Chapter 4

Construction of the algorithms

In this chapter, we describe the construction of the algorithms asserted in our main results. These are based on techniques from compressed sensing [8, 13, 61] on the premise that the polynomial coefficients of a holomorphic function are approximately sparse. There are several main differences between standard compressed sensing and what we develop below. First, following [2, 7, 8, 37, 119, 121], we work in a weighted setting in order to promote sparsity in lower or anchored sets (recall Section 2.7). Second, following [52], we work with Hilbert-valued vectors, whose entries take values in the Hilbert space $\mathcal{V}$. Finally, so as to avoid unrealistic assumptions on the functions being approximated, we use consider *noise-blind* decoders, as in [3]. See also Remark 4.1.

4.1 Recovery via Hilbert-valued, weighted ℓ^1-minimization

We first require some additional notation. Given $N \in \mathbb{N}$ we let $\mathcal{V}^N$ be the vector space of Hilbert-valued vectors of length N, i.e., $\boldsymbol{v} = (v_i)_{i=1}^N$ where $v_i \in \mathcal{V}$, $i = 1, \dots, N$. Next, given $\Lambda \subseteq \mathcal{F}$ and a vector of positive weights $\boldsymbol{w} = (w_{\boldsymbol{\nu}})_{\boldsymbol{\nu} \in \Lambda}$, where $\boldsymbol{w} > \boldsymbol{0}$, we define the weighted $\ell^p_{\boldsymbol{w}}(\Lambda; \mathcal{V})$ space, $0 < p \leq 2$, as the set of $\mathcal{V}$-valued sequences $\boldsymbol{v} = (v_{\boldsymbol{\nu}})_{\boldsymbol{\nu} \in \Lambda}$ for which

$$\|\boldsymbol{v}\|_{p,\boldsymbol{w};\mathcal{V}} := \left(\sum_{\boldsymbol{\nu} \in \Lambda} w_{\boldsymbol{\nu}}^{2-p} \|v_{\boldsymbol{\nu}}\|_{\mathcal{V}}^p \right)^{1/p} < \infty.$$

Notice that $\ell^2_{\boldsymbol{w}}(\Lambda; \mathcal{V})$ coincides with the unweighted space $\ell^2(\Lambda; \mathcal{V})$.

Now, let $\Lambda \subset \mathcal{F}$ be a finite multi-index set of size $|\Lambda| = N$ and consider the ordering $\Lambda = \{\boldsymbol{\nu}_1, \dots, \boldsymbol{\nu}_N\}$. Note that we will, in practice, choose either $\Lambda = \Lambda^{\mathsf{HC}}_{n,d}$ when $d < \infty$ or $\Lambda = \Lambda^{\mathsf{HCI}}_n$ when $d = \infty$, where the order n is as described in the corresponding theorem (Theorems 3.4–3.12). With this in mind, given $f \in L^2_\varrho(\mathcal{U}; \mathcal{V})$, define

$$f_\Lambda = \sum_{\boldsymbol{\nu} \in \Lambda} c_{\boldsymbol{\nu}} \Psi_{\boldsymbol{\nu}} \tag{4.1}$$

as the truncated expansion of f based on the index set Λ and

$$\boldsymbol{c}_\Lambda = (c_{\boldsymbol{\nu}_j})_{j=1}^N \in \mathcal{V}^N \tag{4.2}$$

as the finite vector of coefficients of f with indices in Λ. As explained in Section 3.2, our objective is, in effect, to approximate these coefficients.

We do this as follows. Given $y_1, \dots, y_m \in \mathcal{U}$, we define the normalized measurement matrix

$$\boldsymbol{A} = \left(\frac{\Psi_{\boldsymbol{v}_j}(\boldsymbol{y}_i)}{\sqrt{m}} \right)_{i,j=1}^{m,N} \in \mathbb{C}^{m \times N} \tag{4.3}$$

and the normalized measurement and error vectors

$$\boldsymbol{b} = \frac{1}{\sqrt{m}} (f(\boldsymbol{y}_i) + n_i)_{i=1}^m \in \mathcal{V}_h^m, \quad \boldsymbol{e} = \frac{1}{\sqrt{m}} (n_i)_{i=1}^m \in \mathcal{V}^m. \tag{4.4}$$

Notice that any $m \times N$ matrix $\boldsymbol{A} = (a_{ij})_{i,j=1}^{m,N}$ extends to a bounded linear operator $\mathcal{V}^N \to \mathcal{V}^m$ (or $\mathcal{V}_h^N \to \mathcal{V}_h^m$) in the obvious way, i.e.,

$$\boldsymbol{x} = (x_i)_{i=1}^N \in \mathcal{V}^N \mapsto \boldsymbol{A}\boldsymbol{x} = \left(\sum_{j=1}^N a_{ij} x_j \right)_{i=1}^m \in \mathcal{V}^m.$$

For ease of notation, we make no distinction between the matrix $\boldsymbol{A} \in \mathbb{C}^{m \times N}$ and the linear operator $\boldsymbol{A} \in \mathcal{B}(\mathcal{V}^N, \mathcal{V}^m)$ (or $\boldsymbol{A} \in \mathcal{B}(\mathcal{V}_h^N, \mathcal{V}_h^m)$) in what follows. Using this, we obtain

$$\boldsymbol{A}\boldsymbol{c}_\Lambda = \frac{1}{\sqrt{m}} (f_\Lambda(\boldsymbol{y}_i))_{i=1}^m = \frac{1}{\sqrt{m}} (f(\boldsymbol{y}_i))_{i=1}^m - \frac{1}{\sqrt{m}} (f(\boldsymbol{y}_i) - f_\Lambda(\boldsymbol{y}_i))_{i=1}^m,$$

and therefore

$$\boldsymbol{A}\boldsymbol{c}_\Lambda + \boldsymbol{e} + \boldsymbol{e}' = \boldsymbol{b}, \tag{4.5}$$

where

$$\boldsymbol{e}' = \frac{1}{\sqrt{m}} (f(\boldsymbol{y}_i) - f_\Lambda(\boldsymbol{y}_i))_{i=1}^m.$$

We have now formulated the recovery of $\boldsymbol{c}_\Lambda$ as the solution of a noisy linear system (4.5), where the noise term $\boldsymbol{e} + \boldsymbol{e}'$ encompasses both the noise $\boldsymbol{e} = (n_i)_{i=1}^m / \sqrt{m}$ in the sample values and the error $\boldsymbol{e}'$ due to the truncation (4.1) of the infinite expansion (2.8) via the index set Λ.

Due to the discussion in Sections 2.5–2.7, we expect the coefficients $\boldsymbol{c}_\Lambda$ to not only be approximately sparse, but also well approximated by a subset of s coefficients whose indices define a lower or anchored set. In classical compressed sensing, one exploits sparse structure via minimizing an ℓ^1-norm. To exploit sparse and lower structure, we follow ideas of [2, 7, 8, 37] and use a weighted ℓ^1-norm penalty. Specifically, we now compute an approximate solution via the Hilbert-valued, *weighted Square-Root LASSO (SR-LASSO)* optimization problem

$$\min_{\boldsymbol{z} \in \mathcal{V}_h^N} \mathcal{G}(\boldsymbol{z}), \quad \mathcal{G}(\boldsymbol{z}) := \lambda \|\boldsymbol{z}\|_{1,\boldsymbol{w};\mathcal{V}} + \|\boldsymbol{A}\boldsymbol{z} - \boldsymbol{b}\|_{2;\mathcal{V}}. \tag{4.6}$$

Here $\lambda > 0$ is a tuning hyperparameter.

Remark 4.1. As an alternative to solve this Hilbert-valued compressed sensing problem, we could use a formulation based on a constrained basis pursuit or unconstrained LASSO problem. However, we consider the SR-LASSO problem (4.6) instead. While other approaches are arguably more common, based on [3] the SR-LASSO has the desirable property that the optimal values of its hyperparameter λ is independent of the noise term (in this case $\boldsymbol{e} + \boldsymbol{e}'$). This is not the case for other formulations, whose hyperparameters need to be chosen in terms of the (unknown) magnitude of the noise in order to ensure good theoretical and practical performance (see, e.g., [13, Chapter 6]). This is particularly problematic in the setting of function approximation, where such terms are function dependent (for instance, the term $\boldsymbol{e}'$ depends on the expansion tail $f - f_\Lambda$) and therefore generally unknown. See [3] and [8, Section 6.6] for further discussion.

Notice that (4.6) is solved over $\mathcal{V}_h^N$ not $\mathcal{V}^N$, since the latter would not be numerically solvable in general. As we see below, it can be reformulated an optimization problem over $\mathbb{C}^{N\times K}$, where $K = \dim(\mathcal{V}_h)$. However, since the true coefficients of f are elements of $\mathcal{V}$ and not $\mathcal{V}_h$, this discretization inevitably results in an additional error, which must also be accounted for in the analysis. This leads precisely to the physical discretization error (term (iii) in Section 3.3.1).

Finally, we now also specify the weights. Following [2,7,37] (see also [8, Remark 2.14]), a good choice of weights (for promoting lower or anchored structure) is given by the so-called *intrinsic weights*

$$\boldsymbol{w} = \boldsymbol{u} = (u_{\boldsymbol{\nu}})_{\boldsymbol{\nu}\in\Lambda}, \quad u_{\boldsymbol{\nu}} = \|\Psi_{\boldsymbol{\nu}}\|_{L^\infty(\mathcal{U})}, \quad \boldsymbol{\nu} \in \Lambda. \tag{4.7}$$

In particular, for Chebyshev and Legendre polynomials these are given explicitly by

$$u_{\boldsymbol{\nu}} = \|\Psi_{\boldsymbol{\nu}}\|_{L^\infty(\mathcal{U})} = \begin{cases} \prod_{j=1}^d \sqrt{2\nu_j + 1}, & \text{Legendre,} \\ 2^{\|\boldsymbol{\nu}\|_0/2}, & \text{Chebyshev,} \end{cases}$$

where $\|\boldsymbol{\nu}\|_0 := |\mathrm{supp}(\nu)|$. Typically, we index these weights over the multi-indices $\boldsymbol{\nu} \in \Lambda$. However, we will, for convenience, often write w_i instead of $w_{\boldsymbol{\nu}_i}$ in what follows, where, as above, $\{\boldsymbol{\nu}_1, \dots, \boldsymbol{\nu}_N\}$ is an ordering of Λ.

4.2 Reformulation as a matrix recovery problem and the mappings in Theorems 3.4, 3.7 and 3.10

We now describe the mappings whose existence is asserted in Theorems 3.4, 3.7 and 3.10. These maps all arise via exact solutions of weighted SR-LASSO optimization problems. However, since (4.6) yields a vector in $\mathcal{V}_h^N$ and the mappings should yield outputs in $\mathbb{C}^{N\times K}$, we first need to reformulate (4.6) using the basis $\{\varphi_i\}_{i=1}^K$ for $\mathcal{V}_h$.

Notice first that any vector of coefficients $\boldsymbol{c} = (c_{\boldsymbol{v}_i})_{i=1}^N \in \mathcal{V}_h^N$ is equivalent to a matrix of coefficients

$$\boldsymbol{C} = (c_{ik})_{i,k=1}^{N,K} \in \mathbb{C}^{N\times K},$$

via the relation

$$c_{\boldsymbol{v}_i} = \sum_{k=1}^K c_{ik}\varphi_k, \quad i \in [N].$$

Next, observe that if $g = \sum_{k=1}^K d_k\varphi_k \in \mathcal{V}_h$ then

$$\|g\|_{\mathcal{V}} = \|\boldsymbol{d}\|_{\boldsymbol{G}} = \sqrt{\boldsymbol{d}^*\boldsymbol{G}\boldsymbol{d}},$$

where $\boldsymbol{d} = (d_k)_{k=1}^K \in \mathbb{C}^K$ and $\boldsymbol{G} \in \mathbb{C}^{K\times K}$ is the Gram matrix for $\{\varphi_k\}_{k=1}^K$, given by (3.3). Since $\boldsymbol{G}$ is positive definite, it has a unique positive definite square root matrix $\boldsymbol{G}^{1/2}$. Hence, we may write

$$\|g\|_{\mathcal{V}} = \|\boldsymbol{G}^{1/2}\boldsymbol{d}\|_2.$$

We now use some additional notation. Given $1 \leq p \leq \infty$ and $1 \leq q \leq 2$, we define the weighted $\ell_{\boldsymbol{w}}^{p,q}$-norm of a matrix $\boldsymbol{C} = (c_{ik})_{i,k=1}^{N,K} \in \mathbb{C}^{N\times K}$ as

$$\|\boldsymbol{C}\|_{p,q,\boldsymbol{w}} = \left(\sum_{i=1}^N w_i^{2-p}\left(\sum_{k=1}^K |c_{ik}|^q\right)^{p/q}\right)^{1/p}.$$

Note that this is precisely the weighted $\ell_{\boldsymbol{w}}^p$-norm of the vector of $(\|\boldsymbol{c}_i\|_q)_{i=1}^N$, where $\boldsymbol{c}_i = (c_{ik})_{k=1}^K \in \mathbb{C}^K$ is the ith row of $\boldsymbol{C}$. Further, if $p = q = 2$, then this is just the unweighted $\ell^{2,2}$-norm of a matrix (which is simply its Frobenius norm). In this case, we typically write $\|\cdot\|_{2,2}$.

Now let $\boldsymbol{z} \in \mathcal{V}_h^N$ be arbitrary, $\boldsymbol{Z} \in \mathbb{C}^{N\times K}$ be the corresponding matrix and $\boldsymbol{z}_i \in \mathbb{C}^K$ be the ith row of $\boldsymbol{Z}$. Then

$$\|\boldsymbol{z}\|_{1,\boldsymbol{w};\mathcal{V}} = \sum_{i=1}^N w_i\|z_{\boldsymbol{v}_i}\|_{\mathcal{V}} = \sum_{i=1}^N w_i\|\boldsymbol{G}^{1/2}\boldsymbol{z}_i\|_2 = \|\boldsymbol{Z}\boldsymbol{G}^{1/2}\|_{2,1,\boldsymbol{w}}.$$

Similarly, let $\boldsymbol{A} = (a_{ij})_{i,j=1}^{m,N} \in \mathbb{C}^{m\times N}$ and $\boldsymbol{b} = (b_i)_{i=1}^m \in \mathcal{V}_h^m$ be as in (4.3) and (4.4), respectively, and let $\boldsymbol{B} \in \mathbb{C}^{m\times K}$ be the matrix corresponding to $\boldsymbol{b}$. Then

$$\|\boldsymbol{A}\boldsymbol{z} - \boldsymbol{b}\|_{2;\mathcal{V}}^2 = \sum_{i=1}^m \left\|\sum_{j=1}^N a_{ij}z_{\boldsymbol{v}_i} - b_i\right\|_{\mathcal{V}}^2 = \|(\boldsymbol{A}\boldsymbol{Z} - \boldsymbol{B})\boldsymbol{G}^{1/2}\|_{2,2}^2.$$

Therefore, we now consider the minimization problem

$$\min_{\boldsymbol{Z}\in\mathbb{C}^{N\times K}} \left\{\lambda\|\boldsymbol{Z}\|_{2,1,\boldsymbol{w}} + \|(\boldsymbol{A}\boldsymbol{Z} - \boldsymbol{B})\boldsymbol{G}^{1/2}\|_{2,2}\right\}. \tag{4.8}$$

- Let m, ϵ and n be as given in the particular theorem and set $\Lambda = \Lambda^{\mathrm{HC}}_{n,d}$ (Theorems 3.4 and 3.10) or $\Lambda = \Lambda^{\mathrm{HCI}}_{n}$ (Theorem 3.7).
- Set $\lambda = (4\sqrt{m/L})^{-1}$, where $L = L(m, d, \epsilon)$ is as in (3.8).
- Let $\boldsymbol{D} = (d_{ik})_{i,k=1}^{m,K} \in \mathbb{C}^{m\times K}$ and $\boldsymbol{Y} = (\boldsymbol{y}_i)_{i=1}^{m}$ be an input, as in (3.1), and set $\boldsymbol{B} = \frac{1}{\sqrt{m}}\boldsymbol{D}$.
- Let $\boldsymbol{G}$, $\boldsymbol{A}$ and $\boldsymbol{w}$ be as in (3.3), (4.3) and (4.7), respectively.
- Define the output $\widehat{\boldsymbol{C}} = \mathcal{M}(\boldsymbol{Y}, \boldsymbol{D})$ as the minimizer of (4.8) with smallest $\ell^{2,2}$-norm.

Table 4.1. The mappings $\mathcal{M} : \mathcal{U}^m \times \mathbb{C}^{m\times K} \to \mathbb{C}^{N\times K}$ used in Theorems 3.4, 3.7 and 3.10.

This is equivalent to (4.6) in the following sense. A vector $\hat{\boldsymbol{c}} = (\hat{c}_{\boldsymbol{v}_i})_{i=1}^{N} \in \mathcal{V}_h^N$ is a minimizer of (4.6) if and only if the matrix $\widehat{\boldsymbol{C}} = (\hat{c}_{ik})_{i,k=1}^{N,K} \in \mathbb{C}^{N\times K}$ with entries defined by the relation

$$\hat{c}_{\boldsymbol{v}_i} = \sum_{k=1}^{K} \hat{c}_{ik}\varphi_k, \quad i \in [N],$$

is a minimizer of (4.8).

With this in hand, we are now ready to define the mappings used in Theorems 3.4, 3.7 and 3.10. These are described in Table 4.1. Note that these are indeed well-defined mappings, since the minimizer of (4.8) with smallest $\ell^{2,2}$-norm is unique (this follows from the facts that (4.8) is a convex problem, therefore its set of minimizers is a convex set, and the function $\boldsymbol{Z} \mapsto \|\boldsymbol{Z}\|_{2,2}^2$ is strongly convex). This particular choice is arbitrary, and is made solely so as to have a well-defined mapping. It is of no consequence whatsoever in our analysis, since the various error bounds we prove later hold for any minimizer of (4.8).

4.3 The primal-dual iteration

To derive the algorithms described in the other main theorems, we need methods for approximately solving the optimization problems (4.6) and (4.8). We use the *primal-dual iteration* [30] (also known as the Chambolle–Pock algorithm) to this end. We first briefly describe the primal-dual iteration in the general case (see [30, 31, 31], as well as [13, Section 7.5] for more detailed treatments), before specializing to the weighted SR-LASSO problem in the next section.

Let $(\mathcal{X}, \langle\cdot,\cdot\rangle_{\mathcal{X}})$ and $(\mathcal{Y}, \langle\cdot,\cdot\rangle_{\mathcal{Y}})$ be (complex) Hilbert spaces, $g : \mathcal{X} \to \mathbb{R} \cup \{\infty\}$, $h : \mathcal{Y} \to \mathbb{R} \cup \{\infty\}$ be proper, lower semicontinuous and convex functions and

$A \in \mathcal{B}(\mathcal{X}, \mathcal{Y})$ be a bounded linear operator satisfying

$$\mathrm{dom}(h) \cap A(\mathrm{dom}(g)) \neq \emptyset.$$

The primal-dual iteration is a general method for solving the convex optimization problem

$$\min_{x \in \mathcal{X}} \{g(x) + h(A(x))\}. \tag{4.9}$$

Under this setting the (Fenchel–Rockafeller) dual problem is

$$\min_{\xi \in \mathcal{Y}} \{g^*(A^*\xi) + h^*(-\xi)\},$$

where g^* and h^* are the convex conjugate functions of g and h, respectively. Recall that, for a function $f : \mathcal{X} \to \mathbb{R} \cup \{\infty\}$, its convex conjugate is defined by

$$f^*(z) = \sup_{x \in \mathcal{X}} (\mathrm{Re}\langle x, z\rangle_{\mathcal{Y}} - f(x)), \quad z \in \mathcal{X}. \tag{4.10}$$

The Lagrangian of (4.9) is defined by

$$\mathcal{L}(x, \xi) = g(x) + \mathrm{Re}\langle A(x), \xi\rangle_{\mathcal{Y}} - h^*(\xi), \quad x \in \mathrm{dom}(g),\ \xi \in \mathrm{dom}(h^*), \tag{4.11}$$

and $\mathcal{L}(x, \xi) = \infty$ if $x \notin \mathrm{dom}(g)$ or $\mathcal{L}(x, \xi) = -\infty$ if $\xi \notin \mathrm{dom}(h^*)$. This in turn leads to the saddle-point formulation of the problem

$$\min_{x \in \mathcal{X}} \max_{\xi \in \mathcal{Y}} \mathcal{L}(x, \xi).$$

The primal-dual iteration seeks a solution $(\hat{x}, \hat{\xi})$ of the saddle-point problem by solving the following fixed-point equation

$$\begin{aligned} \hat{x} &= \mathrm{prox}_{\tau g}(\hat{x} - \tau A^*(\hat{\xi})), \\ \hat{\xi} &= \mathrm{prox}_{\sigma h^*}(\hat{\xi} + \sigma A(\hat{x})), \end{aligned}$$

where $\tau, \sigma > 0$ are stepsize parameters and prox is the proximal operator, which is defined by

$$\mathrm{prox}_f(z) = \arg\min_{x \in \mathcal{X}} \left\{ f(x) + \frac{1}{2}\|x - z\|_{\mathcal{X}}^2 \right\}, \quad z \in \mathrm{dom}(f).$$

To be precise, given initial values $(x^{(0)}, \xi^{(0)}) \in \mathcal{X} \times \mathcal{Y}$ the primal-dual iteration defines a sequence $\{(x^{(n)}, \xi^{(n)})\}_{n=1}^{\infty} \subset \mathcal{X} \times \mathcal{Y}$ as follows:

$$\begin{aligned} x^{(n+1)} &= \mathrm{prox}_{\tau g}\big(x^{(n)} - \tau A^*\big(\xi^{(n)}\big)\big), \\ \xi^{(n+1)} &= \mathrm{prox}_{\sigma h^*}\big(\xi^{(n)} + \sigma A\big(2x^{(n+1)} - x^{(n)}\big)\big). \end{aligned} \tag{4.12}$$

4.4 The primal-dual iteration for the weighted SR-LASSO problem

We now apply this scheme to (4.6) and (4.8). We first describe an algorithm to approximately solve the Hilbert-valued problem (4.6), before using the equivalence between elements of $\mathcal{V}_h^N$ and $\mathbb{C}^{N\times K}$ to obtain an algorithm for approximately solving (4.8).

Consider (4.6). We define $\mathcal{X} = (\mathcal{V}_h^N, \langle\cdot,\cdot\rangle_{2;\mathcal{V}})$, $\mathcal{Y} = (\mathcal{V}_h^m, \langle\cdot,\cdot\rangle_{2;\mathcal{V}})$ and $g : \mathcal{X} \to \mathbb{R}\cup\{\infty\}$, $h : \mathcal{Y} \to \mathbb{R}\cup\{\infty\}$ as the proper, lower semicontinuous and convex functions

$$g(\boldsymbol{x}) = \lambda\|\boldsymbol{x}\|_{1,\boldsymbol{w};\mathcal{V}}, \quad h(\boldsymbol{y}) = \|\boldsymbol{y}-\boldsymbol{b}\|_{2;\mathcal{V}}, \quad \boldsymbol{x}\in\mathcal{V}_h^N,\ \boldsymbol{y}\in\mathcal{V}_h^m.$$

We first find the proximal maps of g and h^*. Using (4.10), we see that

$$\begin{aligned} h^*(\boldsymbol{\xi}) &= \sup_{\boldsymbol{v}\in\mathcal{V}_h^m} (\mathrm{Re}\langle\boldsymbol{v},\boldsymbol{\xi}\rangle_{\mathcal{V}} - \|\boldsymbol{v}-\boldsymbol{b}\|_{2;\mathcal{V}}) \\ &= \mathrm{Re}\langle\boldsymbol{b},\boldsymbol{\xi}\rangle_{\mathcal{V}} + \sup_{\boldsymbol{v}\in\mathcal{V}_h^m} (\mathrm{Re}\langle\boldsymbol{v},\boldsymbol{\xi}\rangle_{\mathcal{V}} - \|\boldsymbol{v}\|_{2;\mathcal{V}}), \end{aligned}$$

for all $\boldsymbol{\xi}\in\mathcal{V}_h^m$. From [22, Examples 13.3 and 13.4] it follows that

$$(\|\cdot\|_{\mathcal{V}})^* = \delta_B, \quad B := \{\boldsymbol{\xi}\in\mathcal{V}_h^m : \|\boldsymbol{\xi}\|_{2;\mathcal{V}}\le 1\},$$

where δ_B is the indicator function of the set B, taking value $\delta_B(\boldsymbol{\xi}) = 0$ when $\boldsymbol{\xi}\in B$ and $+\infty$ otherwise. Hence,

$$h^*(\boldsymbol{\xi}) = \mathrm{Re}\langle\boldsymbol{b},\boldsymbol{\xi}\rangle_{\mathcal{V}} + \delta_B(\boldsymbol{\xi}). \tag{4.13}$$

Using this, we obtain

$$\begin{aligned} \mathrm{prox}_{\sigma h^*}(\boldsymbol{\xi}) &= \arg\min_{\boldsymbol{z}\in\mathcal{V}_h^m}\left\{\sigma\delta_B(\boldsymbol{z}) + \sigma\,\mathrm{Re}\langle\boldsymbol{b},\boldsymbol{z}\rangle_{\mathcal{V}} + \frac{1}{2}\|\boldsymbol{z}-\boldsymbol{\xi}\|_{2;\mathcal{V}}^2\right\} \\ &= \arg\min_{\boldsymbol{z}:\|\boldsymbol{z}\|_{2;\mathcal{V}}\le 1}\left\{\frac{1}{2}\|\boldsymbol{z}-(\boldsymbol{\xi}-\sigma\boldsymbol{b})\|_{2;\mathcal{V}}^2\right\} \\ &= \mathrm{proj}_B(\boldsymbol{\xi}-\sigma\boldsymbol{b}), \end{aligned}$$

where proj_B is the projection onto B, which is given explicitly by

$$\mathrm{proj}_B(\boldsymbol{\xi}) = \min\left\{1, \frac{1}{\|\boldsymbol{\xi}\|_{2;\mathcal{V}}}\right\}\boldsymbol{\xi}.$$

On the other hand, applying the definition of the proximal operator to the function τg with parameter $\tau > 0$, we deduce that

$$(\mathrm{prox}_{\tau g}(\boldsymbol{x}))_i = \mathrm{prox}_{\tau w_i\lambda\|\cdot\|_{\mathcal{V}}}(x_i), \quad i = 1,\dots,N, \quad \text{where } \boldsymbol{x} = (x_i)_{i=1}^N \in \mathcal{V}_h^N.$$

Algorithm 1: `primal-dual-wSRLASSO` – the primal-dual iteration for the weighted SR-LASSO problem (4.6)

inputs : measurement matrix $\boldsymbol{A} \in \mathbb{C}^{m\times N}$, measurements $\boldsymbol{b} \in \mathcal{V}_h^N$, positive weights $\boldsymbol{w} = (w_i)_{i=1}^N$, parameter $\lambda > 0$, stepsizes $\tau, \sigma > 0$, maximum number of iterations $T \geq 1$, initial values $\boldsymbol{c}^{(0)} \in \mathcal{V}_h^N$, $\boldsymbol{\xi}^{(0)} \in \mathcal{V}_h^m$

output : $\bar{\boldsymbol{c}} = \texttt{primal-dual-wSRLASSO}(\boldsymbol{A}, \boldsymbol{b}, \boldsymbol{w}, \lambda, \tau, \sigma, T, \boldsymbol{c}^{(0)}, \boldsymbol{\xi}^{(0)})$, an approximate minimizer of (4.6)

initialize: $\bar{\boldsymbol{c}}^{(0)} = \boldsymbol{0} \in \mathcal{V}_h^N$

for $n = 0, 1, \ldots, T-1$ **do**

$\quad \boldsymbol{p} = (p_i)_{j=1}^N = \boldsymbol{c}^{(n)} - \tau \boldsymbol{A}^* \boldsymbol{\xi}^{(n)}$

$\quad \boldsymbol{c}^{(n+1)} = \left(\max\{\|p_i\|_{\mathcal{V}} - \tau\lambda w_i, 0\} \frac{p_i}{\|p_i\|_{\mathcal{V}}}\right)_{i=1}^N$

$\quad \boldsymbol{q} = \boldsymbol{\xi}^{(n)} + \sigma \boldsymbol{A}(2\boldsymbol{c}^{(n+1)} - \boldsymbol{c}^{(n)}) - \sigma \boldsymbol{b}$

$\quad \boldsymbol{\xi}^{(n+1)} = \min\left\{1, \frac{1}{\|\boldsymbol{q}\|_{2;\mathcal{V}}}\right\} \boldsymbol{q}$

$\quad \bar{\boldsymbol{c}}^{(n+1)} = \frac{n}{n+1} \bar{\boldsymbol{c}}^{(n)} + \frac{1}{n+1} \boldsymbol{c}^{(n+1)}$

end

$\bar{\boldsymbol{c}} = \bar{\boldsymbol{c}}^{(T)}$

Moreover, a simple adaptation of [22, Example 14.5] with the $\|\cdot\|_{\mathcal{V}}$-norm gives

$$\operatorname{prox}_{\tau\|\cdot\|_{\mathcal{V}}}(x) = \max\{\|x\|_{\mathcal{V}} - \tau, 0\} \frac{x}{\|x\|_{\mathcal{V}}}, \quad \forall x \in \mathcal{V}_h \setminus \{0\}.$$

Hence,

$$\operatorname{prox}_{\tau g}(\boldsymbol{x}) = \left(\max\{\|x_i\|_{\mathcal{V}} - \tau\lambda w_i, 0\} \frac{x_i}{\|x_i\|_{\mathcal{V}}}\right)_{i=1}^N, \quad \boldsymbol{x} = (x_i)_{i=1}^N \in \mathcal{V}_h^N \setminus \{\boldsymbol{0}\}.$$

With this in hand, we are now ready to define the primal-dual iteration for (4.6). As we see later, the analysis of convergence for the primal-dual iteration is given in terms of the *ergodic* sequence

$$\bar{\boldsymbol{c}}^{(n)} = \frac{1}{n} \sum_{i=1}^{n} \boldsymbol{c}^{(i)}, \quad n = 1, 2, \ldots,$$

where $\boldsymbol{c}^{(i)} \in \mathcal{V}_h^N$ is the primal variable obtained at the ith step of the iteration. Hence, we now include the computation of these sequences in the primal-dual iteration for the weighted SR-LASSO problem (4.6), and take this as the output. The resulting procedure is described in Algorithm 1.

Algorithm 2: `primal-dual-wSRLASSO-C` – the primal-dual iteration for the weighted SR-LASSO problem (4.8)

inputs : measurement matrix $\boldsymbol{A} \in \mathbb{C}^{m\times N}$, measurements $\boldsymbol{B} \in \mathbb{C}^{m\times K}$, positive weights $\boldsymbol{w} = (w_i)_{i=1}^N$, Gram matrix $\boldsymbol{G} \in \mathbb{C}^{K\times K}$, parameter $\lambda > 0$, stepsizes $\tau, \sigma > 0$, maximum number of iterations $T \geq 1$, initial values $\boldsymbol{C}^{(0)} \in \mathbb{C}^{N\times K}$, $\boldsymbol{\Xi}^{(0)} \in \mathbb{C}^{m\times K}$

output : $\overline{\boldsymbol{C}} =$ `primal-dual-wSRLASSO-C`$(\boldsymbol{A}, \boldsymbol{b}, \boldsymbol{w}, \boldsymbol{G}, \lambda, \tau, \sigma, T, \boldsymbol{C}^{(0)}, \boldsymbol{\Xi}^{(0)})$, an approximate minimizer of (4.8)

initialize: $\overline{\boldsymbol{C}}^{(0)} = \boldsymbol{0} \in \mathbb{C}^{N\times K}$

for $n = 0, 1, \ldots, T-1$ **do**
  $\boldsymbol{P} = (p_{ik})_{j,k=1}^{N,K} = \boldsymbol{C}^{(n)} - \tau \boldsymbol{A}^* \boldsymbol{\Xi}^{(n)}$
  for $i = 1, \ldots, N$ **do**
    $\boldsymbol{p}_i = (p_{ik})_{k=1}^K$
    $\big(c_{ik}^{(n+1)}\big)_{k=1}^K = \max\{\|\boldsymbol{G}^{1/2}\boldsymbol{p}_i\|_2 - \tau\lambda w_i, 0\} \frac{\boldsymbol{p}_i}{\|\boldsymbol{G}^{1/2}\boldsymbol{p}_i\|_2}$
  end
  $\boldsymbol{C}^{(n+1)} = \big(c_{ik}^{(n+1)}\big)_{i,k=1}^{N,K}$
  $\boldsymbol{Q} = \boldsymbol{\Xi}^{(n)} + \sigma \boldsymbol{A}(2\boldsymbol{C}^{(n+1)} - \boldsymbol{C}^{(n)}) - \sigma \boldsymbol{B}$
  $\boldsymbol{\Xi}^{(n+1)} = \min\Big\{1, \frac{1}{\|\boldsymbol{Q}\boldsymbol{G}^{1/2}\|_{2,2}}\Big\} \boldsymbol{Q}$
  $\overline{\boldsymbol{C}}^{(n+1)} = \frac{n}{n+1}\overline{\boldsymbol{C}}^{(n)} + \frac{1}{n+1}\boldsymbol{C}^{(n+1)}$
end
$\overline{\boldsymbol{C}} = \overline{\boldsymbol{C}}^{(T)}$

Having done this, we next adapt Algorithm 1 in the way mentioned previously to obtain an algorithm for (4.8). This is given in Algorithm 2.

Remark 4.2. Note that even though the square-root matrix $\boldsymbol{G}^{1/2}$ is used in Algorithm 2, this matrix does not need to be computed. Indeed,

$$\|\boldsymbol{G}^{1/2}\boldsymbol{d}\|_2 = \sqrt{\boldsymbol{d}^*\boldsymbol{G}\boldsymbol{d}}, \quad \boldsymbol{d} \in \mathbb{C}^K,$$

and for a matrix $\boldsymbol{C} \in \mathbb{C}^{N\times K}$, we have

$$\|\boldsymbol{C}\boldsymbol{G}^{1/2}\|_{2,2} = \sqrt{\sum_{i=1}^N \|\boldsymbol{G}^{1/2}\boldsymbol{c}_i\|_2^2} = \sqrt{\sum_{i=1}^N \boldsymbol{c}_i^*\boldsymbol{G}\boldsymbol{c}_i},$$

where $\boldsymbol{c}_i \in \mathbb{C}^K$ is the ith row of $\boldsymbol{C}$. In particular, computing $\|\boldsymbol{G}^{1/2}\boldsymbol{d}\|$ involves

at most $c(F(\boldsymbol{G}) + K)$ arithmetic operations, and computing $\|\boldsymbol{C}\boldsymbol{G}^{1/2}\|_{2,2}$ involves $cm(F(\boldsymbol{G}) + K)$ arithmetic operations, for some universal constant $c > 0$.

To conclude this section, we now state and prove a lemma on the computational cost of Algorithm 2. This will be used later when proving the main theorems.

Lemma 4.3 (Computational cost of Algorithm 2). *The computational cost of Algorithm 2 is bounded by*

$$c \cdot (m \cdot N \cdot K + (m + N) \cdot (F(\boldsymbol{G}) + K)) \cdot T,$$

where $c > 0$ is a universal constant.

Proof. We proceed line-by-line. Line 2 involves a matrix-matrix multiplication and matrix subtraction, for a total of at most

$$c \cdot m \cdot N \cdot K \quad \text{(line 2)}$$

arithmetic operations for some universal constant c. Now consider lines 3–5. By the previous remark, we may calculate $\|\boldsymbol{G}^{1/2}\boldsymbol{p}_i\|_2 = \sqrt{\boldsymbol{p}_i^*\boldsymbol{G}\boldsymbol{p}_i}$ using one multiplication with the matrix $\boldsymbol{G}$, one inner product of vectors of length K and one square root (recall from Definition 3.1 that we count square roots as arithmetic operations). This involves at most $c \cdot (F(\boldsymbol{G}) + K)$ arithmetic operations. Hence, the cost of line 5 is at most

$$c \cdot (F(\boldsymbol{G}) + K) \quad \text{(line 5)},$$

for a possibly different universal constant c. Therefore, the total cost of lines 3–5 is

$$c \cdot (F(\boldsymbol{G}) + K) \cdot N \quad \text{(lines 3–5)}.$$

Line 7 involves no arithmetic operations and line 8 involves at most

$$c \cdot m \cdot N \cdot K \quad \text{(line 8)}$$

operations. Consider line 9. Because of the previous remark, the computation of the term $\|\boldsymbol{Q}\boldsymbol{G}^{1/2}\|_{2,2}$ can be performed in at most $c \cdot m \cdot (F(\boldsymbol{G}) + K)$ operations (since $\boldsymbol{Q}$ is of size $m \times K$). Hence, line 9 involves at most

$$c \cdot m \cdot (F(\boldsymbol{G}) + K) \quad \text{(line 9)}$$

operations. Finally, line 10 involves at most

$$c \cdot N \cdot K \quad \text{(line 10)}$$

operations. After simplifying, we deduce that lines 2–10 involve at most

$$c \cdot (m \cdot N \cdot K + (K + F(\boldsymbol{G})) \cdot (N + m)) \quad \text{(lines 2–10)}$$

operations. The result follows by multiplying this by the number of iterations T. ∎

Algorithm 3: `construct-A` – constructing the measurement matrix (4.3)

inputs : sample points $\boldsymbol{y}_1, \dots, \boldsymbol{y}_m \in \mathcal{U}^d$, finite index set $\Lambda = \{\boldsymbol{\nu}_1, \dots, \boldsymbol{\nu}_N\} \subset \mathcal{F}$

output : $\boldsymbol{A} = \texttt{construct-A}((\boldsymbol{y}_i)_{i=1}^m, \Lambda) \in \mathbb{C}^{m \times N}$, the measurement matrix (4.3)

initialize: $\bar{\boldsymbol{C}}^{(0)} = \boldsymbol{0} \in \mathbb{C}^{N \times K}$

1 $k = \max\{j : (\boldsymbol{\nu}_i)_j \neq 0,\ i = 1, \dots, N,\ j = 1, \dots, d\}$
2 $n = \max\{(\boldsymbol{\nu}_i)_j : i = 1, \dots, N,\ j = 1, \dots, n\}$
3 **for** $i = 1, \dots, m$ **do**
4 Set $\boldsymbol{z} = (z_j)_{j=1}^k = ((\boldsymbol{y}_i)_j)_{j=1}^k$
5 $b_{ij} = \Psi_j(z_i),\ i = 1, \dots, k,\ j = 0, \dots, n,$
6 **for** $j = 1, \dots, N$ **do**
7 $a_{ij} = \prod_{l=1}^n b_{l,(\boldsymbol{\nu}_j)_l}$
8 **end**
9 **end**
10 $\boldsymbol{A} = \frac{1}{\sqrt{m}}(a_{ij})_{i,j=1}^{m,N}$

4.5 The algorithms in Theorems 3.5, 3.8 and 3.11

We are now almost ready to specify the algorithms used in Theorems 3.5, 3.8 and 3.11. Notice that Algorithms 1 and 2 require the measurement matrix $\boldsymbol{A}$ as an input. Hence, we first describe the computation of this matrix for Chebyshev and Legendre polynomials. This is summarized in Algorithm 3. Notice that line 5 of this algorithm involves evaluating the first k one-dimensional Chebyshev or Legendre polynomials. This can be done efficiently via the three-term recurrence relation, as explained in the proof of the following result.

Lemma 4.4 (Computational cost of Algorithm 3). *The computational cost of Algorithm 3 is bounded by*

$$c \cdot m \cdot (n + N) \cdot k,$$

where $c > 0$ is a universal constant and k and n are as in lines 1 and 2 of the algorithm.

Proof. Consider line 5 of the algorithm. Evaluation of the first $k + 1$ Chebyshev or Legendre polynomials can be done via the three-term recurrence relation. In the Chebyshev case, this is given by

$$\Psi_0(z) = 1, \quad \Psi_1(z) = \sqrt{2}z, \quad \Psi_{j+1}(z) = 2z\Psi_j(z) - c_j\Psi_{j-1}(z), \quad j = 1, \dots, k,$$

- Let m, ϵ, n and t be as given in the particular theorem and set:
  - $\Lambda = \Lambda^{\mathrm{HC}}_{n,d}$ (Theorems 3.4 and 3.10) or $\Lambda = \Lambda^{\mathrm{HCI}}_{n}$ (Theorem 3.7),
  - $\lambda = (4\sqrt{m/L})^{-1}$, where $L = L(m,d,\epsilon)$ is as in (3.8),
  - $\tau = \sigma = (\Theta(n,d))^{-\alpha}$, where $\Theta(n,d)$ and α are as in (3.5) and (3.7), respectively,
  - $T = \lceil 2(\Theta(n,d))^{\alpha} t \rceil$.
- Let $\boldsymbol{D} = (d_{ik})_{i,k=1}^{m,K} \in \mathbb{C}^{m\times K}$ and $\boldsymbol{Y} = (\boldsymbol{y}_i)_{i=1}^{m}$ be an input, as in (3.1), and set $\boldsymbol{B} = \frac{1}{\sqrt{m}}\boldsymbol{D}$.
- Compute $\boldsymbol{A} = \texttt{construct-A}(\boldsymbol{Y}, \Lambda)$.
- Let $\boldsymbol{G}$ and $\boldsymbol{w}$ be as in (3.3) and (4.7), respectively.
- Define the output $\overline{\boldsymbol{C}} = \mathcal{A}(\boldsymbol{D})$, where

$$\mathcal{A}(\boldsymbol{D}) = \texttt{primal-dual-wSRLASSO-C}\big(\boldsymbol{A}, \boldsymbol{B}, \boldsymbol{w}, \boldsymbol{G}, \lambda, \tau, \sigma, T, \boldsymbol{0}, \boldsymbol{0}\big)$$

Table 4.2. The algorithms $\mathcal{A} : \mathcal{U}^m \times \mathbb{C}^{m\times K} \to \mathbb{C}^{N\times K}$ used in Theorems 3.5, 3.8 and 3.11.

where $c_j = 1$ if $j \geq 1$ and $1/\sqrt{2}$ otherwise, and in the Legendre case, it is given by

$$\Psi_0(z) = 1, \quad \Psi_1(z) = \sqrt{3}z,$$
$$\Psi_{j+1}(z) = \frac{\sqrt{j+3/2}}{j+1}\left(\frac{2j+1}{\sqrt{j+1/2}} z\Psi_j(z) - \frac{j}{\sqrt{j-1/2}}\Psi_{j-1}(z)\right), \quad j = 2,\ldots,k,$$

(recall that these polynomials are normalized with respect to their respective probability measures). Hence, the computational cost for line 5 is bounded by $c \cdot n \cdot k$. The computational cost for lines 6–8 is precisely $N \cdot (k-1)$. Hence, the computational cost for forming each row of $\boldsymbol{A}$ is bounded by $c \cdot (n \cdot k + N \cdot k)$. The result now follows. ∎

With this in hand, we are now ready to specify the algorithms used in Theorems 3.5, 3.8 and 3.11. These are given in Table 4.2.

4.6 An efficient restarting procedure for the primal-dual iteration and the algorithms used in Theorems 3.6, 3.9 and 3.12

While the primal-dual iteration converges under very general conditions, it typically does so very slowly, with the error in the objective function decreasing like $\mathcal{O}(1/n)$,

Algorithm 4: `primal-dual-rst-wSRLASSO` – the restarted primal-dual iteration for the weighted SR-LASSO problem (4.6)

inputs : measurement matrix $\boldsymbol{A} \in \mathbb{C}^{m\times N}$, measurements $\boldsymbol{b} \in \mathcal{V}_h^N$, positive weights $\boldsymbol{w} = (w_i)_{i=1}^N$, parameter $\lambda > 0$, stepsizes $\tau, \sigma > 0$, number of primal-dual iterations $T \geq 1$, number of restarts $R \geq 1$, tolerance $\zeta' > 0$, scale parameter $0 < r < 1$, constant $s > 0$, initial values $\boldsymbol{c}^{(0)} = \boldsymbol{0} \in \mathcal{V}_h^N$ $\boldsymbol{\xi}^{(0)} = \boldsymbol{0} \in \mathcal{V}_h^m$.

output : $\tilde{\boldsymbol{c}} = \texttt{primal-dual-rst-wSRLASSO}(\boldsymbol{A}, \boldsymbol{b}, \boldsymbol{w}, \lambda, \tau, \sigma, T, R, \zeta', r, s)$, an approximate minimizer of (4.6)

initialize: $\tilde{\boldsymbol{c}}^{(0)} = \boldsymbol{0} \in \mathcal{V}_h^N$, $\varepsilon_0 = \|\boldsymbol{b}\|_{2;\mathcal{V}}$

for $l = 0, \ldots, R-1$ **do**
  $\varepsilon_{l+1} = r(\varepsilon_l + \zeta')$
  $a_l = s\varepsilon_{l+1}$
  $\tilde{\boldsymbol{c}}^{(l+1)} = a_l \cdot \texttt{primal-dual-wSRLASSO}(\boldsymbol{A}, \boldsymbol{b}/a_l, \boldsymbol{w}, \lambda, \tau, \sigma, T, \tilde{\boldsymbol{c}}^{(l)}/a_l, \boldsymbol{0})$
end
$\tilde{\boldsymbol{c}} = \tilde{\boldsymbol{c}}^{(R)}$

where n is the iteration number. To obtain exponential convergence (down to some controlled tolerance) we employ a restarting procedure. This is based on recent work of [47, 48].

Restarting is a general concept in optimization, where the output of an algorithm after a fixed number of steps is then fed into the algorithm as input, after suitably scaling the parameters of the algorithm [122–124]. In the case of the primal-dual iteration for the weighted SR-LASSO problem, this procedure involves three hyperparameters: a *tolerance* $\zeta' > 0$ and *scale* parameters $0 < r < 1$ and $s > 0$. After applying one step of the primal-dual iteration (Algorithms 1 or 2) yielding an output $\boldsymbol{c}^{(1)}$, it then scales this vector and the right-hand side vector $\boldsymbol{b}$ by an exponentially decaying factor a_l (defined in terms of ζ', r and s), before feeding in these values into the primal-dual iteration as input.

We explain the motivations behind the specific form of the restart procedure for the primal-dual iteration later in Section 9.2. For now, we simply state the resulting procedures in the case of the weighted SR-LASSO problems (4.6) and (4.8). These are given in Algorithms 4 and 5, respectively. With these in hand, we can also give the algorithms used in Theorems 3.6, 3.9 and 3.12. See Table 4.3.

Note that these algorithms involve a number $c^\star$, which is a universal constant. It is possible to provide a precise numerical value of this constant by carefully tracking the constants in several of the proof steps. Since doing so is not especially illuminative,

Algorithm 5: `primal-dual-rst-wSRLASSO-C` – the restarted primal-dual iteration for the weighted SR-LASSO problem (4.8)

inputs : measurement matrix $\boldsymbol{A} \in \mathbb{C}^{m\times N}$, measurements $\boldsymbol{B} \in \mathbb{C}^{N\times K}$, positive weights $\boldsymbol{w} = (w_i)_{i=1}^N$, Gram matrix $\boldsymbol{G} \in \mathbb{C}^{K\times K}$, parameter $\lambda > 0$, stepsizes $\tau, \sigma > 0$, number of primal-dual iterations $T \geq 1$, number of restarts $R \geq 1$, tolerance $\zeta' > 0$, scale parameter $0 < r < 1$, constant $s > 0$, initial values $\boldsymbol{C}^{(0)} = \boldsymbol{0} \in \mathbb{C}^{N\times K}$, $\boldsymbol{\Xi}^{(0)} = \boldsymbol{0} \in \mathbb{C}^{m\times K}$

output : $\widetilde{\boldsymbol{C}} =$ `primal-dual-rst-wSRLASSO-C`$(\boldsymbol{A}, \boldsymbol{b}, \boldsymbol{w}, \boldsymbol{G}, \lambda, \tau, \sigma, T, R, \zeta', r, s)$, an approximate minimizer of (4.8)

initialize: $\widetilde{\boldsymbol{C}}^{(0)} = \boldsymbol{0} \in \mathbb{C}^{N\times K}$, $\varepsilon_0 = \|\boldsymbol{B}\boldsymbol{G}^{1/2}\|_{2;2}$

for $l = 0, \ldots, R-1$ **do**
  $\varepsilon_{l+1} = r(\varepsilon_l + \zeta)$
  $a_l = s\varepsilon_{l+1}$
  $\widetilde{\boldsymbol{C}}^{(l+1)} =$
    $a_l \cdot$ `primal-dual-wSRLASSO-C`$(\boldsymbol{A}, \boldsymbol{B}/a_l, \boldsymbol{w}, \boldsymbol{G}, \lambda, \tau, \sigma, T, \widetilde{\boldsymbol{C}}^{(l)}/a_l, \boldsymbol{0})$
end
$\widetilde{\boldsymbol{C}} = \widetilde{\boldsymbol{C}}^{(R)}$

we forgo this additional effort. Instead, we now give a little more detail on this constant.

Remark 4.5. From (10.10) we see that

$$c^\star = 3296\sqrt{c_0},$$

where c_0 is the universal constant that arises in (3.10). As shown in the proof of Theorem 8.2, the constant c_0 needs to be chosen sufficiently large so that the measurement matrix $\boldsymbol{A}$ satisfies the so-called *weighted RIP*. In particular, it is related to the universal constant $c > 0$ defined in Lemma 8.1. See, in particular, (8.2). A numerical value for this constant can indeed be found using results shown in [37]. With this in hand, one can then keep track of the constant c_0 in the proof of Theorem 8.2 to find its numerical value. This discussion also highlights why tracking the value of $c^\star$ is not particularly illuminative. Indeed, it is well known that universal constants appearing in RIP estimates in compressed sensing are generally very pessimistic [8, 13, 61].

- Let m, ϵ, n, t and ζ' be as given in the particular theorem and set:
  - $\Lambda = \Lambda^{\mathsf{HC}}_{n,d}$ (Theorems 3.6 and 3.12) or $\Lambda = \Lambda^{\mathsf{HCI}}_{n}$ (Theorem 3.9),
  - $\lambda = (4\sqrt{m/L})^{-1}$, where $L = L(m, d, \epsilon)$ is as in (3.8),
  - $\tau = \sigma = (\Theta(n, d))^{-\alpha}$, where $\Theta(n, d)$ and α are as in (3.5) and (3.7), respectively,
  - $T = \lceil (\Theta(n, d))^{\alpha} c^{\star} \rceil$, where $c^{\star}$ is a universal constant,
  - $R = t$
  - $r = \mathrm{e}^{-1}$
  - $s = \frac{(\Theta(n,d))^{\alpha} T}{2}$
- Let $\boldsymbol{D} = (d_{ik})_{i,k=1}^{m,K} \in \mathbb{C}^{m \times K}$ and $\boldsymbol{Y} = (\boldsymbol{y}_i)_{i=1}^{m}$ be an input, as in (3.1), and set $\boldsymbol{B} = \frac{1}{\sqrt{m}} \boldsymbol{D}$.
- Compute $\boldsymbol{A} = \texttt{construct-A}(\boldsymbol{Y}, \Lambda)$.
- Let $\boldsymbol{G}$, $\boldsymbol{A}$ and $\boldsymbol{w}$ be as in (4.3), (3.3) and (4.7), respectively.
- Define the output $\widetilde{\boldsymbol{C}} = \mathcal{A}(\boldsymbol{D})$, where

$$\mathcal{A}(\boldsymbol{D}) = \texttt{primal-dual-rst-wSRLASSO-C}(\boldsymbol{A}, \boldsymbol{B}, \boldsymbol{w}, \boldsymbol{G}, \lambda, \tau, \sigma, T, R, \zeta, r, c)$$

Table 4.3. The algorithms $\mathcal{A} : \mathcal{U}^m \times \mathbb{C}^{m \times K} \to \mathbb{C}^{N \times K}$ used in Theorems 3.6, 3.9 and 3.12.