3 in this work, to ensure that the problem is well defined. The matrix $P_{\beta^\star,\mathrm{f}}$ in Eq. (6) combines the information about the instrument operations and the sky properties. It can be expressed as

$$P_{\beta^\star,\mathrm{f}} = P_\mathrm{f} \cdot M_{\beta^\star,\mathrm{f}}, \tag{7}$$

where $M_{\beta^\star,\mathrm{f}} \in \mathbb{R}^{2n_\mathrm{pix}\times 6n_\mathrm{pix}}$ is a so-called mixing matrix, and it determines how different sky components mix at all observed frequencies to yield the observed signal. The mixing matrix explicitly depends on the foreground scaling parameters, which we denote as $\beta^\star$, and the frequency of the observation, $f$. $P_\mathrm{f} \in \mathbb{R}^{n_t\times 2n_\mathrm{pix}}$ is in turn a pointing matrix defining which pixel of the sky each detector operating at a given frequency observed at every time. While it does not explicitly depend on frequency or scaling parameters, it therefore is in principle different for different frequencies because it encodes pointing of detectors specific to this frequency. This is highlighted by the subscript $f$. We have

$$P_\mathrm{f} : \left[s^\star_{\mathrm{f},q}, s^\star_{\mathrm{f},u}\right] \mapsto d_\mathrm{f}, \qquad M_{\beta^\star,\mathrm{f}} : \mathtt{s}_\star \mapsto \mathtt{s}_\mathrm{f} \equiv \left[s^\star_{\mathrm{f},q}, s^\star_{\mathrm{f},u}\right], \tag{8}$$

where $\mathtt{s}^\star_\mathrm{f}$ is a single-frequency map expressing the combined sky signal at frequency $f$. The data vector, $d_\mathrm{f}$, is time-ordered because its elements are indexed by the time at which the measurement was taken.

### 2.3. Component separation

The goal of the component separation procedure is to solve an inverse problem, Eq. (6), and estimate the components, $\mathtt{s}_\star$, given the full data set, $d\,(:=\{d_\mathrm{f}\})$, made of data taken at all observational frequencies. This is typically solved by assuming that the noise, $n_\mathrm{f}$, is Gaussian, with a zero mean and a known variance, $N_\mathrm{f}$, and writing a data likelihood,

$$-2\ln\mathcal{L}(\beta, \mathtt{s}; d) = (\widetilde{d} - \widetilde{P}_\beta\, \mathtt{s})^\top\, N^{-1}\, (\widetilde{d} - \widetilde{P}_\beta\, \mathtt{s}) \;+\; \mathrm{CONST}. \tag{9}$$

Here we have dropped the star to distinguish an estimate from the true value, and we have introduced a tilde to denote multifrequency objects. We have

$$\widetilde{P}_\beta = \begin{bmatrix} P_{\beta,1} \\ \vdots \\ P_{\beta,n_\mathrm{freq}} \end{bmatrix} = \begin{bmatrix} P_1 \cdot M_{\beta,1} \\ \vdots \\ P_{n_\mathrm{freq}} \cdot M_{\beta,n_\mathrm{freq}} \end{bmatrix}, \tag{10}$$

which follows from Eq. (7), and

$$\widetilde{N} = \begin{bmatrix} N_1 & & 0 \\ & \ddots & \\ 0 & & N_{n_\mathrm{freq}} \end{bmatrix}, \quad \widetilde{d} = \begin{bmatrix} d_1 \\ \vdots \\ d_{n_\mathrm{freq}} \end{bmatrix}, \tag{11}$$

which assumes no noise correlations between different frequency channels. In addition, throughout this work we also assume that while the component mixing represented by $M_\beta$ may involve (potentially) all components, it is always done on a pixel-by-pixel basis, so that all the elements of $M_\beta$ corresponding to different pixels vanish. Similarly, and in agreement with assumptions made in map-making procedures, we assume that the noise matrices, $N_\mathrm{f}$, are block diagonal, with each block representing a banded Toeplitz matrix.

The standard two-step component separation procedure proceeds by first estimating for each frequency band, $f$, a single-frequency map, $m_\mathrm{f}$, and its covariance, $\hat{N}_\mathrm{f}$. These are given by

$$m_\mathrm{f} = (P_\mathrm{f}^\top\, N_\mathrm{f}^{-1}\, P_\mathrm{f})^{-1}\, P_\mathrm{f}^\top\, N_\mathrm{f}^{-1}\, d_\mathrm{f}, \tag{12}$$

$$\hat{N}_\mathrm{f} = (P_\mathrm{f}^\top\, N_\mathrm{f}^{-1}\, P_\mathrm{f})^{-1}. \tag{13}$$

The follow-up component separation step is then performed assuming that the single-frequency maps yielded by the first step can be represented as

$$m_\mathrm{f} = M_{\beta^\star,\mathrm{f}}\, \mathtt{s}_\star + \hat{n}_\mathrm{f}, \tag{14}$$

where $\hat{n}_\mathrm{f}$ stands for a pixel-domain noise and is a Gaussian variable with variance $\hat{N}_\mathrm{f}$. We can therefore write the corresponding likelihood as

$$-2\ln\mathcal{L}(\beta, \mathtt{s}; \{m_\mathrm{f}\}) = \sum_\mathrm{f} (m_\mathrm{f} - M_{\beta,\mathrm{f}}\, \mathtt{s})^\top\, \hat{N}_\mathrm{f}^{-1}\, (m_\mathrm{f} - M_{\beta,\mathrm{f}}\, \mathtt{s}) \;+\; \mathrm{CONST}. \tag{15}$$

This procedure is equivalent to directly solving the maximum likelihood problem defined by Eq. (9). However, it requires an explicit calculation of $\hat{N}_\mathrm{f}^{-1}$ that for the current and forthcoming experiment is typically prohibitive because of restrictions on both the available computer memory and computational cycles. An alternative might be solving the original problem directly without explicitly invoking any pixel-domain objects. This is the option we study in this work. We note here in passing that intermediate approaches are also possible: for instance, one that relies on the likelihood in Eq. (15), but does not assume that $\hat{N}_\mathrm{f}^{-1}$ is given explicitly. Instead, it computes a product of the covariance and a vector using an iterative procedure, which only requires applying the inverse covariance to a vector. This is performed using its implicit representation, Eq. (13), as is done in the map-making solvers. On the algorithmic level, such approaches are equivalent to solving the problem in the time domain, and the methods considered hereafter would be applicable to that approach as well.

To estimate $\beta$ and $\mathtt{s}$ directly from Eq. (9), we may either maximize this likelihood or sample from a posterior derived from it assuming some priors on the spectral parameters[1]. Alternatively, a so-called spectral likelihood may be used (Stompor et al. 2009), where $\mathtt{s}$ is already either marginalized or maximized over, that is,

$$2\ln\mathcal{L}_\mathrm{spec}(\beta; \widetilde{d}) = \widetilde{d}^\top\, \widetilde{N}^{-1}\, \widetilde{P}_\beta\, (\widetilde{P}_\beta^\top\, \widetilde{N}^{-1}\, \widetilde{P}_\beta)^{-1}\, \widetilde{P}_\beta^\top\, \widetilde{N}^{-1}\, \widetilde{d} \;+\; \mathrm{CONST}, \tag{16}$$

which again can be either minimized or sampled from.

In both these cases, a key operation is a solution of a linear system of equations given by

$$\widetilde{P}_{\beta_i}^\top \widetilde{N}^{-1} \widetilde{P}_{\beta_i}\, \mathtt{s}_{\beta_i} = \widetilde{P}_{\beta_i}^\top \widetilde{N}^{-1} \widetilde{d}, \tag{17}$$

[1] We note that in sampling from the posterior, some priors for the signal would typically also be postulated, which would lead to a different system of equations than the one studied in this work. We leave this case to follow-up work.

for a sequence of tentative values of the spectral parameters, $\beta_i$. These can be either a chain produced as a result of sampling, or a sequence of values obtained in the course of a minimization. We note that Eq. (17) is essentially a so-called map-making equation (e.g., Natoli et al. 2001; Szydlarski et al. 2014; Puglisi et al. 2018), but with a pointing matrix now replaced by $\widetilde{P}_{\beta_i}$. We can thus hope that as in the map-making problem, we can capitalize on special structures of the involved matrices and very efficient iterative solvers for solving linear systems to render the problem feasible. We point out that in the applications considered here, a subsequent value of the parameter $\beta$, that is, $\beta_{i+1}$, can only be known after the system for the current value, $\beta_i$, is fully resolved. A simultaneous computation of all systems for all values of $\beta_i$ is therefore not possible, and any computational speedup has to come from using better solvers for the linear systems and/or their numerical implementations.

When we separate parts that are dependent and independent of $\beta$, the system in Eq. (17) can also be written as

$$\begin{aligned}
&\overbrace{\begin{bmatrix} M_{\beta,1} \\ \vdots \\ M_{\beta,n_{\rm freq.}} \end{bmatrix}^{\top} \begin{bmatrix} P_1^{\top} N_1^{-1} P_1 & & 0 \\ & \ddots & \\ 0 & & P_{n_{\rm freq}}^{\top} N_{n_{\rm freq}}^{-1} P_{n_{\rm freq}} \end{bmatrix}}^{\equiv \widetilde{A}} \overbrace{\begin{bmatrix} M_{\beta,1} \\ \vdots \\ M_{\beta,n_{\rm freq.}} \end{bmatrix}}^{\equiv \widetilde{M}_{\beta}} \mathfrak{s}_{\beta} \\
&\qquad = \begin{bmatrix} M_{\beta,1} \\ \vdots \\ M_{\beta,n_{\rm freq.}} \end{bmatrix}^{\top} \underbrace{\begin{bmatrix} P_1^{\top} N_1^{-1} d_1 \\ \vdots \\ P_{n_{\rm freq}}^{\top} N_{n_{\rm freq}}^{-1} d_{n_{\rm freq.}} \end{bmatrix}}_{= \widetilde{P}^{\top} \widetilde{N}^{-1} \widetilde{d}}
\end{aligned} \tag{18}$$

The approach we propose here is based on two observations. First, our system has some essential similarities to that of the map-making problem, we should therefore be able to capitalize on novel iterative techniques proposed in that case. Second, we expect that consecutive values of $\beta_i$ in realistic sequences should not vary arbitrarily, and therefore subsequent linear systems (17) should show some resemblance. Consequently, it should be possible to shorten the time to solution for the next value of $\beta_{i+1}$ by capitalizing on the solution for the current one, $\beta_i$.

### 2.4. Block-diagonal preconditioner

The block-diagonal preconditioner is the most common preconditioner used in the preconditioned conjugate gradient solvers applied in the context of the CMB map-making problem (Natoli et al. 2001), which has demonstrated a very good performance in a number of applications. It is also the basis for the construction of more advanced preconditioners (e.g., Szydlarski et al. 2014). The block-diagonal preconditioner is derived by replacing the noise covariance $N_{\rm f}^{-1}$ in Eq. (13) by its diagonal. In the map-making case, when pixel-wise ordering is assumed, this leads to a block-diagonal matrix with the block-size defined by the number of the considered Stokes parameters. In the component separation case, this preconditioner is given by $\widetilde{P}_{\beta}^{\top}\,\mathrm{diag}(\widetilde{N}^{-1})\,\widetilde{P}_{\beta}$, and in the pixel-wise ordering, it is block-diagonal. The diagonal block size is now equal to the product of the number of Stokes parameters and the number of sky components, that is, $6 \times 6$ in the specific case considered here. Consequently, the preconditioner can easily be inverted in any ordering scheme adapted.

Hereafter, we denote the $\beta$-independent part of the preconditioner as $\widetilde{B} := \widetilde{P}^{\top}\,\mathrm{diag}(\widetilde{N}^{-1})\,\widetilde{P}$ so that

$$\widetilde{P}_{\beta}^{\top}\,\mathrm{diag}(\widetilde{N}^{-1})\,\widetilde{P}_{\beta} = \widetilde{M}_{\beta}^{\top}\,\widetilde{B}\,\widetilde{M}_{\beta}. \tag{19}$$

By preconditioning the system (17) from the left, we obtain

$$(\widetilde{M}_{\beta}^{\top}\,\widetilde{B}\widetilde{M}_{\beta})^{-1}\widetilde{M}_{\beta}^{\top}\,\widetilde{A}\widetilde{M}_{\beta}\,\mathfrak{s}_{\beta} = (\widetilde{M}_{\beta}^{\top}\,\widetilde{B}\widetilde{M}_{\beta})^{-1}\widetilde{M}_{\beta}^{\top}\,\widetilde{P}^{\top}\widetilde{N}\widetilde{d}. \tag{20}$$

To simplify the notation in the following, we define

$$\mathbb{A} := \widetilde{M}_{\beta}^{\top}\,\widetilde{A}\,\widetilde{M}_{\beta}, \quad \mathbb{B} := \widetilde{M}_{\beta}^{\top}\,\widetilde{B}\,\widetilde{M}_{\beta}, \quad \mathfrak{b} := \widetilde{M}_{\beta}^{\top}\,\widetilde{P}^{\top}\widetilde{N}\,\widetilde{d}. \tag{21}$$

### 2.5. Component mixing

For concreteness, we assume throughout the paper the following component mixing scheme:

$$\begin{aligned}
s_{{\rm f},q} &= \alpha_{{\rm f},1}\, s_{{\rm cmb},q} + \alpha_{{\rm f},2}(\beta_{\rm d})\, s_{{\rm dust},q} + \alpha_{{\rm f},3}(\beta_{\rm s})\, s_{{\rm sync},q}, \\
s_{{\rm f},u} &= \alpha_{{\rm f},1}\, s_{{\rm cmb},u} + \alpha_{{\rm f},2}(\beta_{\rm d})\, s_{{\rm dust},u} + \alpha_{{\rm f},3}(\beta_{\rm s})\, s_{{\rm sync},u},
\end{aligned} \tag{22}$$

which follows the standard assumptions that there is no $Q$ and $U$ mixing, and that the scaling laws for the Stokes parameters $Q$ and $U$ of each components are the same. In the component-wise ordering, such mixing corresponds to the mixing matrix of the form ($I$ is the identity matrix, $2 \times 2$ in this case)

$$M_{\beta,{\rm f}} = \begin{bmatrix} \alpha_{{\rm f},1} I & 0 & \alpha_{{\rm f},2}(\beta_{\rm d})\, I & 0 & \alpha_{{\rm f},3}(\beta_{\rm s})\, I & 0 \\ 0 & \alpha_{{\rm f},1} I & 0 & \alpha_{{\rm f},2}(\beta_{\rm d})\, I & 0 & \alpha_{{\rm f},3}(\beta_{\rm s})\, I \end{bmatrix}. \tag{23}$$

The coefficients $\alpha_{{\rm f},i}$ encode the assumed scaling laws of the CMB, $i = 1$, dust, $i = 2$, and synchrotron, $i = 3$, where the last two depend on unknown scaling parameters, $\beta_{\rm d}$ and $\beta_{\rm s}$. This matrix can be rewritten with the help of the Kronecker product as

$$M_{\beta,{\rm f}} = \begin{bmatrix} \alpha_{{\rm f},1} & 0 & \alpha_{{\rm f},2}(\beta_{\rm d}) & 0 & \alpha_{{\rm f},3}(\beta_{\rm s}) & 0 \\ 0 & \alpha_{{\rm f},1} & 0 & \alpha_{{\rm f},2}(\beta_{\rm d}) & 0 & \alpha_{{\rm f},3}(\beta_{\rm s}) \end{bmatrix} \otimes I. \tag{24}$$

Hereafter, we drop the explicit dependence of the mixing coefficients on $\beta$, denoting them simply as $\alpha_{{\rm f},k}$.

## 3. Solution procedure for the parametric component separation problem

A complete solution to the component separation problem has to successfully address two aspects. First, it needs to propose an efficient approach to solving the sequences of linear systems as in Eq. (20). Second, it has to combine it with an optimized procedure for the efficient determination of the new values of the parameters $\beta$. This study addresses the former problem and focuses on the solution of a sequence of linear systems obtained for some sequences of the spectral parameters. In order to provide a fair comparison of various proposed techniques, we generate a sequence $\{\beta_i\}$ beforehand and therefore, unlike in the actual applications, in our experiments, $\beta_{i+1}$ is in fact independent of the results of the preceding solution. This ensures that the performance of all the considered solvers is evaluated on the identical sequences of linear systems.

The overall solution scheme we adapt here is then as follows:

(0) Initialize $\beta_0$ and $\mathfrak{s}_{\beta_0}^{(0)}$ (typically $\mathfrak{s}_{\beta_0}^{(0)} := 0$), set $i := 0$.

(1) Given $\beta_i$ and the initial guess $\mathfrak{s}_{\beta_i}^{(0)}$, solve the preconditioned problem, Eq. (20), deriving the current approximation $\mathfrak{s}_{\beta_i}^{(\rm final)}$.

(2a) Determine the new parameters $\beta_{i+1}$.

(2b) Compute a new deflation space for the system associated with $\beta_{i+1}$ using a recycling technique (see details below). This should not involve the value of $\beta_{i+1}$ so that this step can be made in parallel with (2a).

(3) Compute the initial guess $\mathbb{s}^{(0)}_{\beta_{i+1}}$.

(4) Set $i := i + 1$ and go to (1).

In the subsections below, we discuss steps (1), (2b), and (3) in more detail.

### 3.1. PCG with deflation and two-level preconditioners

Although the block-diagonal preconditioner has been shown to ensure good performance in the map-making experience, it has been pointed out that even better performance can often be achieved by employing so-called two-level preconditioners (Szydlarski et al. 2014; Puglisi et al. 2018). Such preconditioners are built from the block-diagonal preconditioner, constituting the first level, and the second level is constructed from a limited number of vectors that are to be deflated (i.e., suppressed in the operator) in order to accelerate the convergence. These vectors are typically taken to be approximate eigenvectors of the system matrix corresponding to its smallest eigenvalues, which often hamper the convergence of PCG with the block-diagonal preconditioner.

We start from the case of deflation for the (unpreconditioned) conjugate gradient (CG) method. CG applied to a linear system $\mathbb{A}\mathbb{s} = \mathbb{b}$ with a given initial vector $\mathbb{s}^{(0)}$ and an initial residual $\mathbb{r}^{(0)} := \mathbb{b} - \mathbb{A}\mathbb{s}^{(0)}$ builds implicitly orthogonal (residuals) and $\mathbb{A}$-orthogonal (search directions) bases of the Krylov subspace,

$$\mathcal{K}_j(\mathbb{A}, \mathbb{r}^{(0)}) = \mathrm{span}\{\mathbb{r}^{(0)}, \mathbb{A}\mathbb{r}^{(0)}, \mathbb{A}^2\mathbb{r}^{(0)}, \ldots, \mathbb{A}^{j-1}\mathbb{r}^{(0)}\}, \tag{25}$$

and the $j$th CG approximation $\mathbb{s}^{(j)} \in \mathbb{s}^{(0)} + \mathcal{K}_j(\mathbb{A}, \mathbb{r}^{(0)})$ is determined by the orthogonality condition on the $j$th residual $\mathbb{r}^{(j)} := \mathbb{b} - \mathbb{A}\mathbb{r}^{(j)}$,

$$\mathbb{r}^{(j)} \perp \mathcal{K}_j(\mathbb{A}, \mathbb{r}^{(0)}). \tag{26}$$

For a given set of deflation vectors, that is, the vectors to be suppressed, we denote by $\mathcal{U}$ the subspace spanned by these vectors. The deflation techniques replace the original operator $\mathbb{A} : \mathbb{R}^n \to \mathbb{R}^n$ by a deflated operator $\widehat{\mathbb{A}} : (\mathbb{R}^n \setminus \mathcal{U}) \to (\mathbb{R}^n \setminus \mathcal{U})$. The approximation is then sought over the augmented subspace (see, e.g., Gaul et al. 2013),

$$\mathbb{s}^{(j)} \in \widehat{\mathbb{s}}_0 + \mathcal{K}_j(\widehat{\mathbb{A}}, \widehat{\mathbb{r}}^{(0)}) \cup \mathcal{U}, \tag{27}$$

and the $j$th residual is required to be orthogonal to $\mathcal{K}_j(\widehat{\mathbb{A}}, \widehat{\mathbb{r}}^{(0)}) \cup \mathcal{U}$. This effectively prevents the solver from exploring the subspace $\mathcal{U}$.

An extension of this for the PCG with the (first-level) preconditioner $\mathbb{B}$ is straightforward because we can use the PCG to implicitly build the Krylov subspace $\mathcal{K}_j(\mathbb{B}^{-1}\mathbb{A}, \mathbb{B}^{-1}\mathbb{r}^{(0)})$. In the considered application, the preconditioner $\mathbb{B}$ is the block-diagonal preconditioner. There are many variants of two-level preconditioners, and we summarize them briefly in Appendix B.2. A more thorough survey can be found in Tang et al. (2009), for example.

Each iteration of a deflated (P)CG, that is, with or without the first level, is more costly than a single iteration of a standard (P)CG. The additional cost primarily depends on the number of deflated vectors, that is, the dimension of $\mathcal{U}$, but also on the deflation variant. Building the subspace $\mathcal{U}$ typically requires some preliminary computations, which can be as costly as solving the system (see, e.g., Szydlarski et al. 2014; Puglisi et al. 2018). Another approach, applicable to the cases when multiple systems need to be solved, is to construct the vectors "on the fly" during the solution of the systems themselves, thus hiding the additional cost. This is the approach we detail in the next sections.

### 3.2. Subspace recycling

Several constructions of the deflation space have been adapted to solving a sequence of linear systems, for instance, those of Saad et al. (2000), Parks et al. (2006), Kilmer & de Sturler (2006), O'Connell et al. (2017), and Jolivet & Tournier (2016). In this work, where the system matrix is symmetric positive definite (SPD), we follow Saad et al. (2000). We build a subspace $\mathcal{Z} \subset \mathcal{K}_j(\widehat{\mathbb{A}}, \widehat{\mathbb{r}}^{(0)})$ by storing some of the vectors computed during the previous run of (P)CG solver and determine the slowest eigenvectors of the operator $\mathbb{A}$ restricted on the subspace $\mathcal{U} \cup \mathcal{Z}$. These are taken as the deflation vectors for the next solution. The resulting algorithm is given in Appendix C.

We can determine the subspace $\mathcal{Z}$ using either the residual or the search direction vectors forming (assuming the exact arithmetic) the orthogonal or an $\widehat{\mathbb{A}}$-orthogonal basis of $\mathcal{K}_j(\widehat{\mathbb{A}}, \widehat{\mathbb{r}}^{(0)})$. Following Saad et al. (2000), we choose here to use the search direction vectors. We retain the first $\dim_{\mathrm{p}}$ search direction vectors, where $\dim_{\mathrm{p}}$ defines the dimension of the so-called recycle subspace. We use the first vectors because the orthogonality among the computed vectors is gradually lost in CG; it is therefore better preserved in the initial iterations.

The techniques for approximating $k$ eigenvectors of the operator over a given subspace are well established. Among them, we note the Ritz and harmonic Ritz projections, which are described in detail in Appendix B.1.1. They lead to solving a (generalized) eigenvalue problem of small dimension, in our case, of $\dim(\mathcal{U}) + \dim(\mathcal{Z})$. While Saad et al. (2000) suggested using the harmonic Ritz projection, we found the Ritz projection slightly more efficient in our numerical experiments, and we therefore include this in the full algorithm described in Appendix C. In another difference with Saad et al. (2000), we assemble the (small) generalized eigenvalue problem matrices in the harmonic Ritz projection using the matrix-matrix products (see Algorithm 2 in Appendix B.1.1) instead of the optimized algorithm from (Sect. 5.1, Saad et al. 2000). This is because we expect that the additional computational cost in our application is negligible and we therefore opted for simplicity.

There is no general recommendation for the choice of the number of deflation vectors, $k$, and the dimension of the recycling subspace, $\dim_{\mathrm{p}}$. Higher $k$ may result in an increase of the overall number of matrix-vector products (the system matrix has to be applied to $k$ deflation vectors before the deflated (P)CG is started for each system) and high $\dim_{\mathrm{p}}$ may cause numerical instabilities in solving the eigenvalue problems that determine the new deflation vectors. On the other hand, the low values of $k$ and $\dim_{\mathrm{p}}$ may not speed up the process sufficiently. We test this numerically in Sect. 4.

### 3.3. Effect of the eigenvalue multiplicity

One limiting factor to the efficiency of this approach, and more generally, to the performance of any two-level preconditioner with the deflation space estimated using standard iterative techniques such as Arnoldi or Lanczos iterations, comes from a

higher multiplicity of eigenvalues, that is, multiple eigenvectors with the same corresponding eigenvalue. This can arise either as a result of some symmetries in the scanning strategies in the case of the map-making systems of equations, or as similarities in the noise covariances of the different single-frequency maps in the case of the component separation problem as studied here; see Appendix A for an example. Admittedly, such symmetries and/or similarities are not typically expected in the cases of real data analysis, but they can arise in the cases of simulated data, in particular if simplifying assumptions are adopted in order to speed up and/or simplify the simulation process.

To shed light on this problem, we consider an SPD matrix $A$ and assume that $\lambda$ is an eigenvalue with multiplicity higher than one. This means that there exists a subspace $V$ with $\dim(V) > 1$ such that

$$Av = \lambda v, \quad \forall v \in V. \tag{28}$$

Let $w$ be an arbitrary vector used to initiate the construction of a Krylov subspace and $w_V$ its projection onto $V$, that is,

$$w = w_V + w', \quad w_V \in V,\ w' \perp V. \tag{29}$$

Then

$$A^\ell w = A^\ell w_V + A^\ell w' = \lambda^\ell w_V + A^\ell w', \quad A^\ell w' \perp V. \tag{30}$$

Therefore, the $j$th Krylov subspace satisfies

$$\begin{aligned}\mathcal{K}_j(A, w) &= \mathrm{span}\{w, Aw, A^2w, \ldots, A^{j-1}w\} \\ &= \mathrm{span}\{w_V\} \cup \mathrm{span}\{w', Aw', A^2w', \ldots, A^{j-1}w'\},\end{aligned} \tag{31}$$

and the intersection of $\mathcal{K}_j(A, w)$ and $V$ is at most a one-dimensional subspace spanned by $w_V$,

$$\mathcal{K}_j(A, w) \cap V = \mathrm{span}\{w_V\}. \tag{32}$$

Consequently, methods based on the Krylov subspace approximation, therefore including Lanczos and Arnoldi iterations (see Appendix B.1.2 for more details), can recover one vector at most from the subspace spanned by multiple eigenvectors with the same eigenvalue. This may not be sufficient to allow for a construction of an efficient two-level preconditioner, however, in particular if the eigenvalue with many corresponding eigenvectors happens to be small: with the single precomputed vector we can only deflate a one-dimensional subspace of the entire multidimensional space as spanned by all these eigenvectors, and the remainder may continue hampering the convergence.

This problem can be overcome by using a more advanced eigenvalue solver that can detect and handle the higher multiplicity of eigenvalues. An efficient implementation of such a solver is for instance provided by the ARPACK library (Lehoucq et al. 1998). In this case, the preconditioner may need to be precomputed with the help of such advanced routines, instead of constructing it on the fly as proposed here. If the presence of the eigenvalue multiplicity and the corresponding eigenvectors is known ahead of time, these vectors can be accomodated in the on-the-fly procedure proposed here. This is indeed the case we have encountered in one of the test cases discussed below.

We point out that the multiplicity of the eigenvalues is in principle advantageous for the standard (P)CG. In exact arithmetic, the effect of the whole subspace might be then eliminated at the cost of a single iteration. This fact is often used in the analysis of preconditioning methods based on preconditioners shifting some, possibly many, eigenvalues to the same value.

Last but not least, we emphasize that an eigenvalue multiplicity implies neither any indistinguishability of the corresponding eigenvectors nor a presence of degenerate modes in the solution, at least as long as the eigenvalue is not (numerically) zero. If the eigenvalue is not zero, the multiplicity only tells us that components of the right-hand side of the linear system that belong to the subspace spanned by the corresponding eigenvectors are merely weighted by the inverse system matrix in exactly the same way.

### 3.4. Choice of the initial guess

The simplest and standard way to solve the sequence is to run the PCG method with the initial guess set to zero. However, some evidence exists showing that at least in the map-making case, this may not always be the best choice (Papež et al. 2018), in particular in cases with high signal-to-noise ratios. In the case of a sequence of linear systems, all the systems involve the same initial data set with the same signal and noise content. Even in data with a low signal-to-noise ratio, it may therefore be expected that adapting the initial guess following previous results may speed the process up in an interesting way. Consequently, we explore here two alternatives and show by numerical experiments that they are indeed much more efficient.

#### 3.4.1. Previous solution as the initial guess

A natural idea is to run the PCG for the (new) problem corresponding to $\beta_{i+1}$ starting with the computed approximation $\mathfrak{s}_{\beta_i}^{(\mathrm{final})}$,

$$\mathfrak{s}_{\beta_{i+1}}^{(0)} := \mathfrak{s}_{\beta_i}^{(\mathrm{final})}. \tag{33}$$

This can be in particular efficient when the parameters $\beta_i$ and $\beta_{i+1}$ do not significantly differ and it is expected that so do $\mathfrak{s}_{\beta_i}$ and $\mathfrak{s}_{\beta_{i+1}}$.

#### 3.4.2. Adapted previous solution as the new initial guess

Equation (33) can be further adapted by capitalizing on the particular structure of the mixing matrix. To start, we rewrite Eq. (17) as

$$\widetilde{M}_\beta^\top (\widetilde{A}\widetilde{M}_\beta\, \mathfrak{s}_\beta - \widetilde{P}^\top \widetilde{N}^{-1}\widetilde{d}) = 0. \tag{34}$$

If the matrix $\widetilde{M}_\beta$ were square (and nonsingular), then

$$\widetilde{M}_\beta\, \mathfrak{s}_\beta = \widetilde{A}^{-1}\widetilde{P}^\top \widetilde{N}^{-1}\widetilde{d} \tag{35}$$

would be the vector independent of $\beta$. The solution $\mathfrak{s}_\beta$ might then be interpreted as the coefficients with respect to the basis given by the columns of $\widetilde{M}_\beta$. Therefore we would have

$$\mathfrak{s}_{\bar\beta} = (\widetilde{M}_{\bar\beta})^{-1}\widetilde{M}_\beta\, \mathfrak{s}_\beta \tag{36}$$

for arbitrary $\bar\beta$ (for which $\widetilde{M}_{\bar\beta}$ is nonsingular).

In our case, matrix $\widetilde{M}_\beta$ is rectangular of size $2\, n_{\mathrm{freq}}\, n_{\mathrm{pix}} \times 6\, n_{\mathrm{pix}}$ and has full column rank. When the number of frequencies $n_{\mathrm{freq}}$ is not significantly higher than 3, we can generalize the above idea and use as the initial guess for the new system the vector

$$\mathfrak{s}_{\beta_{i+1}}^{(0)} := (\widetilde{M}_{\beta_{i+1}})^\dagger \widetilde{M}_{\beta_i}\, \mathfrak{s}_{\beta_i}^{(\mathrm{final})}, \tag{37}$$

where $M^\dagger$ is the (Moore–Penrose) pseudo-inverse of $M$,

$$M^\dagger \equiv (M^\top M)^{-1} M^\top. \tag{38}$$

We recall our assumption that $M$ is of full column rank. Clearly, for $\beta_{i+1} = \beta_i$,

$$(\widetilde{M}_{\beta_{i+1}})^\dagger \widetilde{M}_{\beta_i} = I \quad \text{holds, and therefore} \quad (\widetilde{M}_{\beta_{i+1}})^\dagger \widetilde{M}_{\beta_i}\, \mathfrak{s}_{\beta_i} = \mathfrak{s}_{\beta_i}. \tag{39}$$

Finally, we note that the computation of the vector in Eq. (37) is very cheap because of the Kronecker structure (24) of the matrices $\widetilde{M}_\beta$. Writing $\widetilde{M}_\beta = K_\beta \otimes I$, we obtain

$$(\widetilde{M}_{\beta_{i+1}})^\dagger \widetilde{M}_{\beta_i} = \left((K^\top_{\beta_{i+1}} K_{\beta_{i+1}})^{-1} K^\top_{\beta_{i+1}} K_{\beta_i}\right) \otimes I, \tag{40}$$

in other words, only the matrices of size $2\,n_{\rm freq} \times 6$ need to be handled, and the cost of the proposed adaptation is nearly negligible.

## 4. Numerical experiments

### 4.1. Simulated data

For our numerical tests we use a simulated data set composed of time-ordered multifrequency observations with a correlated, "1/f", noise. The characteristics of this data set are as follows.

#### 4.1.1. Pointing matrix

We adopt the simple scanning strategy used in Papež et al. (2018). The entire time-ordered data set is composed of $O(10^8)$ measurements per frequency and divided into four consecutive subsets. The pointing is assumed to be the same for each frequency. The underlying sky is pixelized using the Healpix pixelization scheme (Górski et al. 2005) with the resolution parameter $n_{\rm side}$ set to 1024. The scan consists of repetitive scanning of a rectangular patch made of 256 pixel rows and columns. The scanning is either horizontal, that is, along the pixel rows, for the first and third subset, or vertical, that is, along the pixel columns for the second and fourth subset. During a single left-to-right, or bottom-up sweep, each sky pixel is sampled only once, and the direction of the polarizer, $\varphi_t$, is fixed for each of the four subsets and is equal, with respect to the sky, to 0, $\pi/4$, $\pi/2$, and, $3\pi/4$.

The sky signal contribution to every measurement is modeled as

$$d_c(t) = Q^\star_c(p(t)) \cos 2\varphi_t \;+\; U^\star_c(p(t)) \sin 2\varphi_t, \tag{41}$$

where $p(t)$ denotes the pixel observed at time $t$, we do not include the total intensity, and $Q_c$ and $U_c$ stand for $Q$ and $U$ Stokes parameters of the combined, CMB + foregrounds, sky signal observed at frequency $\nu_c$.

#### 4.1.2. Sky maps

We assume six frequency channels that approximately correspond to those accessible for a ground-based experiment. These are

$$\nu_c \in \left\{30, 40, 90, 150, 220, 270\right\} \text{GHz}. \tag{42}$$

The sky signal is composed of emissions from three sources: CMB, dust, and synchrotron. The CMB signal is simulated using the current best-fit CMB model (Planck Collaboration XIII 2016), while we use the so-called COMMANDER templates (Planck Collaboration X 2016) to model the dust and synchrotron signals that we scale to our reference frequency, $\nu_{\rm ref}$ = 150 GHz, using Planck's fiducial laws.

For the scaling laws we take a blackbody for the CMB component ($T_{\rm CMB}$ = 2.7525 K), a power law for the synchrotron, and a modified blackbody for the dust, therefore

$$\mathcal{S}^{\rm sync}(\nu, \beta^\star_{\rm s}) = \nu^{\beta^\star_{\rm s}} \tag{43}$$

$$\mathcal{S}^{\rm dust}(\nu, \beta^\star_{\rm d}, T^\star_{\rm d}) = \left(\frac{h\nu}{kT^\star_{\rm d}}\right)^{\beta^\star_{\rm d}} B(\nu, T^\star_{\rm d}), \tag{44}$$

where the star distinguishes the true values of the parameters,

$$\beta^\star \equiv [\beta^\star_{\rm s}, \beta^\star_{\rm d}, T^\star_{\rm d}] = [-3.1, 1.59, 19.6\,\text{K}], \tag{45}$$

and $B(\nu, T)$ denotes a blackbody at temperature, $T$. The simulated maps are expressed in thermodynamic units and are given by

$$Q^\star_{\rm p}(\nu) = Q^{\rm cmb,\,\star}_{\rm p} + \Gamma_{\rm RJ}(\nu)\left[\frac{\mathcal{S}^{\rm dust}(\nu, \beta^\star_{\rm d}, T^\star_{\rm d})}{\mathcal{S}^{\rm dust}(\nu_{\rm ref}, \beta^\star_{\rm d}, T^\star_{\rm d})}\, Q^{\rm dust,\,\star}_{\rm p}(\nu_{\rm ref}) + \frac{\mathcal{S}^{\rm sync}(\nu, \beta^\star_{\rm s})}{\mathcal{S}^{\rm sync}(\nu_{\rm ref})}\, Q^{\rm sync,\,\star}_{\rm p}(\nu_{\rm ref}, \beta^\star_{\rm s})\right] \tag{46}$$

for each frequency $\nu = \nu_c$ and each observed sky pixel $p$. An analogous formula holds for the Stokes $U$ parameter. Here, $\Gamma_{\rm RJ}(\nu)$ stands for a conversion factor from Rayleigh-Jeans to thermodynamic units. This expression is consistent with Eq. (24) upon a suitable definition of the coefficients $\alpha$.

In our numerical experiments, we fix the dust temperature, $T_{\rm d}$, to its true value and assume that only the spectral indices, $\beta = [\beta_{\rm s}, \beta_{\rm d}]$, are determined from the data. We assume that these are estimated by maximizing the spectral likelihood, Eq. (16), using a truncated Newton maximization procedure. We use this approach to generate a single sequence of $\{\beta_i\}$, which, as explained in Sect. 3, we adopt consistently in all our runs. The sequence is made of 26 values and is shown in Fig. 1. In Appendix D we show for completeness the results of a similar test, but performed for a sequence of $\beta$ derived by sampling of the spectral likelihood. The main conclusions derived in these two examples are consistent.

#### 4.1.3. Noise

We assume a correlated noise in the time domain with a spectrum given by

$$P(f) = \sigma^2_{\rm rms}\left(1 + \frac{f_{\rm knee}}{f}\right), \tag{47}$$

where $f$ is the time-domain frequency. The values of $f_{\rm knee}$ adopted here are different for different frequency channels and taken to be such that there are strong noise correlations within a single sweep. They span the range from 0.5 up to 3 Hz from the lowest to the highest frequency channel, respectively. The noise is apodized at very low frequencies, so that the noise power is finite. $\sigma^2_{\rm rms}$ is taken to be about 30 $\mu$K per sample, reflecting the fact that each measurement effectively corresponds to the combined measurement of many modern detectors operating at the same frequency. This leads to sky maps with a noise $\sigma^{Q/U}_{\rm rms} \sim 2$–$3\,\mu$K per pixel.

### 4.2. Multiplicity of the eigenvalues as a result of the particular scanning strategy

In this section we address the issue of multiple eigenvectors with the same eigenvalues, which we have identified in our numerical

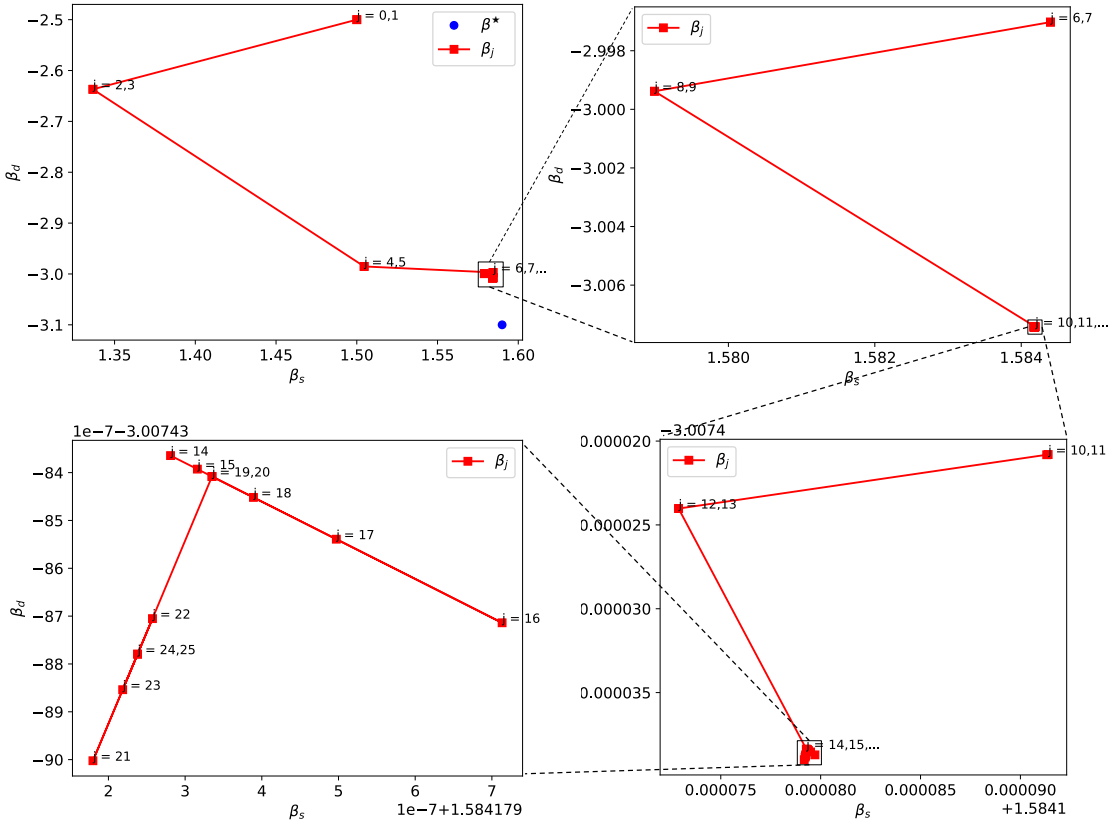


**Fig. 1.** Sequence of the spectral parameters $\beta_i = [\beta_{i,s}, \beta_{i,d}]$ used in our experiments and derived from the maximization of the spectral likelihood, Eq. (16). The panels in clockwise order show consecutive zooms on the sequence that converged on the likelihood peak values of $\beta = [-3.006, 1.584]$, slightly off the true values marked by the blue solid point in the top left panel and given by Eq. (45) (with $T_d$ fixed in our test cases). The sequence was generated as described at the end of Sect. 4.1.2.

tests. In agreement with Sect. 3.3, these were found to have significant effect on the performance of the proposed solvers. We show here how they can be traced back to the specific scanning strategy adopted in our simulations. We then describe corrective measures we included in order to minimize their effect and to ensure that our results are indeed representative of more typical cases. These considerations are given here for completeness and as a potentially useful example. However, as the proposed measures may not be necessary in most of the cases of realistic data, this section can be skipped without affecting the further reading.

We denote two pointing matrices for two horizontal scans as $P_0$ and $P_{\pi/2}$ and two pointing matrices for two vertical scans as $P_{\pi/4}$ and $P_{3\pi/4}$, where the subscript stands for the polarizer angle in the sky coordinates. They are related as

$$P_{\pi/2} = P_0\,\mathcal{R}_2(\pi/4), \qquad (48)$$
$$P_{3\pi/4} = P_{\pi/4}\,\mathcal{R}_2(\pi/4),$$

where $\mathcal{R}$ is a $12\,n_{\rm pix}$-by-$12\,n_{\rm pix}$ block-diagonal matrix with each diagonal block equal to a 2-by-2 spin-2 rotation matrix for each pixel of the six frequency maps. While this is clearly a result of the simplifying assumption made in the simulations, this example may be of interest also in more realistic circumstances where certain relations of this sort may happen to be fulfilled approximately following from some common but inexact symmetries of typically adopted scans. We therefore briefly explore the consequences of this here.

In the case at hand, we can represent the total pointing matrix as

$$\widetilde{P} = \begin{bmatrix} P_0 \\ P_{\pi/4} \\ P_{\pi/2} \\ P_{3\pi/4} \end{bmatrix} = \begin{bmatrix} P_{-\pi/4} \\ P_0 \\ P_{\pi/4} \\ P_{\pi/2} \end{bmatrix} \mathcal{R}_{\pi/4} = \begin{bmatrix} P_{3\pi/4} \\ P_0 \\ P_{\pi/4} \\ P_{\pi/2} \end{bmatrix} \mathcal{R}_{\pi/4} = \widetilde{P}'\,\mathcal{R}_{\pi/4}, \qquad (49)$$

given that all four scan subsets observe exactly the same sky.

When in addition the noise covariance for each of the four scan subsets is exactly the same, we have

$$\widetilde{P}^\top\,\widetilde{N}^{-1}\,\widetilde{P} = \widetilde{P}'^\top\,\widetilde{N}^{-1}\,\widetilde{P}' = \mathcal{R}_{\pi/4}^\top\,\widetilde{P}^\top\,\widetilde{N}^{-1}\,\widetilde{P}\,\mathcal{R}_{\pi/4}. \qquad (50)$$

We note that this holds if the noise properties vary from one frequency channel to another, as is indeed the case in our simulations.

If now $v$ is an eigenvector of the matrix $\widetilde{A} = \widetilde{P}^\top\,\widetilde{N}^{-1}\,\widetilde{P}$ with a corresponding eigenvalue, $\lambda_v$, then

$$\widetilde{A}\,v = \widetilde{P}^\top\,\widetilde{N}^{-1}\,\widetilde{P}\,v = \mathcal{R}_{\pi/4}^\top\,\widetilde{P}^\top\,\widetilde{N}^{-1}\,\widetilde{P}\,\mathcal{R}_{\pi/4}\,v = \lambda_v\,v, \qquad (51)$$

and therefore,

$$\widetilde{A}\,\mathcal{R}_{\pi/4}\,v = \widetilde{P}^\top\,\widetilde{N}^{-1}\,\widetilde{P}\,\mathcal{R}_{\pi/4}\,v = \lambda_v\,\mathcal{R}_{\pi/4}\,v. \qquad (52)$$

This means that also $\mathcal{R}_{\pi/4}\,v$ is an eigenvector of the matrix $A$ with the same eigenvalue, $\lambda_v$. Because this reasoning applies as much to the matrix $A$ as the matrix $B = \widetilde{P}^\top\,\mathrm{diag}\,\widetilde{N}^{-1}\,\widetilde{P}$, we have

$$\widetilde{P}^\top\,\widetilde{N}^{-1}\,\widetilde{P}\,v = \lambda'_v\,\widetilde{P}^\top\,\mathrm{diag}\,\widetilde{N}^{-1}\,\widetilde{P}\,v \qquad (53)$$
$$\widetilde{P}^\top\,\widetilde{N}^{-1}\,\widetilde{P}\,(\mathcal{R}_{\pi/4}\,v) = \lambda'_v\,\widetilde{P}^\top\,\mathrm{diag}\,\widetilde{N}^{-1}\,\widetilde{P}\,(\mathcal{R}_{\pi/4}\,v).$$

In general, this does not yet imply that the component separation system matrix preconditioned with the block-diagonal preconditioner, Eq. (21), given by

$$\mathbb{B}^{-1}\,\mathbb{A} = (\widetilde{M}_\beta^\top\,\widetilde{B}\,\widetilde{M}_\beta)^{-1}(\widetilde{M}_\beta^\top\,\widetilde{A}\,\widetilde{M}_\beta), \tag{54}$$

has two eigenvectors corresponding to the same eigenvalue related by the rotation operator acting in the component space. This is the case when the subspace spanned by $v$ and $\mathcal{R}_{\pi/4}\,v$ is contained in the subspace spanned by the columns of the mixing matrix, $\widetilde{M}_\beta$. Otherwise, the preconditioned system matrix may have a single (when these two subspaces merely intersect) or no corresponding eigenvectors (when these two subspaces are disjoint). Which of these situations is actually realized is case dependent and in general also depends on the value of $\beta$.

We found that in our numerical experiments the eigenvalue multiplicity of the preconditioned system matrix due to the assumed scan symmetries was sufficiently common that we opted to account for it explicitly in our analysis. Consequently, we use the subspace recycling to approximate one of the eigenvectors, and we compute the other by applying the rotation operator. We then use both vectors to construct the deflation operator. Given that $\mathcal{R}_{\pi/4} = -\mathcal{R}_{-\pi/4}$, there is no ambiguity because we do not know a priori which of the two vectors we estimate directly through the subspace recycling, and this approach leads to the same deflation space, regardless of the rotation that is applied. This technique has led to a significant speed-up in the cases studied below.

### 4.3. Results

We compare the convergence using the relative norm of the $j$th residual,

$$\frac{\|\widetilde{M}_\beta^\top\,\widetilde{P}^\top\widetilde{N}\widetilde{d} - \widetilde{M}_\beta^\top\,\widetilde{A}\widetilde{M}_\beta\,\mathfrak{s}_\beta^{(j)}\|}{\|\widetilde{M}_\beta^\top\,\widetilde{P}^\top\widetilde{N}\widetilde{d}\|}. \tag{55}$$

The iterations for each system are stopped when this value drops below tolerance TOL = $10^{-8}$, but we always perform at least one iteration.

We first show that the systems corresponding to different $\beta$s from the sequence are indeed "close to each other" in the sense that they display a similar behavior during the solution process. We illustrate this by showing the convergence of PCG with zero initial guess in Fig. 2. We find that all the convergence curves are indeed very similar, even for the initial elements of the sequence where the values of $\beta$ parameters continue to change quite significantly.

We can therefore focus on the "true" system corresponding to $\beta = \beta^\star$ in order to investigate the improvement caused by deflating the eigenvectors corresponding to the smallest eigenvalues of the system matrix. This is depicted in Fig. 3. Here, the eigenvectors are computed using the ARPACK eigensolver (Lehoucq et al. 1998), and as expected, we find that significant improvement is achieved by the deflation.

Then, we illustrate the effect of the deflation space built by recycling. To do this, we first consider six systems and start the iterations always with zero initial guess, $\mathfrak{s}^{(0)} = 0$. The result for $k = 10$ eigenvectors approximated using the dimension of the subspace, $\dim_p = 100$, is given in Fig. 4.

In Fig. 5 we compare the convergence of the full sequence of the 26 systems using the techniques of setting the initial guess proposed in Eqs. (33) and (37) and using subspace recycling. We recall that standard PCG with zero initial vector takes more than 6000 iterations; cf. Fig. 2. Although the subspace recycling variant requires $25 \times 10$ matrix-vector products[2] more than the PCG with block-Jacobi preconditioner for any choice of the initial guess, it still provides an interesting increase in speed.

We also compare the time required by one iteration of the PCG with a deflation with the time required by a standard PCG iteration. In Table 1 we list the relative times of a single iteration of a deflated PCG with 2, 6, and 10 deflation vectors in our experiments (taking an average from five runs, each with ten iterations). The small and negligible overhead introduced by the two-level preconditioner indicates that most of the time in the code is spent on the standard map-making operations,

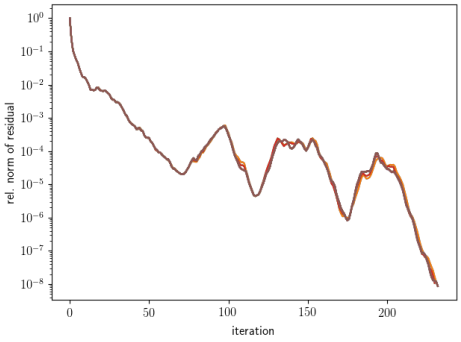


**Fig. 2.** Convergence of PCG with zero initial guess and block-Jacobi preconditioner for all 26 systems in the sequence shown in Fig. 1.

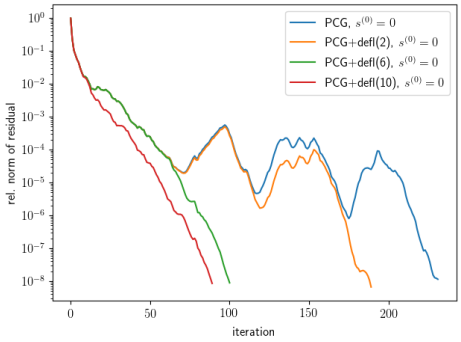


**Fig. 3.** Convergence of PCG with the deflation applied to 2, 6, and 10 slowest eigenvectors for the true system with $\beta = \beta^\star$.

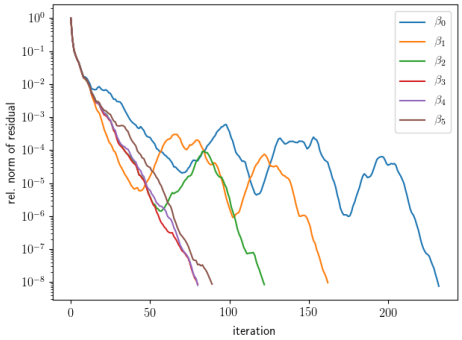


**Fig. 4.** Convergence of deflated PCG with the deflation subspace build by recycling. Here we consider the first six systems from the sequence and start the iterations always with zero initial guess. $k = 10$ eigenvectors are approximated using $\dim_p = 100$.

[2] It is necessary to apply the matrix to deflation vectors at the beginning of the deflated PCG to build the projection matrices, see Algorithm 1 in Appendix C.

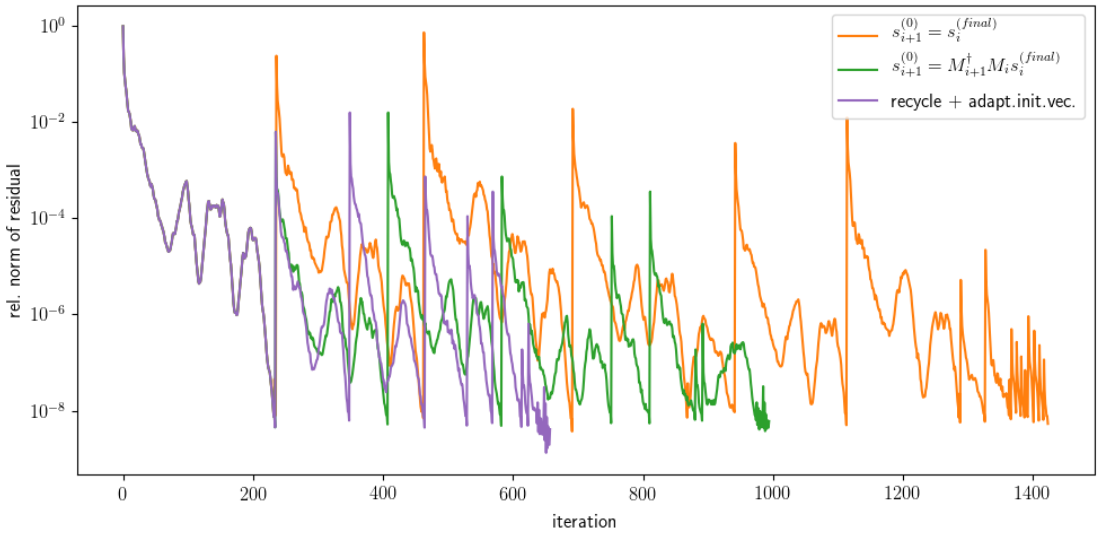


**Fig. 5.** Comparison of the PCG with different choices of initial guess (as in Eqs. (33) and (37)) and the PCG with the subspace recycling (together with the choice of the initial guess as in Eq. (37)). For the recycling we consider $k = 10$ eigenvectors approximated using $\dim_p = 100$.

**Table 1.** Timings of a single deflated PCG iteration with 2, 6, and 10 deflation vectors.

| PCG | PCG + defl(2) | PCG + defl(6) | PCG + defl(10) |
|---|---|---|---|
| 1 | 1.0001 | 1.0013 | 1.0019 |

**Notes.** The timings are relative with respect to a single iteration of the standard (nondeflated) PCG. The table gives the average from five runs, each with ten iterations.

such as (de)projection operators and noise weighting (e.g., Cantalupo et al. 2010). We emphasize that these timings depend on the implementation and the hardware on which it is executed. For massively parallel codes, the cost of a deflated PCG iteration compared to a standard PCG iteration may increase somewhat, but we expect that the proposed algorithm should nevertheless still be offering an attractive increase in speed.

As noted above, the performance of the PCG with the deflation space built by recycling is affected by the number of the deflation vectors, $k$, and the dimension of the recycling subspace, $\dim_p$. There is no general recommendation for their choice. Higher $k$ may result in an increase of the overall number of matrix-vector products (the system matrix has to be applied to $k$ deflation vectors before the deflated PCG is started for each system) and high $\dim_p$ may cause numerical instabilities in solving the eigenvalue problems that determine the new deflation vectors. On the other hand, the low values of $k$ and $\dim_p$ may not increase the speed sufficiently. We compare the convergence of PCG with some choices of $k$ and $\dim_p$ in Fig. 6 and in Table 2. The deflation clearly has a small effect for small $\dim_p$, that is, when the eigenvectors are not approximated accurately.

## 5. Conclusions and further perspectives

We have presented a procedure for efficiently solving a sequence of linear systems arising from the CMB parametric component separation problem. Two main novelties are the proposed choice of the initial vectors and the recycling technique used to determine the deflation space. Motivated by our simulated data, we also emphasized and elaborated on the role of the multiplicity of the eigenvalues, in particular in the context of their effect on the performance of two-level preconditioners.

The overall speed-up factor we obtained, ∼5−7, is significant. The bulk of the improvement comes from reusing the solution of an earlier system as the initial guess of the next solution – a simple but not trivial observation owing to the fact that this is the same data set being used in all the solutions. However, other proposed amendments provide a significant additional performance boost on the order of ∼2. This is particularly significant in the case of the sampling-based application. Further improvements and optimizations are clearly possible. For instance, the number of required matrix-vector products can be decreased by not using the two-level preconditioner for all the systems. As the experiments showed, when two consecutive system solutions are very similar, the PCG with the diagonal preconditioner and a proper initial guess (e.g., as proposed in Sect. 3.4.2) can already be sufficient for convergence in a few iterations.

We emphasize that in practice, we will be only able to capitalize on this type of approach when they are implemented in a form of highly efficient high-performance massively parallel numerical algorithms and codes. We leave a full demonstration of this to future work, noting here only that many of the relevant techniques have been studied in the past and recent literature, showing that this should indeed be feasible (e.g., Cantalupo et al. 2010; Sudarsan et al. 2011; Szydlarski et al. 2014; Papež et al. 2018; Seljebotn et al. 2019).

The techniques we presented, rendered in the form of efficient high-performance codes, should allow for the efficient maximization of the data likelihood or the posterior distributions in the component separation problems and produce reliable sky component estimates for at least some of the forthcoming data sets. However, in the cases of sampling algorithms, when many thousand linear systems may need to be solved, this still remains to be demonstrated, and further improvements will likely be required. They will depend in general on specific details of the employed sampling technique, however, and we leave them here as future work.

The techniques discussed here can also be used in other problems in CMB data analysis that require solving a sequence of linear systems. In particular, they should be directly relevant for applications that estimate instrumental parameters, which

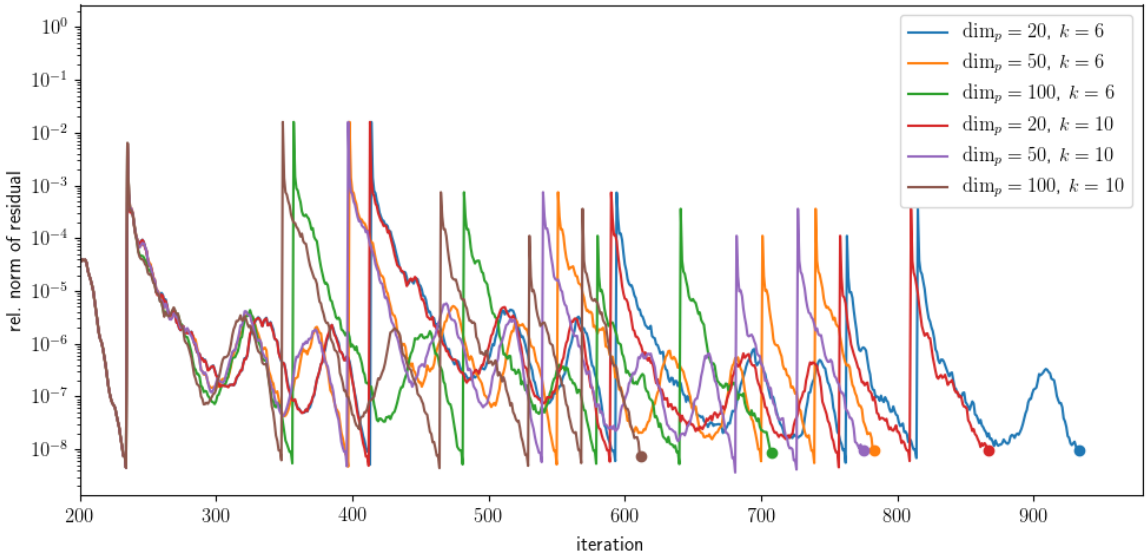


**Fig. 6.** Comparison of the PCG with different choices of $k$ and $\dim_p$ for the first ten systems in the sequence. The initial guess is the same as in Eq. (37). The iteration counts are also listed in Table 2. Because the convergence for the first system is independent of $\dim_p$ and $k$, the $x$-axis is depicted starting from the iteration 200.

**Table 2.** Number of PCG iterations and matrix-vector products (MatVecs) for different choices of $k$ and $\dim_p$ for the first ten systems in the sequence.

| $\dim_p$ | $k$ | #iterations | #MatVecs | |
|---|---|---|---|---|
| | | | Deflation | Total |
| 20 | 6 | 933 | 54 | 987 |
| 50 | 6 | 783 | 54 | 837 |
| 100 | 6 | 708 | 54 | 762 |
| 20 | 10 | 867 | 90 | 957 |
| 50 | 10 | 775 | 90 | 865 |
| 100 | 10 | 612 | 90 | 702 |

**Notes.** The initial guess is the same as in Eq. (37).

commonly have to be included in more realistic data models and estimated from the data.

The codes used in this work are available from a GitHub repository[3].

*Acknowledgements.* We thank Olivier Tissot for insightful discussions and Dominic Beck and Josquin Errard for their help with numerical experiments. The first two authors' work was supported by the NLAFET project as part of European Union's Horizon 2020 research and innovation program under grant 671633. This work was also supported in part by the French National Research Agency (ANR) contract ANR-17-C23-0002-01 (project B3DCMB). This research used resources of the National Energy Research Scientific Computing Center (NERSC), a DOE Office of Science User Facility supported by the Office of Science of the U.S. Department of Energy under Contract No. DE-AC02-05CH11231.

## References

Brandt, W. N., Lawrence, C. R., Readhead, A. C. S., Pakianathan, J. N., & Fiola, T. M. 1994, ApJ, 424, 1
Calvetti, D., Reichel, L., & Sorensen, D. C. 1994, Electron. Trans. Numer. Anal., 2, 1
Cantalupo, C. M., Borrill, J. D., Jaffe, A. H., Kisner, T. S., & Stompor, R. 2010, ApJS, 187, 212
Dostál, Z. 1988, Int. J. Comput. Math., 23, 315
Eriksen, H. K., Jewell, J. B., Dickinson, C., et al. 2008, ApJ, 676, 10
Gaul, A., Gutknecht, M. H., Liesen, J., & Nabben, R. 2013, SIAM J. Matrix Anal. Appl., 34, 495
Górski, K. M., Hivon, E., Banday, A. J., et al. 2005, ApJ, 622, 759
Grigori, L., Stompor, R., & Szydlarski, M. 2012, SC 2012: Proceedings of the International Conference on High Performance Computing, Networking, Storage and Analysis, 1
Hestenes, M. R., & Stiefel., E., 1952, J. Res. Natl. Bureau Stand., 49, 409
Jolivet, P., & Tournier, P. H. 2016, Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis, SC '16 (Piscataway, NJ, USA: IEEE Press), 17:1
Kilmer, M. E., & de Sturler, E. 2006, SIAM J. Sci. Comput., 27, 2140
Lehoucq, R. B., Sorensen, D. C., & Yang, C. 1998, Software, Environments, and Tools. ARPACK Users' Guide (Philadelphia, PA: Society for Industrial and Applied Mathematics (SIAM)), 6, xvi+142
Morgan, R. B. 1995, SIAM J. Matrix Anal. Appl., 16, 1154
Natoli, P., de Gasperis, G., Gheller, C., & Vittorio, N. 2001, A&A, 372, 346
Nicolaides, R. 1987, SIAM J. Numer. Anal., 24, 355
O'Connell, M., Kilmer, M. E., de Sturler, E., & Gugercin, S. 2017, SIAM J. Sci. Comput., 39, B272
Paige, C. C., Parlett, B. N., & van der Vorst, H. A. 1995, Numer. Linear Algebra Appl., 2, 115
Papež, J., Grigori, L., & Stompor, R. 2018, A&A, 620, A59
Parks, M. L., de Sturler, E., Mackey, G., Johnson, D. D., & Maiti, S. 2006, SIAM J. Sci. Comput., 28, 1651
Planck Collaboration X. 2016, A&A, 594, A10
Planck Collaboration XIII. 2016, A&A, 594, A13
Puglisi, G., Poletti, D., Fabbian, G., et al. 2018, A&A, 618, A62
Saad, Y., & Schultz, M. H. 1986, SIAM J. Sci. Stat. Comput., 7, 856
Saad, Y., Yeung, M., Erhel, J., & Guyomarc'h, F. 2000, SIAM J. Sci. Comput., 21, 1909
Seljebotn, D. S., Bærland, T., Eriksen, H. K., Mardal, K. A., & Wehus, I. K. 2019, A&A, 627, A98
Sleijpen, G. L. G., & Van der Vorst, H. A. 2000, SIAM Rev., 42, 267
Sorensen, D. C. 1992, SIAM J. Matrix Anal. Appl., 13, 357
Sorensen, D. C. 2002, Acta Numer., 11, 519
Stewart, G. W. 2001/02, SIAM J. Matrix Anal. Appl., 23, 601
Stompor, R., Leach, S., Stivoli, F., & Baccigalupi, C. 2009, MNRAS, 392, 216
Sudarsan, R., Borrill, J., Cantalupo, C., et al. 2011, Proceedings of the International Conference on Supercomputing, ICS '11 (New York, NY, USA: Association for Computing Machinery), 305
Szydlarski, M., Grigori, L., & Stompor, R. 2014, A&A, 572, A39
Tang, J. M., Nabben, R., Vuik, C., & Erlangga, Y. A. 2009, J. Sci. Comput., 39, 340
Wu, G. 2017, SIAM J. Matrix Anal. Appl., 38, 118
Wu, K., & Simon, H. 2000, SIAM J. Matrix Anal. Appl., 22, 602

[3] https://github.com/B3Dcmb/Accelerated-PCS-solvers

## Appendix A: Eigenvalue multiplicity in the component separation problem. A worked example

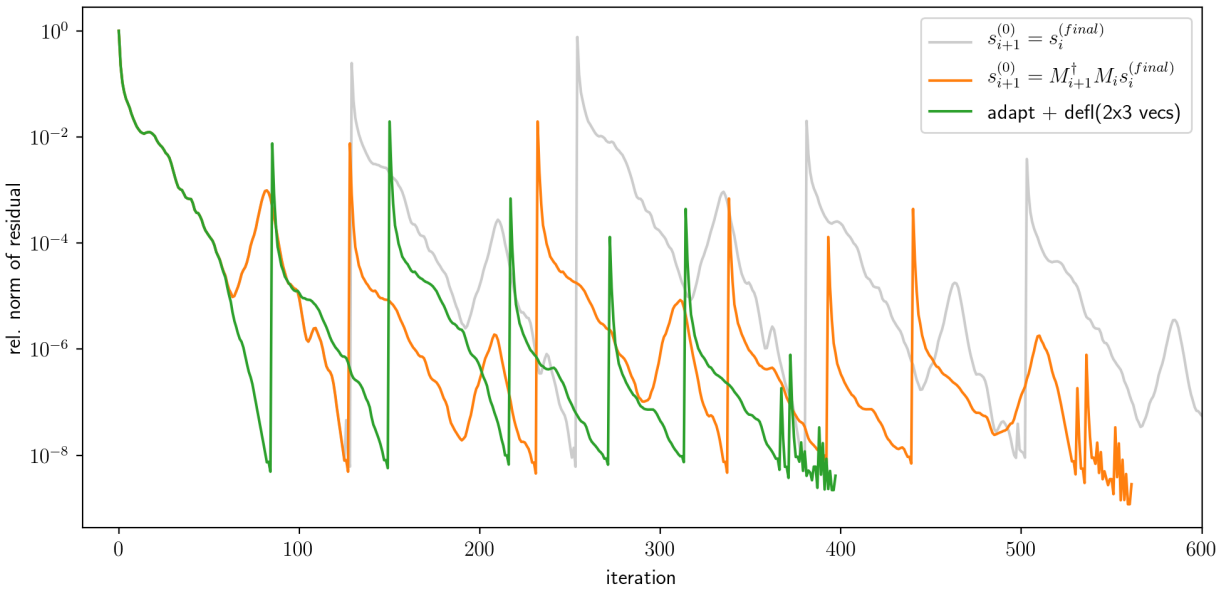


**Fig. A.1.** Same as Fig. 5 for the simplified setting of Appendix A. Comparison of the PCG with different choices of the initial guess (as in Eqs. (33) and (37)) and the deflated PCG with $2 \times 3$ vectors.

In this section we discuss the eigenvector and eigenvalue structure of the preconditioned matrix defined in Eq. (20) in the context of eigenvalue multiplicity in some specific setup that in particular assumes that the pointing matrix is the same for each frequency and that the noise has the same covariance (up to a scaling factor) for each frequency, that is,

$$P_{\rm f} = P, \quad N_{\rm f} = N, \quad f = 1, \ldots, n_{\rm freq.}. \tag{A.1}$$

While these requirements are not very likely to be strictly realized in any actual data set, they can be fulfilled approximately, leading to near multiplicities of the eigenvalues. If these are not accounted for, they may be as harmful to the action of the preconditioner as the exact multiplicities. Moreover, this worked example demonstrates that the component separation problem is in general expected to be more affected by this type of effect than the standard map-making solver, for instance, therefore emphasizing that due diligence is necessary in this former application.

First, let $(\lambda_i, \mathsf{v}_i) = (\lambda_i, \lceil v_{i,q}, v_{i,u} \rfloor)$ be an eigenpair of the map-making matrix, that is, there holds

$$P^\top N^{-1} P\, \mathsf{v}_i = \lambda_i P^\top \operatorname{diag}(N^{-1})\, P\, \mathsf{v}_i. \tag{A.2}$$

We note that

$$\widetilde{M}_\beta \lceil \mathsf{v}_i, 0, 0 \rfloor = \widetilde{M}_\beta \begin{bmatrix} v_{i,q} \\ v_{i,u} \\ 0 \\ \vdots \\ 0 \end{bmatrix} = \begin{bmatrix} \alpha_{1,1}\, v_{i,q} \\ \alpha_{1,1}\, v_{i,u} \\ \vdots \\ \alpha_{n_{\rm freq},1}\, v_{i,q} \\ \alpha_{n_{\rm freq},1}\, v_{i,u} \end{bmatrix} = \begin{bmatrix} \alpha_{1,1}\, \mathsf{v}_i \\ \vdots \\ \alpha_{n_{\rm freq},1}\, \mathsf{v}_i \end{bmatrix} \tag{A.3}$$

because of the form of the mixing we assumed in Eq. (22). Consequently, using Eq. (A.2),

$$\begin{aligned} \widetilde{A}\widetilde{M}_\beta \lceil \mathsf{v}_i, 0, 0 \rfloor &= \begin{bmatrix} \alpha_{1,1}\, P^\top N^{-1} P\, \mathsf{v}_i \\ \vdots \\ \alpha_{n_{\rm freq},1}\, P^\top N^{-1} P\, \mathsf{v}_i \end{bmatrix} \\ &= \begin{bmatrix} \alpha_{1,1}\, \lambda_i P^\top \operatorname{diag}(N^{-1}) P\, \mathsf{v}_i \\ \vdots \\ \alpha_{n_{\rm freq},1} \lambda_i P^\top \operatorname{diag}(N^{-1}) P\, \mathsf{v}_i \end{bmatrix} = \lambda_i \widetilde{B}\widetilde{M}_\beta \lceil \mathsf{v}_i, 0, 0 \rfloor . \end{aligned} \tag{A.4}$$

Because the matrix $\widetilde{M}_\beta^\top \widetilde{A}\widetilde{M}_\beta$ is assumed to be nonsingular (equivalently, because $\widetilde{M}_\beta$ is of full column rank), we can multiply this equation from the left by $\widetilde{M}_\beta^\top$ showing that $(\lambda_i, \lceil \mathsf{v}_i, 0, 0 \rfloor)$ is the eigenpair of the matrix $(\widetilde{M}_\beta^\top \widetilde{B}\widetilde{M}_\beta)^{-1} \widetilde{M}_\beta^\top \widetilde{A}\widetilde{M}_\beta$. We can proceed analogously for the vectors $\lceil 0, \mathsf{v}_i, 0 \rfloor$ and $\lceil 0, 0, \mathsf{v}_i \rfloor$, with replacing in Eq. (A.3) $\alpha_{f,1}$ by $\alpha_{f,2}$ and $\alpha_{f,3}$, respectively.

There are $2\, n_{\rm pix}$ eigenpairs for $(P^\top \operatorname{diag}(N^{-1})P)^{-1}(P^\top N^{-1} P)$. As we showed above, each of them generates three eigenpairs (with the same eigenvalue) of $(\widetilde{M}_\beta^t \widetilde{B}\widetilde{M}_\beta)^{-1} \widetilde{M}_\beta^\top \widetilde{A}\widetilde{M}_\beta$. This gives together $6\, n_{\rm pix}$ eigenpairs, in other words, we have described the full spectrum of the preconditioned system matrix in Eq. (20).

Finally, we remark that all the eigenpairs of the preconditioned matrix in the simplified setting are independent of the parameters $\beta$. In this case, we suggest using a specialized eigensolver (e.g., ARPACK, Lehoucq et al. 1998) to compute the eigenpairs from Eq. (A.2), build the triplets of eigenvectors $\lceil \mathsf{v}_i, 0, 0 \rfloor$, $\lceil 0, \mathsf{v}_i, 0 \rfloor$, and $\lceil 0, 0, \mathsf{v}_i \rfloor$, and then use the deflated PCG with these vectors.

Figure A.1 is the same as Fig. 5, but for the simplified setting. Here two triplets of the eigenvectors are constructed following the procedure described above.

## Appendix B: Ingredients of the proposed procedure

We present in this section two ingredients of the proposed procedure in more detail. Namely, we discuss approaches for estimating the eigenpairs from the computed basis of the Krylov subspace and approaches for combining the deflation of the approximate eigenvectors with another preconditioner. To facilitate presentation, we simplify the notation in this section.

### *B.1. Approximating the eigenvalues using Krylov subspace methods*

We present first the Rayleigh–Ritz approximation, which is used in the Arnoldi and Lanczos algorithms to approximate the eigenvalues of a general nonsingular or, respectively, a hermitian matrix. Then, we recall the Arnoldi and Lanczos algorithms, and finally, we briefly comment on their restarted variants.

The methods discussed below do not represent an exhaustive overview of methods for approximating several eigenvalues and the associated eigenvectors. The omitted methods include the Jacobi–Davidson method (Sleijpen & Van der Vorst 2000), for example, which proved to be particularly efficient for approximating the inner part of the spectrum. For a survey of the methods and a list of references, see Sorensen (2002), for instance.

#### B.1.1. Ritz value and harmonic Ritz value approximations

For a subspace $\mathcal{S} \subset \mathbb{C}^n$, we call $y \in \mathcal{S}$ a Ritz vector of $A$ with Ritz value $\theta$ if

$$Ay - \theta y \perp \mathcal{S}. \tag{B.1}$$

When a (computed) basis $V_j$ of $\mathcal{S}$ is used and $y = V_j w$ is set, the above relation is equivalent to solving

$$V_j^\top A V_j w = \theta V_j^\top V_j w. \tag{B.2}$$

Ritz values are known to approximate the extreme eigenvalues of $A$ well. When an approximation to the interior eigenvalues is required, it can be preferable to compute the harmonic Ritz values. The term harmonic Ritz values was introduced in Paige et al. (1995), where references to previous works using this approximation can be found. Following Parks et al. (2006), we define harmonic Ritz values as the Ritz values of $A^{-1}$ with respect to the space $A\mathcal{S}$,

$$\widetilde{y} \in A\mathcal{S}, \quad A^{-1}\widetilde{y} - \widetilde{\mu}\widetilde{y} \perp A\mathcal{S}. \tag{B.3}$$

We call $\widetilde{\theta} \equiv 1/\widetilde{\mu}$ a harmonic Ritz value and $\widetilde{y}$ a harmonic Ritz vector. When $V_j$ is a basis of $\mathcal{S}$ and $\widetilde{y} = V_j\widetilde{w}$, this relation can be represented as

$$V_j^\top A^\top V_j \widetilde{w} = \widetilde{\mu} V_j^\top A^\top A V_j \widetilde{w} \iff V_j^\top A^\top A V_j \widetilde{w} = \widetilde{\theta} V_j^\top A^\top V_j \widetilde{w}. \tag{B.4}$$

For the properties of the harmonic Ritz value approximations and the relationship with the iteration polynomial in MINRES method, see Paige et al. (1995).

*Remark 1.* There are various presentations and definitions in the literature of the harmonic (Rayleigh–)Ritz procedure; it is often introduced to approximate eigenvalues close to a target $\tau \in \mathbb{C}$. For example, Wu (2017) prescribes the procedure by

$$\widetilde{y} \in \mathcal{S}, \quad A\widetilde{y} - \widetilde{\theta}\widetilde{y} \perp (A - \tau I)\mathcal{S}, \tag{B.5}$$

where $I$ is the identity matrix. With $\widetilde{y} = V_j\widetilde{w}$, this corresponds to the generalized eigenvalue problem

$$V_j^\top (A - \tau I)^\top (A - \tau I) V_j \widetilde{w} = (\widetilde{\theta} - \tau)(V_j^\top (A - \tau I)^\top V_j)\widetilde{w}, \tag{B.6}$$

which becomes for $\tau = 0$ exactly the right-hand side equality in Eq. (B.4).

We note that harmonic Ritz approximation is also often used in the Krylov subspace recycling methods to approximate the smallest (in magnitude) eigenvalues and the associated eigenvectors.

Finally, we comment on the (harmonic) Ritz approximation in the case when we wish to compute the eigenvalues of the matrix $A$ preconditioned from the left by $M$. In the general case, when only $A$ and $M$ are assumed to be nonsingular, the Ritz and harmonic Ritz approximation are applied as above by just replacing in the formulas $A$ by $M^{-1}A$. When the matrix $A$ is hermitian and the preconditioner $M$ is SPD, there is also another option. First, we note that the matrix $M^{-1}A$ is not hermitian, but is self-adjoint with respect to the inner product induced by $M$, that is,

$$(v, M^{-1}Aw)_M = (M^{-1}Av, w)_M, \quad \forall v, w, \tag{B.7}$$

where $(v, w)_M \equiv v^\top M w$. This allows in the definition of Ritz and harmonic Ritz approximation replacing $A$ by $M^{-1}A$ and the standard inner product by the inner product induced by the matrix $M$, giving

$$y \in \mathcal{S}, \quad M^{-1}Ay - \theta y \perp_M \mathcal{S} \tag{B.8}$$

or

$$\widetilde{y} \in M^{-1}A\mathcal{S}, \quad (M^{-1}A)^{-1}\widetilde{y} - \widetilde{\mu}\widetilde{y} \perp_M M^{-1}A\mathcal{S} \tag{B.9}$$

respectively. The corresponding algebraic problems with $y = V_j w$ and $\widetilde{y} = V_j\widetilde{w}$ are

$$V_j^\top A V_j w = \theta V_j^\top M V_j w, \tag{B.10}$$

and

$$V_j^\top A^\top M^{-1} A V_j \widetilde{w} = (1/\widetilde{\mu}) V_j^\top A^\top V_j \widetilde{w}, \tag{B.11}$$

respectively. The problems above involve hermitian matrices only.

#### B.1.2. Arnoldi and Lanczos methods

Arnoldi and Lanczos algorithms for approximating the eigenvalues of a general nonsingular or a hermitian matrix are based on a Ritz approximation with setting $\mathcal{S} = \mathcal{K}_j(A, v_1) = \mathrm{span}(v_1, Av_1, \ldots, A^{j-1}v_1)$, the $j$th Krylov subspace. The methods compute an orthogonal basis $V_j$ of $\mathcal{S}$ such that

$$AV_j = V_j T_j + \beta v_{j+1} e_j^\top, \tag{B.12}$$

where $e_j$ is the last column vector of the identity matrix (of size $j$) and $V_j^\top V_j = I$, $V_j^\top v_{j+1} = 0$. Consequently, the eigenvalue problem in Eq. (B.2) corresponding to the Ritz approximation reads

$$T_j w = \theta w. \tag{B.13}$$

The matrix $T_j$ is available during the iterations. The standard use of the Arnoldi and Lanczos method for eigenvalue approximation consists of solving the above problem and setting the pairs $(\theta, V_j w)$ as the computed approximations.

The Ritz approximation can be replaced by the harmonic Ritz approximation. Then, the matrices in Eq. (B.4) become

$$V_j^\top A^\top A V_j = T_j^\top T_j + \beta^2 e_j e_j^\top, \quad V_j^\top A^\top V_j = T_j^\top. \tag{B.14}$$

*Remark 2.* The Lanczos algorithm is a variant of the Arnoldi algorithm for a hermitian $A$. The matrix $T_j = V_j^\top A V_j$, which is in the Arnoldi method upper Hessenberg, is then also hermitian. Consequently, it is tridiagonal, which means that in each step of the Lanczos method, we orthogonalize the new vector only against the two previous vectors. This ensures that the computational cost of each iteration is fixed, and only when the eigenvalues are to be approximated, storing three vectors $v_{j-1}$, $v_j$ and $v_{j+1}$ is sufficient instead of handling the full matrix $V_j$. The assumption on exact arithmetic is crucial here, however. In finite precision computations, the global orthogonality is typically quickly lost, which can cause several stability issues.

As noted above, an orthonormal basis $V_j$ of $\mathcal{S}$ is advantageous for the Ritz approximation. For the harmonic Ritz approximation applied to an SPD matrix $A$, an $A$-orthonormal basis can instead be constructed, which ensures that the matrix $V_j^\top A^\top V_j = V_j^\top A V_j$ on the right-hand side of Eq. (B.4) is equal to the identity. An $A$-orthonormal basis of a Krylov subspace can be constructed within the iterations of conjugate gradient method using the search direction vectors.

The Arnoldi method can also be applied to the preconditioned matrix $M^{-1}A$ to compute an orthonormal basis $V_j$ of the associated Krylov subspace $\mathcal{K}_j(M^{-1}A, M^{-1}v_1)$, giving

$$M^{-1}AV_j = V_jT_j + \beta v_{j+1}e_j^\top, \quad V_j^\top V_j = I, \quad V_j^\top v_{j+1} = 0. \tag{B.15}$$

For a hermitian $A$ and an SPD preconditioner $M$, we can apply the Lanczos method following the comment made above, using the matrix $M^{-1}A$ and the inner product $(\cdot,\cdot)_M$ induced by $M$ instead of the standard euclidean $(\cdot,\cdot)$, giving

$$M^{-1}AV_j = V_jT_j + \beta v_{j+1}e_j^\top, \quad V_j^\top MV_j = I, \quad V_j^\top Mv_{j+1} = 0. \tag{B.16}$$

The computed basis $V_j$ is therefore $M$-orthonormal.

#### B.1.3. Restarted variants

The number of iterations necessary to converge is not a priori known in Arnoldi and Lanczos algorithms, and it can in general be very high. High iteration counts require a large memory to store the basis vectors, and when a full reorthogonalization is used, a high computational effort because of the growing cost of the reorthogonalization in each step. The idea behind implicitly restarted variants is to limit the dimension of the search space $\mathcal{S}$. This means that the iterations are stopped after a (fixed) number of steps, the dimension of the search space is reduced while maintaining its (Krylov) structure, and the Arnoldi/Lanczos iterations are resumed.

Several restarted variants are described in the literature (a detailed description is beyond the scope of this paper, however): the implicitly restarted Arnoldi (IRA, Sorensen 1992), the implicitly restarted Lanczos (IRL, Calvetti et al. 1994), or the Krylov–Schur method (Stewart 2001/02; Wu & Simon 2000).

The estimation of the spectrum of $A$ is possible within the GMRES, MINRES, and CG iterations (applied to solve a system with $A$) because they are based on Arnoldi (GMRES and MINRES) or Lanczos (CG) algorithms. In contrast, a combination of restarted variants with solving a linear algebraic system is, to the best of our knowledge, not described in the literature.

### B.2. Deflation and two-level preconditioners

In this section we first discuss a deflation preconditioner for Krylov subspace methods that can be regarded as eliminating the effect of several (given) vectors from the operator or, equivalently, augmenting by these vectors the space in which we search for an approximation. Then we describe a combination of the deflation preconditioner with another preconditioner that is commonly used in practice.

The Krylov subspace methods (in particular CG, Hestenes 1952, and GMRES, Saad & Schultz 1986) are well-known for their minimization (optimal) properties over the consecutively built Krylov subspace,

$$\mathcal{K}_j(A, v) = \mathrm{span}\{v, Av, A^2v, \ldots, A^{j-1}v\}. \tag{B.17}$$

A question then arises: given some other subspace $\mathcal{U}$, can we modify the methods such that they have the same optimal properties over the union of $\mathcal{K}_j(A, v)$ and $\mathcal{U}$ (which is often called an augmented Krylov subspace)? The answer is positive and the implementation differs according to the method: it is straightforward for GMRES and requires more attention for CG. Hereafter, we denote by $I$ the identity matrix and by $Z$ the basis of $\mathcal{U}$.

The deflation in GMRES method is often (see, e.g., GCROT by Morgan 1995) considered as a remedy to overcome the difficulties caused by restarts: for computational and memory restrictions, only a fixed number of GMRES iterations is typically performed, giving an approximation that is then used as the initial vector for a new GMRES run. In GCROT, several vectors are saved and used to augment the Krylov subspace built after the restart. The GMRES method with the deflation was used to solve a sequence of linear algebraic systems in Parks et al. (2006), for example.

The augmentation of the Krylov subspace in CG is more delicate because the original CG method can only be applied to an SPD matrix. The first such algorithm was proposed in Nicolaides (1987) and Dostál (1988). We note that it includes the construction of the conjugate projector

$$P_{\mathrm{c.proj.}} = Z(Z^\top AZ)^{-1}Z^\top A, \tag{B.18}$$

and in each iteration, the computation of the preconditioned search direction $q_i = (I - P_{\mathrm{c.proj.}})\, p_i$ and of the vector $Aq_i$. The latter can be avoided at the price of storing $Z$ *and* $AZ$ and performing additional multiplication with $AZ$. In both variants, the cost of a single iteration is higher than the cost of one standard CG iteration.

The combination of a preconditioner with a deflation is widely studied in the literature and therefore we present this only briefly; more details and an extensive list of references can be found in the review paper by Tang et al. (2009), for instance. The preconditioner stemming from the combination of a (typically relatively simple) traditional preconditioner with the deflation is called a two-level preconditioner. As shown in Tang et al. (2009), this shows an analogy with multilevel (multigrid) and domain decomposition methods. While the traditional preconditioner aims at removing the effect of the largest (in magnitude) eigenvalues, the deflation (projection-type preconditioner) is intended to remove the effect of the smallest eigenvalues. Common choices for the traditional preconditioner are block Jacobi, (restricted) additive Schwarz method, and incomplete LU or Cholesky factorizations. In many applications, two-level preconditioners proved to be efficient in the CMB data analysis (see, e.g., Grigori et al. 2012; Szydlarski et al. 2014).

We now present the combination of the traditional and projection-type (deflation) preconditioners following the discussion and notation of Tang et al. (2009). Hereafter, we assume

that the system matrix $A$ and the traditional preconditioner $M$ are SPD. We note that some of the preconditioners $\mathcal{P}_\square$ mentioned below are not symmetric. However, their properties allow us to use them (with possible modification of the initial vector) as left preconditioners in PCG; see Tang et al. (2009) for details.

Let the deflation space span the columns of the matrix $Z$. We denote

$$P \equiv I - AQ, \quad Q \equiv Z(Z^\top AZ)^{-1}Z^\top. \tag{B.19}$$

Two-level preconditioners based on the deflation are given as

$$\mathcal{P}_{\mathrm{DEF1}} \equiv M^{-1}P, \quad \mathcal{P}_{\mathrm{DEF2}} \equiv P^\top M^{-1}. \tag{B.20}$$

Other preconditioners can be determined using the additive combination of two (SPD) preconditioners $C_1, C_2$ as

$$\mathcal{P}_{\mathrm{add}} \equiv C_1 + C_2, \tag{B.21}$$

or, using the multiplicative combination of the preconditioners, as

$$\mathcal{P}_{\mathrm{mult}} \equiv C_1 + C_2 - C_2AC_1. \tag{B.22}$$

Three variants of two-level preconditioners are derived by choosing aan dditive or multiplicative combination and setting $C_1 = M^{-1}$, $C_2 = Q$, or $C_1 = Q$, $C_2 = M^{-1}$. Other preconditioners can be derived using the multiplicative combination of three SPD matrices (see Tang et al. 2009).

The variants of the two-level preconditioner mentioned above differ in the implementation cost and also in the numerical stability; see Tang et al. (2009). The variant $\mathcal{P}_{\mathrm{DEF1}}$, which is often used in the procedures for solving the sequences of linear systems (see, e.g., Saad et al. 2000), was found to be cheap but less robust, especially with respect to the accuracy of solving the coarse problem with the matrix $Z^\top AZ$ and with respect to the demanded accuracy. The conclusion drawn in Tang et al. (2009) is that "A-DEF2 seems to be the best and most robust method, considering the theory, numerical experiments, and the computational cost". Therefore the preconditioner $\mathcal{P}_{\mathrm{A-DEF2}}$,

$$\mathcal{P}_{\mathrm{A-DEF2}} \equiv M^{-1} + Q - QAM^{-1} = P^\top M^{-1} + Q, \tag{B.23}$$

is of interest, in particular in the cases where the dimension of the deflation space (equal to the number of columns in $Z$) is high and/or the matrix $M^{-1}A$ is ill-conditioned.

As noted in Saad et al. (2000), the gradual loss of orthogonality of the computed residuals with respect to the columns of $Z$ can cause stagnation, divergence, or erratic behaviour of errors within the iterations (see also the comment in Tang et al. 2009). The suggested remedy in this case consists of reorthogonalizing the computed residuals as

$$r_j := Wr_j, \quad W \equiv I - Z(Z^\top Z)^{-1}Z^\top. \tag{B.24}$$

However, no such instabilities were observed in our experiments, and the results depicted throughout the paper are for the standard (non-reorthogonalized) variant.

## Appendix C: Full algorithm

In this section we provide the pseudo-codes for the deflated PCG (Algorithm 1), the subspace recycling (Algorithm 2), and for the full procedure (Algorithm 3) proposed in this paper and used in the numerical experiments in Sect. 4.

**Algorithm 1** deflated PCG (variant "def1")

**function** DEFLPCG($\mathbb{A}$, $\mathbb{B}$, $\mathbb{b}$, $\mathbb{s}^{(0)}$, $Z$, $\dim_{\mathrm{p}}$, $j_{\max}$)
  $Q = Z(Z^\top \mathbb{A}Z)^{-1}Z^\top$ ▷ in practice, we save $\mathbb{A}Z$ to use later
  $\mathbb{r}^{(0)} = (I - \mathbb{A}Q)(\mathbb{b} - \mathbb{A}\mathbb{s}^{(0)})$ ▷ with saved $\mathbb{A}Z$, $Q\mathbb{A}$ and $\mathbb{A}Q$ can be computed without applying $\mathbb{A}$
  $\mathbb{p}^{(0)} = \mathbb{r}^{(0)}$
  $\widetilde{\mathbb{r}}^{(0)} = \mathbb{B}^{-1}\mathbb{r}^{(0)}$
  **for** $j = 0, \ldots, j_{\max}$ **do**
    $\mathbb{w}^{(j)} = (I - \mathbb{A}Q)(\mathbb{A}\mathbb{p}^{(j)})$
    **if** $j \le \dim_{\mathrm{p}}$ **then**
      **save** $(I - Q\mathbb{A})\mathbb{p}^{(j)}$ into $\widetilde{Z}$ ▷ in practice, we also save $\mathbb{w}^{(j)}$ to avoid computing $\mathbb{A}\widetilde{Z}$ later
    **end if**
    $\gamma^{(j)} = (\widetilde{\mathbb{r}}^{(j)}, \mathbb{r}^{(j)})/(\mathbb{p}^{(j)}, \mathbb{w}^{(j)})$
    $\mathbb{s}^{(j+1)} = \mathbb{s}^{(j)} + \gamma^{(j)}\mathbb{p}^{(j)}$
    $\mathbb{r}^{(j+1)} = \mathbb{r}^{(j)} - \gamma^{(j)}\mathbb{w}^{(j)}$
    **check** the stopping criterion
    $\widetilde{\mathbb{r}}^{(j+1)} = \mathbb{B}^{-1}\mathbb{r}^{(j+1)}$
    $\delta^{(j)} = (\widetilde{\mathbb{r}}^{(j+1)}, \mathbb{r}^{(j+1)})/(\widetilde{\mathbb{r}}^{(j)}, \mathbb{r}^{(j)})$
    $\mathbb{p}^{(j+1)} = \widetilde{\mathbb{r}}^{(j+1)} + \delta^{(j)}\mathbb{p}^{(j)}$
  **end for**
  $\mathbb{s}^{(\mathrm{final})} := Q\mathbb{b} + (I - Q\mathbb{A})\mathbb{s}^{(j)}$
  **return** $\mathbb{s}^{(\mathrm{final})}$; $\widetilde{Z}$
  ▷ to be efficient, we also return $\mathbb{A}Z$ and the vectors $\{\mathbb{w}^{(j)}\}$
**end function**

**Algorithm 2** subspace recycling (variant "ritz")

**function** SUBSPREC($\mathbb{A}$, $\mathbb{B}$, $U$, $k$)
  $F = U^\top \mathbb{B}U$
  $G = U^\top \mathbb{A}U$ ▷ in practice, we reuse $\mathbb{A}U$ saved during the deflated PCG
  **solve** the generalized eigenvalue problem $GY = \mathrm{diag}(\lambda_i)FY$
  **take** $k$ smallest $\lambda_i$ and the respective columns of $Y$, $Y_k$
  **return** $Z = UY_k$
**end function**

**Algorithm 3** full algorithm of the procedure

**Require:** $\beta_0$, $\mathbb{s}^{(0)}_{\beta_0}$
**Require:** $k$, $\dim_p$
  set $Z := []$
  **for** $i = 0, \ldots, i_{\max}$ **do**
    **assembly** the system matrix $\mathbb{A}$, right-hand side $\mathbb{b}$, and the preconditioner $\mathbb{B}$ corresponding to $\beta_i$
    **if** $i > 0$ **then** $\mathbb{s}^{(0)}_{\beta_i} = (\widetilde{M}_{\beta_i})^{\dagger}\mathbb{s}^{(0)}_{\beta_i}$
    DEFLPCG($\mathbb{A}$, $\mathbb{B}$, $\mathbb{b}$, $\mathbb{s}^{(0)}_{\beta_i}$, $Z$, $\dim_p$, $j_{\max}$) $\longrightarrow$ ($\mathbb{s}^{(\text{final})}_{\beta_i}$, $\widetilde{Z}$)
    **check** the stopping criterion for $\beta_i$, **exit** if $i = i_{\max}$
    $\mathbb{s}^{(0)}_{\beta_{i+1}} = \widetilde{M}_{\beta_i}\mathbb{s}^{(\text{final})}_{\beta_i}$
    (determine $\beta_{i+1}$) ▷ considered here as a black box
    SUBSPREC($\mathbb{A}$, $\mathbb{B}$, $U = [Z, \widetilde{Z}]$, $k$) $\longrightarrow Z$
  **end for**

## Appendix D: Results for an alternative sequence of mixing parameters from a Monte Carlo sampling

In this section we provide results for a sequence of spectral parameters generated by a Monte Carlo sampling algorithm. In realistic circumstances, these sequences may contain up to many thousand samples, but for computational efficiency, we here restrict ourselves to a subsequence made of merely 30 elements. We use them in order to demonstrate the performance of the proposed approach on a sequence with realistic properties but sufficiently different than those of the sequences encountered in the likelihood maximization process. We emphasize that it is not yet our purpose here to validate the method on the actual application, and more work is needed to achieve this goal, see Sect. 5.

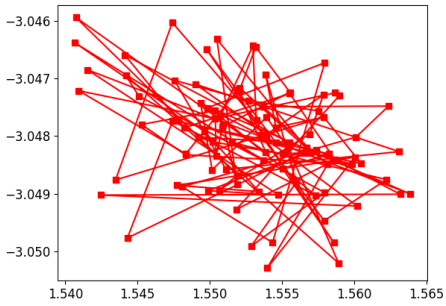


**Fig. D.1.** Plot of a sequence of the spectral parameters $\beta_i = [\beta_{i,s}, \beta_{i,d}]$ drawn through a Monte Carlo sampling technique and used as an alternative test case in the numerical experiments described in Appendix D.

**Table D.1.** Number of matrix-vector products (MatVecs) for different techniques as in Fig. D.2.

| | #MatVecs | | |
|---|---|---|---|
| | Iteration | Deflation | Total |
| $\mathbb{s}^{(0)}_{\beta_{i+1}}$ as in (33) | 4010 | 0 | 4010 |
| $\mathbb{s}^{(0)}_{\beta_{i+1}}$ as in (37) | 1768 | 0 | 1768 |
| Recycle + $\mathbb{s}^{(0)}_{\beta_{i+1}}$ as in (37) | 1228 | 290 | 1518 |

The sequence is depicted in Fig. D.1. It was produced using the publicly available software FGBUSTER[4] and indeed shows qualitatively a very different behavior than that of our standard case displayed in Fig. 1.

In Fig. D.2 and in Table D.1, we compare the results obtained in this case by applying the various techniques discussed and proposed in this work.

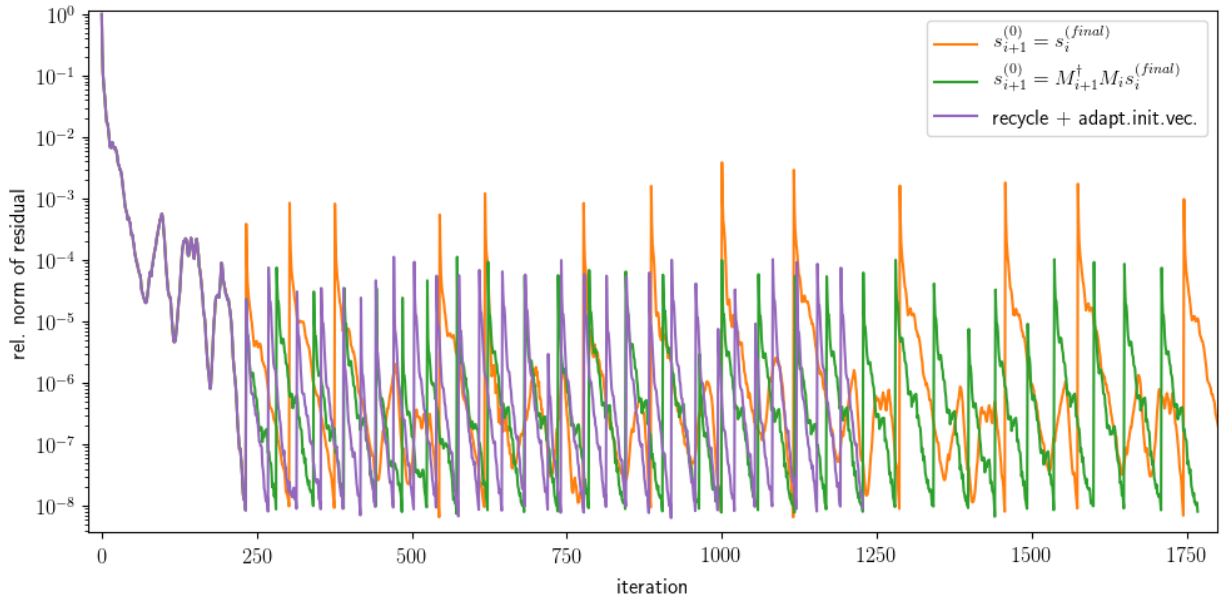


**Fig. D.2.** Comparison of the PCG with different choices of initial guess (as in Eqs. (33) and (37)) and the PCG with the subspace recycling (together with the choice of the initial guess as in Eq. (37)). For the recycling, we consider $k = 10$ eigenvectors approximated using $\dim_p = 100$. The convergence for the whole sequence when the initial guess is as in Eq. (33) (the yellow line) requires 4010 iterations.

[4] FGBUSTER: https://github.com/fgbuster