Signal Processing 108 (2015) 430–439

Contents lists available at ScienceDirect

Signal Processing

journal homepage: www.elsevier.com/locate/sigpro

Drift removal by means of alternating least squares with application to Herschel data ☆

Lorenzo Piazzo [a,*], Pasquale Panuzzo [b], Michele Pestalozzi [c]

[a] DIET Department, Sapienza University of Rome, V. Eudossiana 18, 00184 Rome, Italy
[b] GEPI Laboratory, CNRS – University Paris Diderot, Meudon, France
[c] IAPS Institute, National Institute of Astrophysics, Rome, Italy

ARTICLE INFO

Article history:
Received 13 May 2014
Received in revised form
18 August 2014
Accepted 24 September 2014
Available online 16 October 2014

Keywords:
Least squares
Image formation
Infrared imaging

ABSTRACT

We consider the problem of reconstructing an image observed with a linear, noisy instrument, the output of which is affected by a drift too, causing a slowly varying deviation of the readouts from the baseline level. Since the joint estimation of the image and the drift, which is the optimal approach, is demanding for large data, we consider an alternative approach, where we remove the drift and the noise in two separate steps. In particular, we remove the drift by means of Least Squares (LS) and the noise by means of Generalised Least Squares (GLS). Moreover, we introduce an efficient drift removal algorithm, based on Alternating Least Squares (ALS), and carry out an analysis which proves convergence and gives geometrical insight. Finally, we apply the approach to the Herschel satellite data, discussing the performance and showing that nearly optimal results are achieved.

1. Introduction

Consider a linear instrument observing an image and producing a sequence of readouts. The image formation problem is that of reconstructing the image from the readouts. This problem is complicated by the impairments affecting the output, the first of which is the noise introduced by the instrument. A second impairment often arising in practice is a drift affecting the detector, causing a time varying deviation of the readouts from the baseline level. Such a problem is encountered, for example, in medical imaging [1,2], in electron microscopy [3] and in astronomical imaging [4,5]. Typically, the drift is slowly varying and can be modelled with smooth curves, depending on a few parameters. However the drift may be large, possibly larger than the useful signal, and must therefore be accounted for in order to obtain quality images.

The optimal approach to the image formation is that of jointly estimating the image and the drift, e.g. [2,4]. However the joint estimation is computationally demanding. Another option, e.g. [5], is to use a two steps approach: in the first step an estimate of the drift is produced and subtracted from the raw data, to obtain an updated data set which is, ideally, drift free; in the second step the updated data set is fed to a proper noise removal algorithm. This approach is less demanding and is studied in this paper where, specifically, we propose to use a Joint Least Square (JLS) method to remove the drift and a Generalised Least Squares (GLS) method to remove the noise. As we will see, the JLS preserves the useful signal and, under proper assumptions, completely removes the drift. As a result our approach produces near optimal images. Moreover, the proposed approach is more practical than the one of [5], where the drift estimation is carried out by means of an interactive method, requiring human intervention.

☆ This work was partially funded by the Sapienza University Research Projects no. C26A11825Y/2011 and C26A1324B9/2013.

* Corresponding author. Tel.: +39 06 44585480; fax: +39 06 4873300.
E-mail address: lorenzo.piazzo@uniroma1.it (L. Piazzo).

http://dx.doi.org/10.1016/j.sigpro.2014.09.039
0165-1684/© 2014 Elsevier B.V. All rights reserved.

Even if the two steps approach simplifies the image formation, the drift estimation by means of the JLS remains challenging when the dimension of the data set is large. Therefore, we introduce an iterative drift removal algorithm, which, under proper assumptions, is capable of significantly reduce the computational burden. The algorithm is based on Alternating Least Squares (ALS) which is a well known optimisation method, e.g. [6], strictly linked with Alternating Projection (AP) [7]. The main contribution of the paper is an original analysis proving that the ALS algorithm converges to the JLS solution. The analysis also gives geometrical insight, showing that the drift removal amounts at a data projection onto an appropriate subspace.

As an additional contribution, we discuss the application of the approach to the data of the Herschel satellite, which is a space telescope launched by the European Space Agency (ESA) in year 2009 [8]. Therefore, we specialise the approach to Herschel data, modelling the drift as a polynomial, and discuss the performance, using both theoretical analysis and numerical examples, showing that the two steps approach yields essentially optimal results for such data and that the ALS is an efficient implementation of the JLS. Indeed ALS algorithms are exploited in several Herschel data reduction software [9–11].

The paper is organised as follows. In Section 2 we introduce the data model and discuss the image formation problem. In Section 3 the two steps approach and the ALS algorithm are presented. The ALS algorithm is analysed in Section 4 and the application of the approach to Herschel data is discussed in Section 5. In Section 6 we investigate the performance, by presenting results obtained from simulated and true Herschel data. The conclusions are given in Section 7.

Notation: In the paper we use lowercase letters to denote vectors and uppercase letters to denote matrices. We use a superscript T to denote matrix or vector transposition, e.g. A^T. We use $|v|^2 = v^T v$ to denote the squared magnitude of the column vector v and say that two vectors v and w are orthogonal if $v^T w = 0$. We use $\mathbf{E}[.]$ to denote expectation. Given a vector v with V elements v_i for $i = 1, \ldots, V$, its average is $g = \frac{1}{V}\sum_{i=1}^{V} v_i$ and we denote by $\mathbf{v}$ the vector obtained from v by subtracting the average element-wise, i.e. $\mathbf{v}_i = v_i - g$ for $i = 1, \ldots, V$.

2. Preliminaries

Consider an image of M pixels, represented by an $M \times 1$ vector m. The image is observed with an instrument producing $N \gg M$ readouts, represented by an $N \times 1$ *data* vector d. Assuming a linear, noisy instrument, the data vector is

$$d = Pm + n = s + n$$

where P is an $N \times M$, full-rank matrix, which depends on the instrument and on the observation protocol, n is a zero-mean, random *noise* vector and we introduced the *signal* vector $s = Pm$ representing the ideal, noiseless output. Given this set up, the image formation (or noise removal) problem is that of producing an estimate of m, denoted by $\overline{m}$, knowing d, P and the statistics of n. This is a classical problem having several established solutions. A simple and effective one is based on Least Squares (LS) and is summarised in the following.

Since for a generic image x the ideal output is Px, we can minimise $|d - Px|^2$ for varying x and obtain the image producing the output closest to the actual data vector. This image is the LS estimate of m and is given by [12]

$$\overline{m} = (P^T P)^{-1} P^T d. \tag{1}$$

LS estimation is an effective technique when the noise is white. Indeed, in white Gaussian noise it yields the Maximum Likelihood (ML) estimate. On the contrary, when the noise is correlated, LS performs poorly. In this case, by denoting by $C = E\{nn^T\}$ the noise covariance matrix, we can use a Generalised LS approach (GLS) [13] and minimise $|C^{-\frac{1}{2}}(d - Px)|^2$ for varying x. The solution is the GLS estimate given by

$$\overline{m} = (P^T C^{-1} P)^{-1} P^T C^{-1} d \tag{2}$$

which is the ML solution if the noise is Gaussian.

We now assume that the instrument introduces a drift too. The drift is represented with an $N \times 1$ *drift* vector $y = Xa$, where X is a given, full rank, $N \times K$ matrix, termed the drift matrix, and a is a $K \times 1$ coefficients vector, with $K \ll N$. Then the data vector is

$$d = Pm + Xa + n = s + y + n, \tag{3}$$

where X and P are known, full-rank matrices, m and a are deterministic, unknown vectors, and n is a zero-mean random vector. The last equation is a general and linear model, that will be used in this paper and is exploited, for example, in Diffuse Optical Imaging (DOI) [2] and functional Magnetic Resonance Imaging (fMRI) [1]. It is also well suited for space-born infrared data, as we see later. By constructing an appropriate drift matrix, the model can be specialised to several practical drift types, including polynomial, sinusoidal, piece-wise constant and piece-wise linear. We give an example in Section 5.1.

In the presence of the drift, both the LS and GLS estimates cannot be used directly, since the results are biased by the drift. However the approaches can be extended to the joint estimation of the image and coefficients. Indeed, by introducing the $N \times (M+K)$ matrix $A = [P, X]$ and the $(M+K) \times 1$ vector $z = [m^T, a^T]^T$, the data vector can be written as

$$d = Az + n$$

and the problem becomes that of estimating z. Then, by exploiting the LS approach and assuming that A is full rank, we obtain

$$\overline{z} = \begin{bmatrix} \overline{m} \\ \overline{a} \end{bmatrix} = (A^T A)^{-1} A^T d \tag{4}$$

which will be called the Joint LS (JLS) estimate and is the ML solution in white, Gaussian noise. When the noise is correlated, by exploiting the GLS approach, we obtain

$$\overline{z} = \begin{bmatrix} \overline{m} \\ \overline{a} \end{bmatrix} = (A^T C^{-1} A)^{-1} A^T C^{-1} d$$

which will be called the Joint GLS (JGLS) estimate and is the ML estimate in Gaussian noise.

All the estimates introduced thus far require the inversion of a matrix in order to be computed. When the size of the data is large, the inversion cannot be performed numerically. In some cases the inversion can be performed analytically but, in general, the estimates have to be computed by other means. We will see an example in Section 5.4. In any case, the estimates are demanding to compute and difficult to implement and, in particular, the JGLS is not practical for large data. For these reasons in the next section we consider a simpler approach.

3. Drift removal and the ALS algorithm

In the presence of drift and correlated noise, we can solve the image formation problem in two steps. In the first step, called the drift removal, we use the JLS approach in order to produce an estimate of the drift, denoted by $\overline{y}$, and subtract it from the data to obtain an updated data vector $\tilde{d} = d - \overline{y}$. In particular, the estimate is produced by extracting the coefficients $\overline{a}$ from $\overline{z}$ of (4) and by computing

$$\overline{y} = X\overline{a}. \tag{5}$$

In the second step, called the noise removal, we use the GLS approach to estimate the image from the updated data vector, as better discussed in Section 4.4.

The two steps approach replaces the JGLS estimation with JLS and GLS, thereby achieving a drastic reduction in the computational and implementation complexity. Moreover, the drift estimate is carried out without the knowledge of the noise statistics. This is an advantage when the statistics are not known and have to be estimated from the data. In fact the statistics can be estimated after the drift removal, from the updated data vector, which is normally a simpler option than estimating the statistics from the raw data. Concerning the performance, the two steps approach is in general sub-optimum. However, if the drift depends on a few parameters, the noise rejection is good and we expect that the JLS estimate is accurate, even in correlated noise. In turn, if the drift is well removed, the GLS approach will be close to the optimum and, overall, the two steps approach shall yield a solution close to the JGLS one. This fact will be verified by means of simulations in Section 6.1.

In our approach, the updated data vector $\tilde{d}$ is computed using the JLS estimate, which requires the inversion of the $(M+K) \times (M+K)$ matrix A^TA. When $M+K$ is high, the inversion cannot be performed numerically. In this case the estimate has to be obtained by other means and a well known technique will be discussed in Section 5.4. An additional alternative is to use the following algorithm, based on ALS:

```
1. Initialise $d_0 = d$ and $k=0$.
2. Estimate image: $\overline{m} = (P^T P)^{-1} P^T d_k$.
3. Estimate signal: $\sigma = P\overline{m}$.
4. Remove signal: $\eta = d_k - \sigma$.
5. Estimate coefficients: $\alpha = (X^T X)^{-1} X^T \eta$.
6. Estimate drift: $\delta = X\alpha$.
7. Remove drift: $d_{k+1} = d_k - \delta$.
8. If convergence achieved set $\tilde{d} = d_{k+1}$ and stop.
9. Set $k = k+1$ and go to 2.
```

The algorithm performs a sequence of alternating signal and drift LS estimations. Indeed, by comparing with (1), we see that the vector $\overline{m}$ of step 2 is the LS estimate of the image, so that the vector σ of step 3 is the LS estimate of the signal. Similarly, steps 5 and 6 implement the LS drift estimation. A key point is that the drift is estimated from the vector η, from which the signal has been removed, in step 4. Naturally, the vector σ is a biased estimate of the signal, because of the presence of the drift. Then, also the drift estimate δ is biased. But removing it from the data, in step 7, we obtain an improved data vector, d_{k+1}, where the drift is partially removed. The process can be repeated to improve the drift removal at each iteration. Note that the algorithm produces an image estimate too, namely the vector $\overline{m}$ of step 2.

In the following section we present an analysis of the algorithm which demonstrates convergence and characterizes the performance. In particular, we show that the updated data vector and the image estimate produced by the ALS algorithm are identical to those produced by the JLS. Therefore, the ALS algorithm is actually an iterative way of computing the JLS estimate. However, the ALS only requires the inversion of P^TP and X^TX, which are $M \times M$ and $K \times K$, respectively. This is a significant simplification with respect to the inversion of A^TA. Moreover, there are practical cases where the matrices P and X are structured and the inversions can be performed analytically. In this case the ALS is a quite efficient way of implementing the JLS, as we will verify in Section 6.4.

4. Analysis

In the analysis, we denote by $\Re^N$ the vector space of the $N \times 1$ real column vectors and use basic facts from linear algebra, which can be found in any textbook, e.g. [12].

4.1. Formalisation

In order to analyse the algorithm and to prove the convergence we need an expression linking the updated data vector to the original one. As a preliminary remark, observing (3), we note that the signal vector s belongs to the subspace of $\Re^N$ spanned[1] by the matrix P, that is termed the signal subspace and is denoted by Σ_P. Similarly, the drift vector y belongs to the subspace spanned by the drift matrix X, termed the drift subspace and denoted by Σ_X. Next note that the vector σ produced in step 3 can be written as $\sigma = P\overline{m} = P(P^T P)^{-1} P^T d_k = \Pi_P d_k$ where $\Pi_P = P(P^T P)^{-1} P^T$ is the orthogonal projector onto the signal subspace [12]. Then we have

$$\eta = d_k - \sigma = (I - \Pi_P) d_k = \Pi_P^{\perp} d_k$$

where we introduced $\Pi_P^{\perp} = (I - \Pi_P)$ which is the projector onto the orthogonal complement of the signal subspace. Similarly, the vector δ produced in step 6 is $\delta = X\alpha = X(X^T X)^{-1} X^T \eta = \Pi_X \eta$, where $\Pi_X = X(X^T X)^{-1} X^T$ is the orthogonal projector onto the drift subspace. Then,

[1] The span of a matrix is the set of vectors that can be obtained as a linear combination of the matrix's columns.

replacing η, we get $\delta = \Pi_X \Pi_P^{\perp} d_k$ so that

$$d_{k+1} = d_k - \Pi_X \Pi_P^{\perp} d_k = (I - \Pi_X \Pi_P^{\perp}) d_k.$$

From the latter equation it is simple to derive an expression for $\tilde{d}$. Namely, since $d_0 = d$, we have $d_k = (I - \Pi_X \Pi_P^{\perp})^k d$, and since d_k tends to the updated data vector when $k \to \infty$, we have[2]

$$\tilde{d} = (I - \Pi_X \Pi_P^{\perp})^{\infty} d = Ld \tag{6}$$

where we introduced the matrix $L = (I - \Pi_X \Pi_P^{\perp})^{\infty}$. The latter equation expresses the action of the ALS on the data vector and is exploited in the next subsection to analyse the algorithm.

4.2. Analysis of ALS

Consider the intersection of the signal and drift subspaces, denoted by $\Sigma_{XP} = \Sigma_X \cap \Sigma_P$, and the orthogonal complement of Σ_{XP} in Σ_X, denoted by Σ_Z and constituted by the vectors of Σ_X that are orthogonal to all the vectors of Σ_{XP}, i.e. $\Sigma_Z = \Sigma_{XP}^{\perp} \cap \Sigma_X$. Note that Σ_X is the direct sum[3] of Σ_{XP} and Σ_Z, which is denoted by

$$\Sigma_X = \Sigma_{XP} \oplus \Sigma_Z. \tag{7}$$

Also note that, as is easy to verify,[4] $\Sigma_P \cap \Sigma_Z = 0$, so we introduce the direct sum of these subspaces, denoted by $\Sigma_D = \Sigma_P \oplus \Sigma_Z$, its orthogonal complement in $\Re^N$, denoted by $\Sigma_D^{\perp}$, and obtain the following decomposition:

$$\Re^N = \Sigma_D \oplus \Sigma_D^{\perp} = \Sigma_P \oplus \Sigma_Z \oplus \Sigma_D^{\perp}. \tag{8}$$

We now discuss the output of the algorithm when the input is a generic vector $w \in \Re^N$. Using (6) the output is $\tilde{w} = Lw$. Based on (8) we decompose w as $w = w_P + w_Z + w_{D^{\perp}}$ where $w_P \in \Sigma_P$, $w_Z \in \Sigma_Z$ and $w_{D^{\perp}} \in \Sigma_D^{\perp}$. In Appendix A we separately analyse the behaviour of the ALS in these subspaces and show that the algorithm converges in the three of them. In particular, using the AP theorem [7], we show that

$$Lv = v \quad \text{for } v \in \Sigma_P \text{ and } v \in \Sigma_D^{\perp} \tag{9}$$

$$Lv = 0 \quad \text{for } v \in \Sigma_Z \tag{10}$$

so that $\tilde{w} = w_P + w_{D^{\perp}}$. The last equation clarifies the functioning of the ALS algorithm. In words, the algorithm removes from the input vector the component falling in Σ_Z. The other two components are not modified. Moreover, (8), (9) and (10) show that the matrix L is the oblique projector onto $\Sigma_P \oplus \Sigma_D^{\perp}$ along Σ_Z.

To gain insight, consider the data vector $d = s + y + n$ and note that the updated data vector $\tilde{d} = Ld$ can be written as $\tilde{d} = Ls + Ly + Ln = \tilde{s} + \tilde{y} + \tilde{n}$. Next, using (7) and (8), we decompose the drift and noise as $y = y_{XP} + y_Z$ and $n = n_P + n_Z + n_{D^{\perp}}$. Finally, by noting that $\Sigma_{XP} \subset \Sigma_P$ and $s \in \Sigma_P$, using (9) and (10) we obtain

$$\tilde{s} = s \quad \tilde{y} = y_{XP} \quad \tilde{n} = n_P + n_{D^{\perp}}.$$

The last equations show that the signal component is preserved by the algorithm; in fact $\tilde{s} = s$. However, the drift is not entirely removed; in fact $\tilde{y} \neq 0$. In particular we see that the drift component falling outside the signal subspace, namely y_Z, is removed while the component falling inside the signal subspace, y_{XP}, is not detected and leaks into the updated vector. The updated noise will be discussed soon.

In order to further characterize the residual drift y_{XP} and the algorithm performance, we need to study Σ_{XP}. This cannot be done in general because it requires to specify X and P, i.e. to consider a specific application. However we note that the matrix P is controlled by the observation protocol; then, by properly selecting the protocol, we can enforce $\Sigma_{XP} = 0$, so that $y_{XP} = 0$ and the drift is entirely removed. We give an example in Section 5.2.

4.3. Analysis of JLS and equivalence with ALS

We start by developing an expression for the JLS estimates. We first consider the simple case when $\Sigma_{XP} = 0$ so that $\Sigma_Z = \Sigma_X$ and A is full-rank. Using block matrix pseudo-inverse formulas [14] the JLS estimate of (4) is

$$\hat{z} = \begin{bmatrix} \hat{m} \\ \hat{a} \end{bmatrix} = \begin{bmatrix} (P^T \Pi_X^{\perp} P)^{-1} P^T \Pi_X^{\perp} \\ (X^T \Pi_P^{\perp} X)^{-1} X^T \Pi_P^{\perp} \end{bmatrix} d.$$

Note that, as will be done in the whole subsection, we used a hat to denote the JLS estimates in order to distinguish them from the corresponding ALS estimates. From the last equation, the JLS image estimate is

$$\hat{m} = (P^T \Pi_X^{\perp} P)^{-1} P^T \Pi_X^{\perp} d.$$

Moreover, using (5), the JLS drift estimate is

$$\hat{y} = X\hat{a} = X(X^T \Pi_P^{\perp} X)^{-1} X^T \Pi_P^{\perp} d = Hd, \tag{11}$$

where we introduced the matrix $H = X(X^T \Pi_P^{\perp} X)^{-1} X^T \Pi_P^{\perp}$. The corresponding JLS updated data vector is $\hat{d} = d - \hat{y} = (I - H)d$.

We now compare the JLS and ALS estimate. To this end, in Appendix B we show that H is the oblique projector onto Σ_Z along $\Sigma_P \oplus \Sigma_D^{\perp}$. Then $(I - H)$ is the projector onto $\Sigma_P \oplus \Sigma_D^{\perp}$ along Σ_Z and, since the projector is unique [12], we have $L = (I - H)$. Therefore the ALS and the JLS produce the same updated data vector and drift estimate. Moreover, the ALS image estimate produced in step 2, at convergence, can be written as

$$\overline{m} = (P^T P)^{-1} P^T L d$$

and in Appendix C we show that $\hat{m} = \overline{m}$. We conclude that the ALS and the JLS are equivalent, in the sense that they yield the same estimates.

We now consider the case when $\Sigma_{XP} \neq 0$. In this case A^{TA} is singular and the JLS solution is not unique in general. To deal with this case we need to revise the drift model. Specifically, by denoting by $\hat{K}$ the dimension of Σ_Z, we

[2] Here and in the following, given a matrix G we use the notation G^{∞} to indicate $\lim_{k \to \infty} G^k$.

[3] The sum of two subspaces Θ and Γ is the subspace of all the vectors that can be decomposed as $t + g$ where $t \in \Theta$ and $g \in \Gamma$. If Θ and Γ are disjoint, i.e. $\Theta \cap \Gamma = 0$, the sum is termed a direct sum, it is denoted by $\Theta \oplus \Gamma$ and the decomposition is unique.

[4] Suppose that $v \in \Sigma_P \cap \Sigma_Z$, i.e. $v \in \Sigma_P$ and $v \in \Sigma_Z$. Since, by construction, $\Sigma_Z \subset \Sigma_X$, we also have that $v \in \Sigma_X$. Then $v \in \Sigma_{XP}$ and $v \in \Sigma_Z$ which is possible only if $v = 0$.

produce a $N \times \hat{K}$ matrix $\hat{X}$ the columns of which are a basis for Σ_Z. Then we rewrite the data model as

$$d = Pm + \hat{X}a + n \tag{12}$$

where a is a $\hat{K} \times 1$ vector. Now the matrix $A = [P, \hat{X}]$ is full rank by construction and we can use (4) to estimate the image and the drift. Note that in this way we implicitly assign any component of d belonging to Σ_{XP} to the signal vector. Therefore if the drift has a component in Σ_{XP} this component will not be removed and will constitute a residual drift. More generally, we can repeat the reasoning carried out for $\Sigma_{XP} = 0$ and prove that, again, the ALS and JLS solutions are identical. The details are skipped for the sake of the brevity.

4.4. Noise and GLS

In order to implement the second step of our approach, namely the GLS image estimate, we need to study the updated noise $\tilde{n}$, which is obtained from n by removing the component falling into Σ_Z. It is not difficult to verify that $\tilde{n}$ is zero-mean.[5] Moreover the covariance matrix of $\tilde{n}$ is given by

$$\tilde{C} = \mathbf{E}[\tilde{n}\tilde{n}^T] = \mathbf{E}[Lnn^T L^T] = L\mathbf{E}[nn^T]L^T = LCL^T.$$

However we cannot compute the GLS estimate by directly replacing d and C in (2) with $\tilde{d}$ and $\tilde{C}$, because $\tilde{C}$ is singular. This happens because the updated noise belongs to a subspace of $\Re^N$, namely the one spanned by the projector L, i.e. $\Sigma_P \oplus \Sigma_D^\perp$. In principle, to compute the GLS estimate, we should remap the problem to this subspace. However this is not practical. A simpler solution is to regularise the covariance matrix, by computing it as

$$\tilde{C} = (\alpha L + \beta I)C(\alpha L + \beta I)^T \tag{13}$$

where $0 < \alpha \ll 1$ and $\beta = 1 - \alpha$. A second problem is that the computation of $\tilde{C}$ may be unfeasible, due to the dimension of the matrices. In this case the original covariance matrix can be used as an approximation. Alternatively the noise statistics can be obtained by other means, e.g. measured.

5. Application to Herschel data

The Herschel satellite hosts two infrared instruments, the Photodetector Array Camera and Spectrometer (PACS) [15] and the Spectral and Photometric Imaging Receiver (SPIRE) [16]. The photometers of PACS and SPIRE consist of two and three arrays of bolometers, respectively. The two PACS arrays operate in the 70 μm (or 100 μm) and 160 μm bands and are divided into eight and two subarrays, respectively. The three SPIRE arrays operate in the 250, 350 and 500 μm bands. The field of view of each array is approximately 45 squared arcsec for PACS ad 2 squared arcmin for SPIRE, but a typical photometric Herschel observation covers a much larger sky area, up to some square degrees wide. To observe the area, the telescope is moved along a set of parallel scan lines, covering the area.

[5] In fact $\mathbf{E}[\tilde{n}] = \mathbf{E}[Ln] = L\mathbf{E}[n] = 0$.

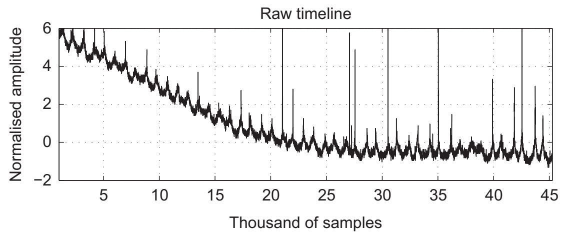

Fig. 1. Timeline of a 70 μm observation. A large drift is observed in the first half of the timeline while the signal stabilises in the second half.

During the scan the bolometers are sampled at regular times, producing a sequence of readouts for each bolometer which is termed a timeline. Redundancy is guaranteed by scanning the area along at least two different directions, so that each bolometer normally produces two or more timelines. The timelines, together with the corresponding pointing information, constitute the observation raw output and are collectively referred as the Time Ordered Data (TOD). A typical 70 μm TOD is constituted by about 4000 timelines, with a total of $N \approx 10^9$ readouts; the image has $M \approx 10^7$ pixels.

The timelines are affected by noise and drift.[6] The noise is typically modelled as a zero-mean, stationary Gaussian process [15,16]. The noise power spectrum is given by the sum of two terms, namely a white noise with flat spectrum $N_w(f) = N_0$ plus a correlated noise with spectrum $N_c(f) = (f_k/f)^\epsilon N_0$ where f_k is called the knee frequency and ϵ the frequency exponent. The correlated noise is also referred as the $1/f$-noise and dominates the spectrum at low frequencies.

The drift is typically modelled as a low order polynomial [17] and may be much larger than the signal, especially for PACS.[7] An example is reported in Fig. 1, showing a 70 μm timeline where a large drift can be seen in the first half. Both a specific drift, different for different timelines, and a common drift, equal for all the timelines of an array or subarray, can be observed [17].

5.1. Data model

We now cast the Herschel data into the model of (3). To this end we need to specify the vector d and the matrices P and X. The data vector d is obtained by stacking all the timelines into a $N \times 1$ column vector. Specifically, if the TOD is composed of N_t timelines each composed of N_r readouts, $N = N_t N_r$.

In order to obtain the matrix P, which is termed the pointing matrix, we assume that the sky is pixellised, i.e. that it is partitioned into a grid of M non-overlapping squares (pixels) where the flux is constant, and is represented by an $M \times 1$ vector m. Next, we assume that the signal component of each readout is equal to the value of

[6] The timelines are affected by other impairments too, like offsets, saturations and glitches. We assume that these impairments have been removed in a prior stage or are negligible.

[7] The drift is mainly due to temperature variations of the focal plane and of the readout electronics. In SPIRE, a large fraction of the drift is compensated using the thermometers output.

the sky pixel where the bolometer was pointed at the sampling time. Then the signal vector can be written as $s=Pm$ where P is an $N \times M$, sparse, binary matrix constructed from the pointing information and such that the element on the k-th row and i-th column is 1 if the k-th readout falls into the i-th sky pixel and is 0 otherwise.

Note that in the last reasoning we made two approximations. First, in practice the readouts do not yield exactly the emission coming from the pointing direction but the emission integrated over a small area centered on the pointing direction and dictated by the telescope Point Spread Function (PSF). As a result, our image estimate will be blurred by the PSF. We could solve this problem by using an appropriate pointing matrix [18], taking into account the PSF, but at the price of a sharp increase in the computational complexity, which is undesirable for Herschel data. Second, the sky is continuous in nature and not pixellised. However our approximation is good when the pixel size is smaller than the instrument PSF. Naturally the pixel size also has to be large enough that several readouts fall into each pixel, so that $N \gg M$ and the redundancy needed for the image formation is guaranteed. Both conditions can be satisfied for Herschel data.

In order to obtain the matrix X, we assume that the drift affecting each timeline is a polynomial of order N_a-1, specified by N_a coefficients.[8] The choice of an appropriate order will be discussed in Section 6.2. Such a drift can be written as $y=Xa$ where a is a $K \times 1$ coefficients vector, obtained by stacking all the $K=N_tN_a$ polynomial coefficients, and X is a $N \times K$ block diagonal matrix, composed by N_t identical blocks and given by

$$X=\begin{pmatrix} \tilde{X} & 0 & 0 & \cdots & 0 \\ 0 & \tilde{X} & 0 & \cdots & 0 \\ 0 & 0 & \tilde{X} & \cdots & 0 \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ 0 & 0 & 0 & \cdots & \tilde{X} \end{pmatrix}$$

where each block $\tilde{X}$ is a $N_r \times N_a$ Vandermonde matrix of the form:

$$\tilde{X}=\begin{pmatrix} 1 & 1 & 1^2 & \cdots & 1^{N_a-1} \\ 1 & 2 & 2^2 & \cdots & 2^{N_a-1} \\ 1 & 3 & 3^2 & \cdots & 3^{N_a-1} \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ 1 & N_r & N_r^2 & \cdots & N_r^{N_a-1} \end{pmatrix}.$$

Note that, as long as $K \le N$, the drift matrix is full-rank.

5.2. Drift removal performance

We now discuss the drift removal performance, by analysing the residual drift y_{XP}. To this end we need to study Σ_{XP}. It is not difficult to verify that the $N \times 1$ constant vector $c=[1,1,\ldots,1]^T$ is in the span of both P and X and therefore $c \in \Sigma_{XP}$. Whether Σ_{XP} contains other linearly independent vectors cannot be discussed in general, since it depends on the specific P matrix, i.e. on the scan strategy. Nevertheless note that a vector $v \in \Sigma_P$ is such that $v=\Pi_P v$. By introducing $w=(P^TP)^{-1}P^Tv$ this can be written as $v=Pw$ and says that the vector v is such that by producing a LS image w from it and the corresponding signal estimate Pw, exactly the same vector v is reobtained. When, like in Herschel data, there is redundancy in the scan and several readouts fall into each pixel, a non constant polynomial vector does not have such a feature. Therefore Σ_{XP} only contains c and its scaled copies and the residual drift y_{XP} is a constant vector. This implies that the image produced from the updated data vector is affected by an unknown offset. However this is not a real problem, since Herschel images need in any case to be calibrated with an external absolute reference. We conclude that, in practice, our approach entirely removes the drift from Herschel data.

5.3. ALS implementation and complexity

The ALS implementation can be simplified by exploiting the structure of the pointing and drift matrices. Indeed, since P is sparse and binary, P^TP is diagonal and the inversion can be performed analytically. In practice, the value of each pixel of the LS image $\overline{m}$ computed in step 2 is just equal to the average of the readouts falling into the pixel. Then step 2 requires N sums and M divisions. Also, the production of the signal estimate in step 3 only requires N memory access operations. Moreover, since X is block diagonal, the inversion of X^TX reduces to the inversion of $\tilde{X}^T\tilde{X}$, which can be carried out numerically. In practice steps 5 and 6 can be realised separately for each timeline, by means of a polynomial fit, with NN_a multiply and add operations overall.

The computational complexity of one iteration of the ALS is dominated by steps 5 and 6 and is $O(NN_a)$. The number of iterations required to achieve convergence depends on the specific data but is normally not greater that one or two tens, as we see in the next section. Moreover, with a clever implementation, the ALS only requires to store one copy of the data vector, which is updated at each iteration.

5.4. JLS and GLS implementation

As a preliminary remark note that, based on the analysis of Section 5.2, $\Sigma_{XP} \neq 0$ and the matrix A^TA is singular. Then we proceed as described in Section 4.3. Namely, we rewrite the data as (12), where the matrix $\hat{X}$ is easily obtained from the matrix X by removing the first column, and use $A=[P,\hat{X}]$, making the problem non-singular. However, the direct computation of the JLS estimate by means of the inversion of A^TA is out of question for Herschel data. For the drift removal we could just compute the drift estimate of (11), but in any case we need to invert the $(K-1)\times(K-1)$ matrix $\hat{X}^T\Pi_P^{\perp}\hat{X}$ which, since K may exceed 10^4, is not possible numerically. Moreover there is no way to perform the inversion analytically. Then the JLS estimate has to be obtained by other means and the ALS is one of the options, but not the only one. A full discussion of this point is beyond the scope of the paper but we will consider a classical alternative, briefly described in the following.

[8] In this way we remove both the specific and the common drift. In order to save computation time, we could remove the common drift only, by using a different drift matrix, see [22].

By introducing the matrix $B = A^T A$ and the vector $b = A^T d$, Eq. (4) yields

$$B\overline{z} = b.$$

The latter is a linear system in the unknown vector $\overline{z}$, which can be solved using a standard method. A good choice is to use the Parallel Conjugate Gradient (PCG) [19] which is an iterative solver that, to be implemented, only requires to perform the multiplication of the matrix B with a vector at each iteration, avoiding explicit inversion. The multiplication can be implemented with $O(NN_a)$ multiply and add operations so that the PCG iteration has a complexity comparable to that of the ALS iteration. However, as we verify in Section 6.4, the PCG has a slower convergence with respect to ALS and is therefore less efficient.

Note that the PCG can be used to compute the JLS drift estimate of (11) directly. In this case each iteration requires the multiplication of a vector by the matrix $\hat{X}^T \Pi_P^{\perp} \hat{X}$, which can be performed with $O(NN_a)$ operations. We tested this solution too and verified that the convergence speed does not change significantly.

Concerning the GLS estimate, again it cannot be computed directly, but the PCG approach can be used to produce an approximation. This requires to compute the inverse of the covariance matrix, which can be done as described, e.g., in [18]. However, the GLS estimate is quite sensitive to the approximations discussed in Section 5.1, which may cause severe distortion in the final map. Indeed a post-processing step is normally needed to remove the distortion [20]. Overall, the implementation of the GLS is non-trivial and usually this step takes a large share of the total computational complexity.

6. Experiments and results

We developed a simulator producing a data vector according to the model of (3). The dimension of the vectors are kept small, in order to allow the evaluation of the estimates by means of matrix inversion. In particular, we will present results obtained using $M=10$, $N=100$ and a single timeline affected by a third order polynomial drift, i.e. $K=4$, $N_t=1$, but we tested other values too. In order to produce the data vector, an image vector m and a coefficients vector a are randomly produced and the matrix X is constructed. The image is next sampled with a random pattern and the corresponding pointing matrix P is produced. Then the signal vector is computed as $s=Pm$ and the drift vector as $y=\alpha Xa$ where α is a scalar gain factor, used to control the Signal to Drift Ratio (SDR), which is defined as $SDR = |\mathbf{s}|^2/|\mathbf{d}|^2$. Moreover, a covariance matrix C is produced, according to a noise spectrum having user-specified knee frequency and exponent, together with a zero-mean, unit-variance, white Gaussian vector w. The noise n is then obtained as $n = \beta C^{\frac{1}{2}} w$ where β is a scalar gain factor, used to control the Signal to Noise Ratio (SNR), which is defined as $SNR = |\mathbf{s}|^2/|\mathbf{n}|^2$.

6.1. Performance of the two steps approach

We compare the performance of the two steps approach (JLS-GLS) with the optimum approach, namely the JGLS. Given an image estimate $\overline{m}$, we compute the error as $e = m - \overline{m}$ and use the Image to Error Ratio (IER),

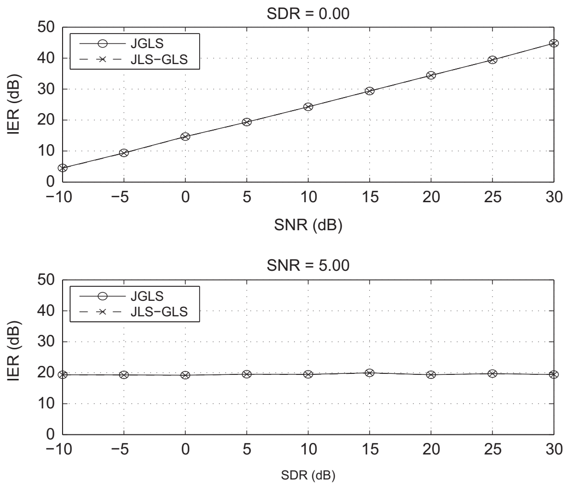

Fig. 2. IER of JLS and JGLS as a function of the SNR and SDR (simulated data).

given by $IER = |\mathbf{m}|^2/|\mathbf{e}|^2$, as the performance index.[9] In Fig. 2 the IER, averaged over 500 realisations, is plotted as a function of the SNR and of the SDR, when the normalised knee frequency is 0.1 and the frequency exponent 1.7. The figure shows that the two-steps approach has essentially the same performance of the JGLS and is therefore optimum in practice. Moreover, it confirms that the drift is entirely removed, since the IER is not affected by the SDR. The same results hold when varying the noise parameters and the polynomial order. More generally, we verified that the two-steps approach is nearly optimum as long as the covariance matrix is constructed assuming a stationary noise. In the GLS step, we used the regularised matrix of (13); however, virtually the same results are obtained using the original covariance matrix.

6.2. Polynomial order

In practice we do not know what is the correct polynomial order and need a way to set this parameter. To this end it is useful to consider the JLS estimation error, given by $e = d - A\overline{z}$, and define the Mean Square Error (MSE) as $MSE = |e|^2/N$. Next, we run the JLS with a varying polynomial order and plot the corresponding MSE. The results are reported in Fig. 3 and show that the MSE is decreasing for increasing order but, past the correct order (three), the change becomes negligible and higher orders yield almost no improvement. This behaviour has been observed in a number of other tests. Therefore a practical selection method is to increase the order until the MSE change gets below a user-specified threshold.

6.3. Data sets

We now switch to true Herschel data. In particular we consider two data sets: the first is taken in the 70 μm band

[9] Note that we use zero-mean vectors in the definition, thereby neglecting a possible offset affecting the estimate. As we have already seen, this is a good choice, because the offset is removed in the calibration phase following the image formation.

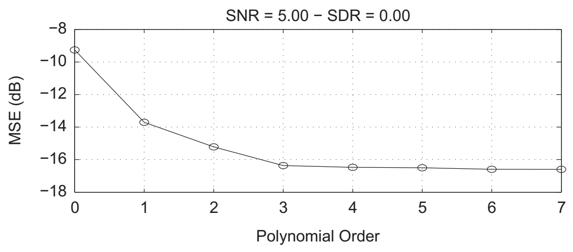

Fig. 3. MSE as a function of the JLS polynomial order when the drift is a third order polynomial (simulated data).

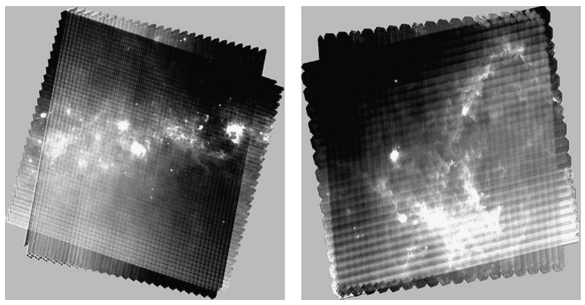

Fig. 4. LS image of L004 (left) and Rosette (right). Note the signal gradient at the edges and the nonflat background, caused by the drift, and the chequered pattern, due to the correlated noise.

over a Galaxy area of 2×2 degrees centered at a galactic longitude of 4 degrees (L004); the second is taken in the 160 μm band over an area of 1×1.7 degrees centered on the Rosette Nebula. Both areas were observed twice, along two orthogonal scans. For L004 each scan consists of 50 parallel lines. For Rosette one scan consists of 28 lines and the other of 36. In order to visualise the raw data, we compute the LS estimate of (1). The resulting images are shown in Fig. 4 and the impact of the drift and of the correlated noise is clearly seen. In particular, the drift causes the strong signal gradient observed at the image borders and the non flat background. The correlated noise causes the stripes following the scan lines and the corresponding chequered pattern.

6.4. *Convergence of ALS and PCG*

We study the convergence of the iterative drift removal methods, namely the ALS and the PCG, on the true Herschel data. To this end, in Fig. 5, we plot the MSE achieved by the two approaches[10] as a function of the number of iterations, when the polynomial order is three. Observing the figure we see that for both methods and both data sets the MSE is decreasing but the change becomes negligible after a some iterations. After this point additional iterations are useless. Then a practical convergence criterion is to stop when the MSE change gets below a user-specified threshold. Moreover, the figure shows that the ALS converges much faster than the

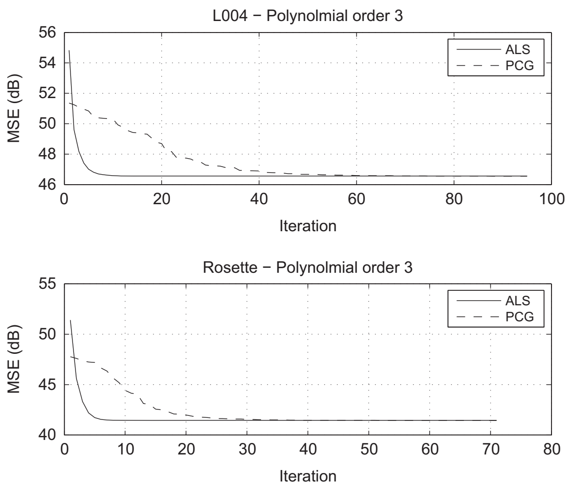

Fig. 5. MSE (dB) of ALS and PCG as a function of the number of iterations for L004 (top) and Rosette (bottom).

PCG. Indeed for L004 the ALS requires about ten iterations while the PCG requires about sixty. For Rosette the ALS requires about eighth iterations while the PCG requires about fifty. This behaviour has been confirmed by several other tests.

6.5. *Image formation example*

We discuss the image formation for L004 and Rosette. As a preliminary task, we select the polynomial order, by running the ALS with a varying order. We obtain the curves of Fig. 6, showing that the MSE decrease becomes negligible after order three for both data sets. Then we select this order in the following.

The impact of the first image formation step, i.e. the drift removal, can be appreciated by observing Fig. 7, which shows the image estimates produced by the ALS. Observing the figure we see that the gradient at the image borders is absent and that the background is flat, indicating that the drift has been removed. However the correlated noise is still affecting the images.

The correlated noise is removed in the second step, using the GLS approach. In particular, we compute the covariance matrix by estimating the noise spectrum from the updated data and implement the GLS estimate iteratively, using the PCG. Moreover, since the GLS approach introduces distortion, we also run the Post-processing for GLS (PGLS) algorithm, described in [20]. The resulting images are shown in Fig. 8. By comparing with Fig. 7 we can appreciate the improved quality.

7. Conclusion

We presented an effective approach to the image formation in the presence of drift and correlated noise, where these disturbances are removed in separate steps. The drift removal is carried out by solving a Joint LS problem and the noise removal by using a Generalised LS method. We also introduced an efficient, iterative method, based on Alternating LS, for solving the JLS problem. We

[10] For the ALS, the MSE can be computed as $|\eta|^2/N$, where η is the vector produced in step 4 of the algorithm. For the PCG, the MSE is computed based on the current estimates of the image and of the coefficients.

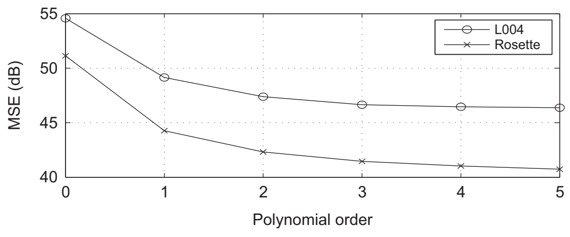

Fig. 6. MSE (dB) of ALS as a function of the polynomial order.

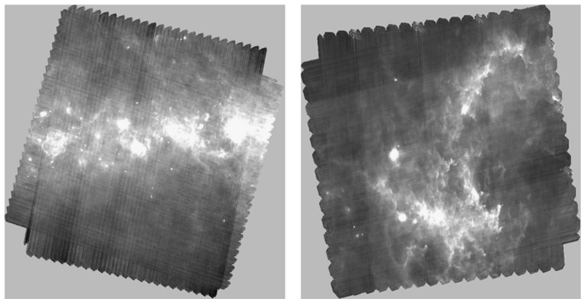

Fig. 7. JLS image of L004 (left) and Rosette (right), obtained by means of the ALS. Comparing with Fig. 4 we see that the gradient at the edges has been removed and that the background is flat.

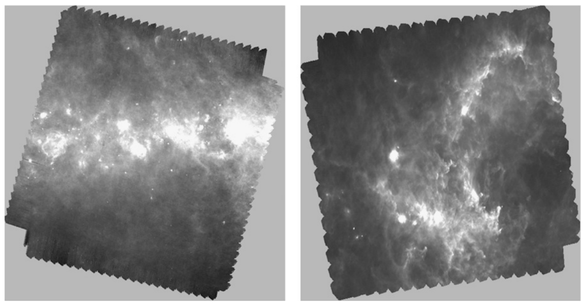

Fig. 8. PGLS image of L004 (left) and Rosette (right). Comparing with Fig. 7 we see that the correlated noise has been removed.

specialised the approach to Herschel data, showing that, for this application, it yields essentially optimal results.

Acknowledgements

The authors would like to thank Pierre Chanial (CEA, Saclay), Gaetano Scarano (DIET, Sapienza University) and Bernhard Schulz (NHSC, Caltech) for fruitful discussions and suggestions.

Appendix A

In order to prove (9) we start by showing that

$$(I-\Pi_X\Pi_P^\perp)v=v \quad \text{for } v\in\Sigma_P, \tag{14}$$

$$(I-\Pi_X\Pi_P^\perp)v=v \quad \text{for } v\in\Sigma_D^\perp. \tag{15}$$

Indeed if $v\in\Sigma_P$ we have $\Pi_P^\perp v=0$ and using this fact it is immediate to verify (14). To prove (15) note that when $v\in\Sigma_D^\perp$ we have[11] $\Pi_P^\perp v=v$ and $\Pi_X v=0$. Then

$$(I-\Pi_X\Pi_P^\perp)v=v-\Pi_X\Pi_P^\perp v=v-\Pi_X v=v.$$

Since $L=(I-\Pi_X\Pi_P^\perp)^\infty$, Eq. (9) follows immediately from (14) and (15), which also show that the ALS algorithm converges at the first step when $v\in\Sigma_P$ or $v\in\Sigma_D^\perp$.

In order to prove (10) we write the matrix L in a different form. Using $\Pi_P^\perp=I-\Pi_P$ and $\Pi_X^\perp=I-\Pi_X$ we have

$$I-\Pi_X\Pi_P^\perp=I-\Pi_X+\Pi_X\Pi_P=\Pi_X^\perp+\Pi_X\Pi_P$$

so that $L=(\Pi_X^\perp+\Pi_X\Pi_P)^\infty$. Now, for $v\in\Sigma_X$, consider $w=(\Pi_X^\perp+\Pi_X\Pi_P)v$ and note that

$$w=(\Pi_X^\perp+\Pi_X\Pi_P)v=\Pi_X^\perp v+\Pi_X\Pi_P v=\Pi_X\Pi_P v.$$

Moreover, the last term shows that $w\in\Sigma_X$. Then, we can repeat the reasoning again and again and obtain

$$v_k=(\Pi_X^\perp+\Pi_X\Pi_P)^k v=(\Pi_X\Pi_P)^k v,$$

with $v_k\in\Sigma_X$. Therefore, in the limit, we have

$$Lv=(\Pi_X\Pi_P)^\infty v \quad \text{for } v\in\Sigma_X$$

showing that L alternatively projects a drift vector onto the signal and drift subspaces infinitely many times. Then we can use the fact that $(\Pi_X\Pi_P)^\infty=\Pi_{XP}$ where Π_{XP} is the orthogonal projector onto Σ_{XP}. This result is due to Von-Neumann [21] and is known as the AP theorem. It can be found in many modern works, e.g. [7]. Using this fact we have

$$Lv=\Pi_{XP}v \quad \text{for } v\in\Sigma_X.$$

Since $\Sigma_Z\subset\Sigma_X$, from the last equation we see that the ALS algorithm converges for an input $v\in\Sigma_Z$. Moreover, Eq. (10) is obtained by noting that $v\in\Sigma_Z$ is orthogonal to all the vectors of Σ_{XP} so that $\Pi_{XP}v=0$.

Appendix B

In order to prove that H is the oblique projector onto Σ_Z along $\Sigma_P\oplus\Sigma_D^\perp$ we have to show that

$$Hv=v \quad \text{for } v\in\Sigma_Z \tag{16}$$

$$Hv=0 \quad \text{for } v\in\Sigma_P \text{ and } v\in\Sigma_D^\perp. \tag{17}$$

Concerning Eq. (16) note that $\Sigma_Z\subset\Sigma_X$, then $v\in\Sigma_X$ and can be written as $v=Xb$ for some vector b. Therefore

$$Hv=X(X^T\Pi_P^\perp X)^{-1}X^T\Pi_P^\perp Xb=Xb=v.$$

Concerning Eq. (17) note that if $v\in\Sigma_P$, then $\Pi_P^\perp v=0$ so that

$$Hv=X(X^T\Pi_P^\perp X)^{-1}X^T\Pi_P^\perp v=0.$$

[11] It is not difficult to verify that $\Sigma_D=\Sigma_P\oplus\Sigma_Z=\Sigma_P+\Sigma_X$. Then if $v\in\Sigma_D^\perp$ we also have $v\in\Sigma_P^\perp$ and $v\in\Sigma_X^\perp$. This fact is used several times in the appendix.

If $v \in \Sigma_D^\perp$, using $\Pi_P^\perp v = v$ and $X^T v = 0$, we have

$$Hv = X(X^T \Pi_P^\perp X)^{-1} X^T \Pi_P^\perp v = X(X^T \Pi_P^\perp X)^{-1} X^T v = 0.$$

Appendix C

We prove that $\overline{m} = \hat{m}$. Using standard matrix algebra it is easy to show that $\overline{m} = m + \overline{e}$ and $\hat{m} = m + \hat{e}$ where

$$\overline{e} = (P^T P)^{-1} P^T L n$$

and

$$\hat{e} = (P^T \Pi_X^\perp P)^{-1} P^T \Pi_X^\perp n$$

are the estimation error vectors. Then we have to show that $\overline{e} = \hat{e}$. To this end, using (8), we write $n = n_Z + n_P + n_{D^\perp}$ and decompose the error vectors accordingly, i.e. $\overline{e} = \overline{e}_Z + \overline{e}_P + \overline{e}_{D^\perp}$ and $\hat{e} = \hat{e}_Z + \hat{e}_P + \hat{e}_{D^\perp}$. Next we note that, since $Ln_Z = 0$ and $\Pi_X^\perp n_Z = 0$, we have $\hat{e}_Z = \overline{e}_Z = 0$. Now consider n_P and note that it can be written as $n_P = Pb$ for some vector b. Then, since $Ln_P = n_P$, we have

$$\begin{aligned}\overline{e}_P &= (P^T P)^{-1} P^T L n_P \\ &= (P^T P)^{-1} P^T n_P = (P^T P)^{-1} P^T P b = b.\end{aligned}$$

Moreover

$$\hat{e}_P = (P^T \Pi_X^\perp P)^{-1} P^T \Pi_X^\perp n_P = (P^T \Pi_X^\perp P)^{-1} P^T \Pi_X^\perp P b = b.$$

Then $\overline{e}_P = \hat{e}_P$. Finally, since $Ln_{D^\perp} = n_{D^\perp}$, $\Pi_X^\perp n_{D^\perp} = n_{D^\perp}$ and $P^T n_{D^\perp} = 0$, we have

$$\overline{e}_{D^\perp} = (P^T P)^{-1} P^T L n_{D^\perp} = (P^T P)^{-1} P^T n_{D^\perp} = 0,$$

$$\hat{e}_{D^\perp} = (P^T \Pi_X^\perp P)^{-1} P^T \Pi_X^\perp n_{D^\perp} = (P^T \Pi_X^\perp P)^{-1} P^T n_{D^\perp} = 0.$$

Then $\overline{e}_{D^\perp} = \hat{e}_{D^\perp}$ and the proof is complete.

References

[1] H. Luo, S. Puthusserypady, Analysis of fMRI data with drift: modified general linear model and Bayesian estimator, IEEE Trans. Biomed. Eng. 55 (May (5)) (2008) 1504–1511.

[2] C. Matteau-Pelletier, M. Dehaes, F. Lesage, J. Lina, Noise in diffuse optical imaging and wavelet-based response estimation, IEEE Trans. Med. Imaging 28 (March (3)) (2009) 415–422.

[3] B. Schaffer, W. Grogger, G. Kothleitner, Automated spatial drift correction for EFTEM image series, Ultramicroscopy 102 (2004) 27–36.

[4] D.J. Fixsen, S.H. Moseley, R.G. Arendt, Calibrating array detectors, Astrophys. J. Suppl. Ser. 128 (June (2)) (2000) 651–658.

[5] A. Traficante, et al., The data reduction pipeline for the Hi-GAL survey, Mon. Not. R. Astron. Soc. 416 (October (4)) (2011) 2932–2943.

[6] H.A.L. Kiers, Setting up alternating least squares and iterative majorization algorithms for solving various matrix optimization problems, Comput. Stat. Data Anal. 41 (2002) 157–170.

[7] F. Deutsch, The method of alternating orthogonal projections, NATO ASI Ser. 356 (1992) 105–121.

[8] G. Pilbratt, et al., Herschel space observatory, Astron. Astrophys. 518 (July (7–8)) (2010).

[9] L. Piazzo, Unimap, ESA Map Making Workshop, Madrid, Jan. 2013. Available: ⟨http://herschel.esac.esa.int/2013MapmakingWorkshop/presentations/Unimap_Lpiazzo_MapMaking2013_DPWS.pdf⟩.

[10] P. Chanial, P. Panuzzo, Tamasis, ESA Map Making Workshop, Madrid, Jan. 2013. Available: ⟨http://herschel.esac.esa.int/2013MapmakingWorkshop/presentations/TAMASIS_PChanial_MapMaking2013_DPWS.pdf⟩.

[11] B. Schulz, et. al, The SPIRE destriper, ESA Map Making Workshop, Madrid, Jan. 2013. Available: ⟨http://herschel.esac.esa.int/2013MapmakingWorkshop/presentations/DestriperMapping_BSchulz_MapMaking2013_DPWS.pdf⟩.

[12] C.D. Meyer, Matrix Analysis and Applied Linear Algebra, SIAM, Philadelphia, PA, 2000.

[13] A.C. Aitken, On least squares and linear combinations of observations, Proc. R. Soc. Edinb. 55 (1935) 42–48.

[14] Block Matrix Pseudoinverse, Wikipedia, The Free Encyclopedia, 2013. Available: ⟨http://en.wikipedia.org/w/index.php?title=Block_matrix_pseudoinverse&oldid=578128609⟩.

[15] A. Poglitsch, et al., The photodetector array camera and spectrometer (PACS) on the herschel space observatory, Astron. Astrophys. 518 (4) (2010).

[16] M.J. Griffin, et al., The Herschel-SPIRE instrument and its in-flight performance, Astron. Astrophys. 518 (July–August) (2010).

[17] PACS Data Reduction Guide, Version 10, Apr. 2012, Available: ⟨http://herschel.esac.esa.int/Data_Processing.shtml⟩.

[18] C.M. Cantalupo, J.D. Borrill, A.H. Jaffe, T.S. Kisner, R. Stompor, MADmap: a massively parallel maximum likelihood cosmic microwave background map-maker, Astrophys. J. Suppl. Ser. 187 (1) (2010) 212–227.

[19] W.T. Vetterling, W.H. Press, S.A. Teukolsky, B.P. Flannery, Numerical Recipes in C, Cambridge University Press, Cambridge, UK, 1992.

[20] L. Piazzo, D. Ikhenaode, P. Natoli, M. Pestalozzi, F. Piacentini, A. Traficante, Artifact removal for GLS map makers by means of post-processing, IEEE Trans. Image Process. 21 (8) (2012) 3687–3696.

[21] J. Von-Neumann, On rings of operators. Reduction theory, Ann. Math. 50 (1949) 401–485.

[22] L. Piazzo, Subspace Least Square Approach for Drift Removal with Application to Herschel Data, arxiv:1301.1246, 2013.