# A FAST SEMIDIRECT LEAST SQUARES ALGORITHM FOR HIERARCHICALLY BLOCK SEPARABLE MATRICES* 

KENNETH L. $\mathrm{HO}^{\dagger}$ AND LESLIE GREENGARD ${ }^{\ddagger}$


#### Abstract

We present a fast algorithm for linear least squares problems governed by hierarchically block separable (HBS) matrices. Such matrices are generally dense but data sparse and can describe many important operators including those derived from asymptotically smooth radial kernels that are not too oscillatory. The algorithm is based on a recursive skeletonization procedure that exposes this sparsity and solves the dense least squares problem as a larger, equality-constrained, sparse one. It relies on a sparse QR factorization coupled with iterative weighted least squares methods. In essence, our scheme consists of a direct component, comprised of matrix compression and factorization, followed by an iterative component to enforce certain equality constraints. At most two iterations are typically required for problems that are not too ill-conditioned. For an $M \times N$ HBS matrix with $M \geq N$ having bounded off-diagonal block rank, the algorithm has optimal $\mathcal{O}(M+N)$ complexity. If the rank increases with the spatial dimension as is common for operators that are singular at the origin, then this becomes $\mathcal{O}(M+N)$ in one dimension, $\mathcal{O}\left(M+N^{3 / 2}\right)$ in two dimensions, and $\mathcal{O}\left(M+N^{2}\right)$ in three dimensions. We illustrate the performance of the method on both overdetermined and underdetermined systems in a variety of settings, with an emphasis on radial basis function approximation and efficient updating and downdating.


Key words. fast algorithms, matrix compression, recursive skeletonization, sparse QR decomposition, weighted least squares, deferred correction, radial basis functions, updating/downdating

AMS subject classifications. 65F05, 65F20, 65F50, 65Y15
DOI. 10.1137/120902677

1. Introduction. The method of least squares is a powerful technique for the approximate solution of overdetermined systems and is often used for data fitting and statistical inference in applied science and engineering. In this paper, we will primarily consider the linear least squares problem

$$
\begin{equation*}
\min _{x}\|A x-b\| \tag{1.1}
\end{equation*}
$$

where $A \in \mathbb{C}^{M \times N}$ is dense and full rank with $M \geq N, x \in \mathbb{C}^{N}, b \in \mathbb{C}^{M}$, and $\|\cdot\|$ is the Euclidean norm. Formally, the solution is given by

$$
\begin{equation*}
x=A^{+} b, \tag{1.2}
\end{equation*}
$$

where $A^{+}$is the Moore-Penrose pseudoinverse of $A$, and can be computed directly via the QR decomposition at a cost of $\mathcal{O}\left(M N^{2}\right)$ operations [8, 42]. This can be prohibitive when $M$ and $N$ are large. If $A$ is structured so as to support fast multiplication, then iterative methods such as LSQR [46] or GMRES [37, 49] present an attractive

[^0]Table 1
Examples of radial kernels $\phi(r)$ admitting HBS representations: $H_{0}^{(1)}$, zeroth order Hankel function of the first kind; $K_{0}$, zeroth order modified Bessel function of the second kind.

| Type | Name | Kernel |  |
| :--- | :--- | :---: | :--- |
| Green's function |  | 2 D | Notes |
|  | Laplace | $\log r$ | $1 / r$ |
|  |  |  |  |  |
|  | Helmholtz | $H_{0}^{(1)}(k r)$ | $e^{\imath k r} / r$ |
| Yukawa | not too large |  |
|  |  | Polyharmonic | $e^{-k r} / r$ |
| Radial basis function | Multiquadric | $r^{2 n} \log r$ | $r^{2 n-1}$ |
|  | Inverse multiquadric | $\sqrt{r^{2}+c^{2}}$ | $c$ not too large |
|  |  | $1 / \sqrt{r^{2}+c^{2}}$ |  |

alternative. However, such solvers still have several key disadvantages when compared with their direct counterparts.
(i) The convergence rate of an iterative solver can depend strongly on the conditioning of the system matrix, which, for least squares problems, can sometimes be very poor. In such cases, the number of iterations required, and hence the computational cost, can be far greater than expected (if the solver succeeds at all). Direct methods, by contrast, are robust in that their performance does not degrade with conditioning. Thus, they are often preferred in situations where reliability is critical.
(ii) Standard iterative schemes are inefficient for multiple right-hand sides. With direct solvers, on the other hand, following an expensive initial factorization, the subsequent cost for each solve is typically much lower (e.g., only $\mathcal{O}(M N)$ work to apply the pseudoinverse given precomputed QR factors). This is especially important in the context of updating and downdating as the least squares problem is modified by adding or deleting data, which can be viewed as low-rank updates of the original system matrix.

In this paper, we present a fast semidirect least squares solver for a class of structured dense matrices called hierarchically block separable (HBS) matrices. Such matrices were introduced by Gillman, Young, and Martinsson [24] and possess a nested low-rank property that enables highly efficient data-sparse representations. The HBS matrix structure is closely related to that of $\mathscr{H}$ - and $\mathscr{H}^{2}$-matrices [32, $33,34,35$ ] and hierarchically semiseparable (HSS) matrices [15, 16, 56], and can be considered a generalization of the matrix features utilized by multilevel summation algorithms like the fast multipole method (FMM) [27, 28]. Many linear operators are of HBS form, notably integral transforms with asymptotically smooth radial kernels. These include those based on the Green's functions of nonoscillatory elliptic partial differential equations [9]. Some examples are shown in Table 1; we highlight, in particular, the Green's functions

$$
\phi_{\Delta}(r)=\frac{1}{r}, \quad \phi_{\Delta^{2}}(r)=r,
$$

for the Laplace and biharmonic equations, respectively, in three dimensions, and their regularizations, the inverse multiquadric and multiquadric kernels

$$
\phi_{\mathrm{IMQ}}(r)=\frac{1}{\sqrt{r^{2}+c^{2}}}, \quad \phi_{\mathrm{MQ}}(r)=\sqrt{r^{2}+c^{2}}
$$

respectively (for $c$ not too large). The latter are well known within the radial basis function (RBF) community and have been used to successfully model smooth surfaces

TABLE 2
Asymptotic complexities for the least squares solver when applied to the operators in Table 1 on data embedded in a d-dimensional domain: $M$ and $N$ are the matrix dimensions; $M \geq N$.

| $d$ | Precomputation | Solution |
| :--- | :--- | :--- |
| 1 | $\mathcal{O}(M+N)$ | $\mathcal{O}(M+N)$ |
| 2 | $\mathcal{O}\left(M+N^{3 / 2}\right)$ | $\mathcal{O}(M+N \log N)$ |
| 3 | $\mathcal{O}\left(M+N^{2}\right)$ | $\mathcal{O}\left(M+N^{4 / 3}\right)$ |

[13, 36]. Also of note is the two-dimensional (2D) biharmonic Green's function, the so-called thin plate spline

$$
\begin{equation*}
\phi_{\mathrm{TPS}}(r)=r^{2} \log r, \tag{1.3}
\end{equation*}
$$

which minimizes a physical bending energy [22]. For an overview of RBFs, see [10, 47].
Remark. Although we focus in this paper on dense matrices, many sparse matrices, e.g., those resulting from local finite-difference-type discretizations, are also of HBS form.

Previous work on HBS matrices exploited their structure to build fast direct solvers for the square $M=N$ case $[24,39,44]$ (similar methods are available for other structured formats). Here, we extend the approach of [39] to the rectangular $M \geq$ $N$ case. Our algorithm relies on the multilevel compression and sparsity-revealing embedding of [39], and recasts the (unconstrained) dense least squares problem (1.1) as a larger, equality-constrained, sparse one. This is solved via a sparse QR factorization coupled with iterative weighted least squares methods. For the former, we use the SuiteSparseQR package [20] by Davis, while for the latter, we employ the iteration of Barlow and Vemulapati [5], which has been shown to require at most two steps for problems that are not too ill-conditioned. Thus, our solver is a semidirect method where the iteration often converges extremely quickly; in such cases, it retains all of the advantages of traditional direct solvers.

It is useful to divide our algorithm into two phases: a direct precomputation phase, comprising matrix compression and factorization, followed by an iterative solution phase using the precomputed QR factors. Clearly, for a given matrix, only the solution phase must be executed for each additional right-hand side. Table 2 lists asymptotic complexities for both phases when applied to the operators in Table 1 on data embedded in a $d$-dimensional domain for $d=1,2$, or 3 . Although the estimates generally worsen as $d$ increases, the solver achieves optimal $\mathcal{O}(M+N)$ complexity for both phases in any dimension in the special case that the source (column) and target (row) data are separated (i.e., the domain and range of the continuous operator are disjoint). This may have applications, for example, in partial charge fitting in computational chemistry [6, 23].

Remark. The increase in cost with $d$ is due to the singular nature of the kernels in Table 1 at the origin, which leads to growth of the off-diagonal block ranks defining the HBS form (section 5). If the data are separated or if the kernel itself is smooth, then this rank growth does not occur. In this paper, we will not specifically address this latter setting, viewing it instead as a special case of the one-dimensional problem.

Our methods can also be generalized to underdetermined systems $(M<N)$ when seeking the minimum-norm solution in $L^{2}$, i.e., the equality-constrained least squares problem

$$
\begin{equation*}
\min _{A x=b}\|x\| \tag{1.4}
\end{equation*}
$$

provided that the solution, which is also given by (1.2), is not too ill-conditioned with respect to $A$.

Fast direct least squares algorithms have been developed in other structured matrix contexts as well, in particular within the $\mathscr{H}$ - and HSS matrix frameworks using various structured orthogonal transformation schemes [7, 14, 21]. Our approach, however, is quite different and explicitly leverages the sparse representation of HBS matrices and the associated sparse matrix technology (e.g., the state-of-the-art software package SuiteSparseQR). This has the possible advantage of producing an algorithm that is easier to implement, extend, and optimize. For example, although we consider here only the standard Moore-Penrose systems (1.1) and (1.4), it is immediate that our techniques can be applied to general equality-constrained least squares problems with any combination of the system and constraint matrices being HBS (but with a possible increase in cost). For related work on other structured matrices including those of Toeplitz, Hankel, and Cauchy type, see, for instance, [30, 41, 52, 57] and references therein.

The remainder of this paper is organized as follows. In the next two sections, we collect and review certain mathematical preliminaries on HBS matrices (section 2) and equality-constrained least squares problems (section 3). In section 4, we describe our fast semidirect algorithm for both overdetermined and underdetermined systems. Complexity estimates are given in section 5 , while section 6 discusses efficient updating and downdating in the context of our solver. Numerical results for a variety of radial kernels are reported in section 7. Finally, in section 8, we summarize our findings and end with some generalizations and concluding remarks.
2. HBS matrices. In this section, we define the HBS matrix property and discuss algorithms to compress such matrices and to sparsify linear systems governed by them. We will mainly follow the treatment of [39], extended to rectangular matrices in the natural way.

Let $A \in \mathbb{C}^{M \times N}$ be a matrix viewed with $p \times p$ blocks, with the $i$ th row and column blocks having dimensions $m_{i}, n_{i}>0$, respectively, for $i=1, \ldots, p$.

Definition 2.1 (block separable matrix [24]). The matrix $A$ is block separable if each off-diagonal submatrix $A_{i j}$ can be decomposed as the product of three low-rank matrices:

$$
\begin{equation*}
A_{i j}=L_{i} S_{i j} R_{j} \quad \text { for } i \neq j \tag{2.1}
\end{equation*}
$$

where $L_{i} \in \mathbb{C}^{m_{i} \times k_{i}^{\mathrm{r}}}, S_{i j} \in \mathbb{C}^{k_{i}^{\mathrm{r}} \times k_{j}^{\mathrm{c}}}$, and $R_{j} \in \mathbb{C}^{k_{j}^{\mathrm{c}} \times n_{j}}$, with (ideally) $k_{i}^{\mathrm{r}} \ll m_{i}$ and $k_{j}^{\mathrm{c}} \ll n_{j}$, for $i, j=1, \ldots, p$.

Clearly, the block separability condition (2.1) is equivalent to requiring that the off-diagonal block rows and columns have low rank (Figure 1). Observe that if $A$ is block separable, then it can be written as

$$
\begin{equation*}
A=D+L S R \tag{2.2}
\end{equation*}
$$



Fig. 1. A block separable matrix has low-rank off-diagonal block rows and columns (black); its diagonal blocks (white) can in general be full rank.


Fig. 2. An example of a tree on the row and column index sets with depth $\lambda=3$. The root (node 1) contains all indices, which are hierarchically partitioned among its children.
where $D=\operatorname{diag}\left(A_{i i}\right), L=\operatorname{diag}\left(L_{i}\right), R=\operatorname{diag}\left(R_{i}\right)$, and $S=\left(S_{i j}\right)$ is dense with $S_{i i}=0$.

Let us now define a tree structure on the row and column indices $I=\{1, \ldots, M\}$ and $J=\{1, \ldots, N\}$, respectively, as follows. Associate with the root of the tree the entire index sets $I$ and $J$. If a given subdivision criterion is satisfied (e.g., based on the sizes $|I|$ and $|J|$ ), partition the root node into a set of children, each associated with a subset of $I$ and $J$ such that they together span all the sets. Repeat this process for each new node to be subdivided, partitioning its row and column indices among its children. In other words, if we label each tree node with an integer $i$ and denote its row and column index sets by $I_{i}$ and $J_{i}$, respectively, then

$$
I_{i}=\bigcup_{j \in \operatorname{child}(i)} I_{j}, \quad J_{i}=\bigcup_{j \in \operatorname{child}(i)} J_{j},
$$

where child $(i)$ gives the set of node indices belonging to the children of node $i$. Furthermore, we also label the levels of the tree, starting with level 0 for the root at the coarsest level to level $\lambda$ for the leaves at the finest level; see Figure 2 for an example.

Remark. Although we require that the number of row and column partitions be the same, we do not impose that $\left|I_{i}\right|=\left|J_{i}\right|$ for any node $i$. Indeed, it is possible for one of these sets to be empty.

Evidently, the tree defines a hierarchy among row and column index sets, each level of which specifies a block partition of the matrix $A$.

Definition 2.2 (HBS matrix [24]). The matrix $A$ is HBS if it is block separable at each level of the tree hierarchy.

HBS matrices arise in many applications, for example, when discretizing the kernels in Table 1 (up to a specified numerical precision), with row and column indices partitioned according to an octree-type ordering on the corresponding data [27, 28, 33, 39], which recursively groups together points that are geometrically collocated [50].
2.1. Multilevel matrix compression. We now review algorithms [24, 39, 44] for computing the low-rank matrices in (2.1) characterizing the HBS form. Our primary tool for this task is the interpolative decompositon (ID) [17].

Definition 2.3 (ID). An ID of a matrix $A \in \mathbb{C}^{m \times n}$ with rank $k$ is a factorization $A=B P$, where $B \in \mathbb{C}^{m \times k}$ consists of a subset of the columns of $A$ and $P \in \mathbb{C}^{k \times n}$ contains the $k \times k$ identity matrix. We call $B$ and $P$ the skeleton and interpolation matrices, respectively.

As stated, the ID clearly compresses the column space of $A$, but we can just as well compress the row space by applying the ID to $A^{\top}$. Efficient algorithms for adaptively computing an ID to any specified precision are available [17, 43, 55], i.e., the required rank $k$ is an output of the ID.


Fig. 3. Schematic of recursive skeletonization, comprising alternating steps of compression and regrouping via ascension of the index tree. The diagonal blocks (white and gray) are extracted at each level; they are shown here only to illustrate the regrouping process.

DEFINITION 2.4 (row and column skeletons). The row indices corresponding to the retained rows in the ID are called the row or incoming skeletons; the column indices corresponding to the retained columns are called the column or outgoing skeletons.

A multilevel algorithm for the compression of HBS matrices then follows. For simplicity, we describe the procedure for matrices with a uniform tree depth (i.e., all leaves are at level $\lambda$ ), with the understanding that it extends easily to the adaptive case. The following scheme [24, 39, 44] is known as recursive skeletonization (RS) (Figure 3).

1. Starting at the leaves of the tree, extract the diagonal blocks and compress the off-diagonal block rows and columns using the ID to a specified precision $\epsilon>0$ as follows. For each block $i=1, \ldots, p$, compress the row space of the $i$ th off-diagonal block row and call $L_{i}$ the corresponding row interpolation matrix. Similarly, for each block $j$, compress the column space of the $j$ th off-diagonal block column and call $R_{j}$ the corresponding column interpolation matrix. Let $S$ be the "skeleton" submatrix of $A$, with each off-diagional block $S_{i j}$ for $i \neq j$ defined by the row and column skeletons associated with $L_{i}$ and $R_{j}$, respectively.
2. Since the off-diagonal blocks $S_{i j}$ are submatrices of the corresponding $A_{i j}$, the compressed matrix $S$ is HBS and so can itself be compressed in the same way. Thus, move up one level in the tree, regroup the matrix blocks accordingly, and repeat.

The result is a telescoping matrix representation of the form

$$
\begin{equation*}
A \approx D^{(\lambda)}+L^{(\lambda)}\left[\cdots D^{(2)}+L^{(2)}\left(D^{(1)}+L^{(1)} D^{(0)} R^{(1)}\right) R^{(2)} \cdots\right] R^{(\lambda)} \tag{2.3}
\end{equation*}
$$

cf. (2.2), where the superscript indexes the tree level $l=0, \ldots, \lambda$, that is accurate to relative precision approximately $\epsilon$. The algorithm is automatically adaptive in the sense that the compression is more efficient if lower precision is required [24, 39, 44].

Remark. For the kernels in Table 1, which obey some form of Green's theorem (at least approximately), it is possible to substantially accelerate the preceding algorithm by using a "proxy" surface to capture all far-field interactions (see also section 5). The key idea is that any such interaction can be represented in terms of some equivalent
density on an appropriate local bounding surface, which can be chosen so that it requires only a constant number of points to discretize, irrespective of the actual number of points in the far field or their detailed structure. This observation hence replaces each global compression step with an entirely local one; see [17, 24, 26, 39, $44,58]$ for details.
2.2. Structured sparse embedding. For $M=N$, the decomposition (2.3) enables a highly structured sparse representation $[15,39]$ of the linear system $A x=b$ as

$$
\left[\begin{array}{ccccccc}
D^{(\lambda)} & L^{(\lambda)} & & & & &  \tag{2.4}\\
R^{(\lambda)} & & -I & & & & \\
& -I & D^{(\lambda-1)} & \ddots & & & \\
& & \ddots & \ddots & -I & & \\
& & & -I & D^{(1)} & L^{(1)} & \\
& & & & R^{(1)} & & -I \\
& & & & & -I & D^{(0)}
\end{array}\right]\left[\begin{array}{c}
x^{(\lambda)} \\
y^{(\lambda)} \\
\vdots \\
\vdots \\
x^{(1)} \\
y^{(1)} \\
x^{(0)}
\end{array}\right]=\left[\begin{array}{c}
b \\
0 \\
\vdots \\
\vdots \\
0 \\
0 \\
0
\end{array}\right]
$$

under the identifications

$$
\begin{align*}
x^{(\lambda)} & =x, & & \text { for } l=\lambda-1, \ldots, 0  \tag{2.5a}\\
y^{(1)} & =D^{(0)} x^{(0)}, & y^{(l+1)} & =R^{(l+1)} x^{(l+1)} \tag{2.5b}
\end{align*}
$$

This expanded embedding clearly exposes the sparsity of HBS matrix equations and permits the immediate application of existing fast sparse solvers (such as UMFPACK [19] as in [39]).

If $M \geq N$, however, then we have to deal with the overdetermined problem (1.1), and (2.4) must be interpreted somewhat more carefully. In particular, the identities (2.5) still hold, so only the first block row of (2.4) is to be solved in the least squares sense. Thus, denoting the first block row of the sparse matrix in (2.4) by $\mathbf{E}$ and the remainder (i.e., its last $2 \lambda$ block rows) by $\mathbf{C}$, and defining $\mathbf{x}=\left(x^{(\lambda)}, y^{(\lambda)}, \ldots, x^{(1)}, y^{(1)}, x^{(0)}\right)^{\top}$, the analogue of (2.4) for (1.1) is the equalityconstrained least squares problem

$$
\begin{equation*}
\min _{\mathbf{C}=\mathbf{0}}\|\mathbf{E x}-b\| \tag{2.6}
\end{equation*}
$$

where both $\mathbf{E}$ and $\mathbf{C}$ are sparse. It is easy to see that $\mathbf{C}$ has full row rank.
Similarly, if $M<N$ and we seek to solve the underdetermined system (1.4), then the corresponding problem is

$$
\begin{equation*}
\min _{\mathbf{M} \mathbf{x}=\mathbf{b}_{1}}\left\|\mathbf{I}_{1} \mathbf{x}\right\| \tag{2.7}
\end{equation*}
$$

where

$$
\mathbf{M}=\left[\begin{array}{c}
\mathbf{E}  \tag{2.8}\\
\mathbf{C}
\end{array}\right], \quad \mathbf{b}_{1}=\left[\begin{array}{c}
b \\
0 \\
\vdots \\
0
\end{array}\right], \quad \mathbf{I}_{1}=\left[\begin{array}{llll}
I & 0 & \cdots & 0
\end{array}\right]
$$

i.e., $\mathbf{M}$ is the entire sparse matrix on the left-hand side of (2.4), which also has full row rank; $\mathbf{b}_{1}$ is the right-hand side of (2.4); and $\mathbf{I}_{1}$ is an operator that picks out the first block row of the vector on which it acts.
3. Equality-constrained least squares. We now turn to the solution of linear least squares problems with linear equality constraints, with special attention to the case that both governing matrices are sparse as in (2.6) and (2.7). For consistency with the linear algebra literature, we adopt the notation of Barlow et al. [3, 4, 5], which unfortunately conflicts somewhat with our previous definitions; the following notation is thus meant to pertain only to this section.

Hence, consider the problem

$$
\begin{equation*}
\min _{C x=g}\|E x-f\| \tag{3.1}
\end{equation*}
$$

where $E \in \mathbb{C}^{m \times n}$ and $C \in \mathbb{C}^{p \times n}$, with

$$
\operatorname{rank}(C)=p, \quad \operatorname{rank}\left(\left[\begin{array}{l}
E \\
C
\end{array}\right]\right)=n
$$

so that the solution is unique. Classical reduction schemes for solving (3.1), such as the direct elimination and nullspace methods, require matrix products that can destroy the sparsity of the resulting reduced, unconstrained systems $[8,42]$.
3.1. Weighted least squares. An attractive alternative when both $E$ and $C$ are sparse is the method of weighting, which recasts (3.1) in the unconstrained form

$$
\begin{equation*}
\min _{x}\|A(\tau) x-b(\tau)\| \tag{3.2}
\end{equation*}
$$

where

$$
A(\tau)=\left[\begin{array}{c}
E \\
\tau C
\end{array}\right], \quad b(\tau)=\left[\begin{array}{c}
f \\
\tau g
\end{array}\right]
$$

for $\tau$ a suitably large weight. Clearly, as $\tau \rightarrow \infty$, the solution of (3.2) approaches that of (3.1). The advantage, of course, is that (3.2) can be solved using standard sparse techniques; this point of view is elaborated in $[4,53]$.

However, the choice of an appropriate weight can be a delicate matter: if $\tau$ is too small, then (3.2) approximates (3.1) poorly, while if $\tau$ is too large, then (3.2) can be ill-conditioned. An intuitive approach is to start with a small weight, then carry out some type of iterative refinement, effectively increasing the weight with each step. Such a scheme was first proposed by Van Loan [53], then further studied and improved by Barlow [3] and Barlow and Vemulapati [5]; we summarize their results in the next section.
3.2. Iterative reweighting by deferred correction. In [5], Barlow and Vemulapati presented the following deferred correction procedure for the solution of the equality-constrained least squares problem (3.1) via the successive solution of the weighted problem (3.2) with a fixed weight $\tau$.

1. Find

$$
x^{(0)}=\underset{x}{\arg \min }\|A(\tau) x-b(\tau)\|
$$

and set

$$
r^{(0)}=f-E x^{(0)}, \quad w^{(0)}=g-C x^{(0)}, \quad \lambda^{(0)}=\tau^{2} w^{(0)}
$$

2. For $k=0,1,2, \ldots$ until convergence, find

$$
\Delta x^{(k)}=\underset{x}{\arg \min }\left\|A(\tau) x-b^{(k)}(\tau)\right\|, \quad b^{(k)}(\tau)=\left[\begin{array}{c}
r^{(k)} \\
\tau w^{(k)}+\tau^{-1} \lambda^{(k)}
\end{array}\right]
$$

and update

$$
\begin{aligned}
x^{(k+1)} & =x^{(k)}+\Delta x^{(k)} \\
r^{(k+1)} & =r^{(k)}-E \Delta x^{(k)} \\
w^{(k+1)} & =w^{(k)}-C \Delta x^{(k)} \\
\lambda^{(k+1)} & =\lambda^{(k)}+\tau^{2} w^{(k+1)} .
\end{aligned}
$$

Terminate when the constraint residual $\left\|w^{(k+1)}\right\|$ is small.
Since $\tau$ is fixed, a single precomputed QR factorization of $A(\tau)$ can be used for all iterations. This algorithm is a slight modification of that employed by Van Loan [53] and has been shown to converge to the correct solution for $\tau$ appropriately chosen, provided that (3.1) is not too ill-conditioned [3,5,53]. In particular, if implemented in double precision, then for $\tau=\epsilon_{\text {mach }}^{-1 / 3} \sim 1.7 \times 10^{5}$, where $\epsilon_{\text {mach }}$ is the machine epsilon, Barlow and Vemulapati [5] showed that their algorithm requires no more than two iterations. Thus, for a broad class of problems for which it is reasonable to expect an accurate answer, the above scheme often converges extremely rapidly (and can, in some sense, even be considered a direct method, which can be made explicit by running the iteration for exactly two steps).

Remark. Although Barlow and Vemulapati [5] considered (3.1) only over the reals, there is no inherent difficulty in extending their solution procedure to the complex case.

Remark. It was recently pointed out to us by Eduardo Corona (personal communication, Aug. 2013) that deferred correction can be applied to ill-conditioned systems as well, provided that $\tau$ is changed appropriately. The relevant analysis can be found in [3, Corollary 3.1], which suggests choosing $\tau=\epsilon_{\text {mach }}^{-1 / 3}[\kappa(M)]^{1 / 3}$, where $\kappa(M)=\|M\|\left\|M^{+}\right\|$is the condition number of the "stacked" matrix

$$
M=\left[\begin{array}{l}
E  \tag{3.3}\\
C
\end{array}\right]
$$

Of course, setting $\tau$ now requires an estimate of $\kappa(M)$, which may not always be available; for this reason, we have elected simply to present our algorithm with $\tau=$ $\epsilon_{\text {mach }}^{-1 / 3}$.
4. Algorithm. We are now in a position to describe our fast semidirect method for solving the overdetermined and underdetermined systems (1.1) and (1.4), respectively, when $A$ is HBS. Let $\epsilon>0$ be a specified numerical precision and set $\tau=\epsilon_{\text {mach }}^{-1 / 3} \sim 1.7 \times 10^{5}$; we assume that all calculations are performed in double precision.
4.1. Overdetermined systems. Let $A \in \mathbb{C}^{M \times N}$ be $\operatorname{HBS}$ with $M \geq N$. Our algorithm proceeds in two phases. First, for the precomputation phase:

1. Compress $A$ to precision $\epsilon$ using RS [39];
2. compute a sparse QR factorization of the weighted sparse matrix

$$
\mathbf{A}=\left[\begin{array}{c}
\mathbf{E} \\
\tau \mathbf{C}
\end{array}\right]
$$

where $\mathbf{E}$ and $\mathbf{C}$ are as defined in section 2.2.

This is followed by the solution phase, which, for a given right-hand side $b$, produces an approximate solution (1.2) by using the precomputed QR factors to solve the equality-constrained least squares embedding (2.6) via deferred correction [5]. Clearly, for a fixed matrix $A$, only the solution phase must be performed for each additional right-hand side. Therefore, the cost of the precomputation phase is amortized over all such solves.

Remark. The algorithm can also easily accommodate various modifications of the standard system (1.1), e.g., the problem

$$
\begin{equation*}
\min _{x}\|A x-b\|^{2}+\mu^{2}\|x\|^{2} \tag{4.1}
\end{equation*}
$$

with Tikhonov regularization, which can be solved using

$$
\mathbf{A}=\left[\begin{array}{c}
\mathbf{E} \\
\mu \mathbf{I}_{1} \\
\tau \mathbf{C}
\end{array}\right], \quad \mathbf{b}=\left[\begin{array}{l}
b \\
0 \\
0
\end{array}\right]
$$

in the weighted formulation (3.2).
4.2. Underdetermined systems. Now let $A \in \mathbb{C}^{M \times N}$ be HBS with $M<N$. Then (1.4) can be solved using the same algorithm as above but with

$$
\mathbf{A}=\left[\begin{array}{c}
\mathbf{I}_{1} \\
\tau \mathbf{M}
\end{array}\right], \quad \mathbf{b}=\left[\begin{array}{c}
0 \\
\mathbf{b}_{1}
\end{array}\right]
$$

4.3. Error analysis. We now give an informal discussion of the accuracy of our method. Assume that $M \geq N$ and let $\kappa(A)=\|A\|\left\|A^{+}\right\|=\sigma_{1}(A) / \sigma_{N}(A)$ be the condition number of $A$, where $\sigma_{1}(A) \geq \cdots \geq \sigma_{N}(A) \geq \sigma_{N+1}(A)=\cdots=\sigma_{M}(A)=0$ are the singular values of $A$. We first estimate the condition number of the compressed matrix $A_{\epsilon}$ in (2.3).

Proposition 4.1. Let $A_{\epsilon}=A+E$ with $\|E\|=\mathcal{O}(\epsilon\|A\|)$. If $\epsilon \kappa(A) \ll 1$, then $\kappa\left(A_{\epsilon}\right)=\mathcal{O}(\kappa(A))$.

Proof. By Weyl's inequality,

$$
\left|\sigma_{i}\left(A_{\epsilon}\right)-\sigma_{i}(A)\right| \leq\|E\|, \quad i=1, \ldots, N
$$

so $\sigma_{i}\left(A_{\epsilon}\right)=\sigma_{i}(A)+e_{i}$ with $\left|e_{i}\right|=\mathcal{O}(\epsilon\|A\|)$. Therefore,

$$
\kappa\left(A_{\epsilon}\right)=\frac{\sigma_{1}\left(A_{\epsilon}\right)}{\sigma_{N}\left(A_{\epsilon}\right)}=\frac{\sigma_{1}(A)+e_{1}}{\sigma_{N}(A)+e_{N}}=\kappa(A)\left[\frac{1+e_{1} / \sigma_{1}(A)}{1+e_{N} / \sigma_{N}(A)}\right]
$$

where

$$
\frac{\left|e_{1}\right|}{\sigma_{1}(A)}=\frac{\left|e_{1}\right|}{\|A\|}=\mathcal{O}(\epsilon), \quad \frac{\left|e_{N}\right|}{\sigma_{N}(A)}=e_{N}\left\|A^{+}\right\|=\mathcal{O}(\epsilon \kappa(A))
$$

so $\kappa\left(A_{\epsilon}\right)=\mathcal{O}(\kappa(A))$.
In other words, if $A$ is not too ill-conditioned, then neither is $A_{\epsilon}$. But the convergence of deferred correction depends on the conditioning of the stacked matrix (3.3), i.e., the sparse embedding $\mathbf{M}$ of $A_{\epsilon}$ in (2.8). Although we have not explicitly studied the spectral properties of $\mathbf{M}$, numerical estimates suggest that $\kappa(\mathbf{M})=\mathcal{O}\left(\kappa\left(A_{\epsilon}\right)\right)$.

Some evidence for this can be seen in the square, single-level case (2.2), for which the analogue of $\mathbf{M}$ is

$$
\left[\begin{array}{ccc}
D & L & \\
R & & -I \\
& -I & S
\end{array}\right]=\left[\begin{array}{ccc}
I & -L S & -L \\
& I & \\
& & I
\end{array}\right]\left[\begin{array}{ccc}
A_{\epsilon} & & \\
& & -I \\
& -I & S
\end{array}\right]\left[\begin{array}{ccc}
I & & \\
-S R & I & \\
-R & & I
\end{array}\right]
$$

with inverse

$$
\left[\begin{array}{ccc}
D & L & \\
R & & -I \\
& -I & S
\end{array}\right]^{-1}=\left[\begin{array}{ccc}
I & L S & L \\
& I & \\
& & I
\end{array}\right]\left[\begin{array}{ccc}
A_{\epsilon}^{-1} & & \\
& S & -I \\
& -I &
\end{array}\right]\left[\begin{array}{ccc}
I & & \\
S R & I & \\
R & & I
\end{array}\right]
$$

Assume without loss of generality that $\|A\|=1$. Since $S$ is a submatrix of $A$, this also implies that $\|S\| \leq 1$. Furthermore, it is typically the case that $\|L\|$ and $\|R\|$ are not too large since they come from the ID [17]. It then follows that $\|\mathbf{M}\|=\mathcal{O}\left(\left\|A_{\epsilon}\right\|\right)$ and $\left\|\mathbf{M}^{-1}\right\|=\mathcal{O}\left(\left\|A_{\epsilon}^{-1}\right\|\right)$; thus, $\kappa(\mathbf{M})=\mathcal{O}\left(\kappa\left(A_{\epsilon}\right)\right)=\mathcal{O}(\kappa(A))$.

This argument can be extended to the multilevel setting (2.4), but only for square matrices. Still, in practice, we observed that the claim seems to hold also for rectangular matrices, so that if $\kappa(A)$ is not too large then neither is $\kappa(\mathbf{M})$ and we may expect deferred correction to succeed.

Even if the iteration converges to the exact solution, however, there is still an error arising from the use of $A_{\epsilon}$ in place of $A$ itself. The extent of this error is governed by standard perturbation theory. The following is a restatement of Theorem 20.1 in [38], originally due to Wedin, specialized to the current setting.

TheOrem 4.2 (Wedin). Let $A \in \mathbb{C}^{M \times N}$ be full-rank with $M \geq N$, and let

$$
x=\underset{y}{\arg \min }\|A y-b\|, \quad x_{\epsilon}=\underset{y}{\arg \min }\left\|A_{\epsilon} y-b\right\|
$$

be the solutions of the corresponding overdetermined systems, with residuals

$$
r=b-A x, \quad r_{\epsilon}=b-A_{\epsilon} x_{\epsilon},
$$

respectively. If $\epsilon \kappa(A)<1$, then

$$
\frac{\left\|x-x_{\epsilon}\right\|}{\|x\|} \leq \frac{\epsilon \kappa(A)}{1-\epsilon \kappa(A)}\left[1+\kappa(A) \frac{\|r\|}{\|A\|\|x\|}\right], \quad \frac{\left\|r-r_{\epsilon}\right\|}{\|b\|} \leq 2 \epsilon \kappa(A)
$$

A somewhat simpler bound holds for underdetermined systems [38, Theorem 21.1].
Theorem 4.3 (Demmel and Higham). Let $A \in \mathbb{C}^{M \times N}$ be full rank with $M<N$, and let $x$ and $x_{\epsilon}$ be the minimum-norm solutions to the underdetermined systems $A x=b$ and $A_{\epsilon} x_{\epsilon}=b$, respectively, for $b \neq 0$. If $\left\|A^{+}\left(A-A_{\epsilon}\right)\right\|<1$, then

$$
\frac{\left\|x-x_{\epsilon}\right\|}{\|x\|} \leq 2 \epsilon \kappa(A)
$$

A more rigorous analysis is not yet available, but we note that our numerical results (section 7) indicate that the algorithm is accurate and stable.
5. Complexity analysis. In this section, we analyze the complexity of our solver for a representative example: the HBS matrix $A$ defined by a kernel from Table 1, acting on source and target data distributed uniformly over the same $d$ dimensional domain (but at different densities). We follow the approach of [39].

Sort both sets of data together in one hyperoctree (the multidimensional generalization of an octree; cf. [50]) as outlined in section 2, subdividing each node until it contains no more than a set number of combined row and column indices, i.e., $\left|I_{i}\right|+\left|J_{i}\right|=\mathcal{O}(1)$ for each leaf $i$. For each tree level $l=0, \ldots, \lambda$, let $p_{l}$ denote the number of matrix blocks; $m_{l}$ and $n_{l}$, the row and column block sizes, respectively, in the compressed representation (2.3), assumed equal across all blocks for simplicity; and $k_{l}$, the skeleton block size, which is clearly of the same order for both rows and columns as it depends only on $\min \left\{\mathcal{O}\left(m_{l}\right), \mathcal{O}\left(n_{l}\right)\right\}$ (this can be made precise by explicitly considering proxy compression, which produces interaction matrices of size $\left.\mathcal{O}\left(m_{l}\right) \times \mathcal{O}\left(n_{l}\right)\right)$. Note that $m_{l}$ and $n_{l}$ are not the row and column block sizes in the tree; they are the result of hierarchically "pulling up" skeletons during the compression process; see (iv) below. Moreover, since $M \neq N$ in general, we can define an additional level parameter $\lambda^{\prime} \leq \lambda$ corresponding to the depth of the tree constructed via the same process on only the smaller of the source or target data, e.g., on only the source data if $M \geq N$. For the remainder of this discussion, we assume that $M \geq N$. Analogous results can be recovered for $M<N$ simply by switching the roles of $M$ and $N$ in what follows. We start with some useful observations:
(i) By construction, $p_{\lambda}\left(m_{\lambda}+n_{\lambda}\right)=M+N \sim M$, where $m_{\lambda}, n_{\lambda}=\mathcal{O}(1)$, so $p_{\lambda} \sim M$. Similarly, $p_{\lambda^{\prime}} \sim N$;
(ii) each subdivision increases the number of blocks by a factor of roughly $2^{d}$, so $p_{l+1} \sim 2^{d} p_{l}$. In particular, $p_{0}=1$, so $\lambda \sim(1 / d) \log M$ and $\lambda^{\prime} \sim(1 / d) \log N$;
(iii) for $l=\lambda^{\prime}+1, \ldots, \lambda, k_{l}=\mathcal{O}(1)$ since $\min \left\{\mathcal{O}\left(m_{l}\right), \mathcal{O}\left(n_{l}\right)\right\}=\mathcal{O}(1)$, while for $l=0, \ldots, \lambda^{\prime}$, it can be shown [39] that

$$
k_{l} \sim \begin{cases}l & \text { if } d=1  \tag{5.1}\\ 2^{(d-1) l} & \text { if } d>1\end{cases}
$$

(iv) the total number of row and column indices at level $l<\lambda$ is equal to the total number of skeletons at level $l+1$, i.e., $p_{l} m_{l}=p_{l} n_{l}=p_{l+1} k_{l+1}$, so $m_{l}, n_{l} \sim k_{l+1}$.
5.1. Matrix compression. From [17, 43, 55], the cost of computing a rank- $k$ ID of an $m \times n$ matrix is $\mathcal{O}(k m n)$. If proxy compression is used, then $m \sim m_{l}$ for a block at level $l$. Therefore, the total cost of matrix compression is

$$
T_{\mathrm{cm}} \sim \sum_{l=0}^{\lambda} p_{l} k_{l} m_{l} n_{l} \sim \sum_{l=0}^{\lambda} p_{l} k_{l}^{3} .
$$

We break this into two sums, one over $l=\lambda^{\prime}+1, \ldots, \lambda$ and another over $l=0, \ldots, \lambda^{\prime}$, with estimates

$$
\sum_{l=\lambda^{\prime}+1}^{\lambda} p_{l} k_{l}^{3} \sim M, \quad \sum_{l=0}^{\lambda^{\prime}} p_{l} k_{l}^{3} \sim \begin{cases}N & \text { if } d=1, \\ N^{3(1-1 / d)} & \text { if } d>1,\end{cases}
$$

respectively. Hence,

$$
T_{\mathrm{cm}} \sim \begin{cases}M+N & \text { if } d=1,  \tag{5.2}\\ M+N^{3(1-1 / d)} & \text { if } d>1 .\end{cases}
$$

5.2. Compressed QR factorization. We consider the QR decomposition of a block tridiagonal matrix with the same sparsity structure as that of $\mathbf{M}$ in (2.4), computed using Householder reflections. This clearly encompasses the factorization


Fig. 4. Sparsity structure of $Q R$ factors for the structured embedding (2.4) with $M=N$, where the orthogonal matrix $\mathbf{Q}$ is given in terms of the elementary Householder reflectors $\mathbf{q}_{i}$.


Fig. 5. Sparsity structure of the triangular $Q R$ factor $\mathbf{R}$ for the structured embedding (2.4) in the $M>N$ (left) and $M<N$ (right) cases.
of $\mathbf{A}=\mathbf{Q R}$ for both overdetermined and underdetermined systems. We begin by studying the square $M=N$ case, for which it is immediate that $\mathbf{R}$ is block upper bidiagonal (Figure 4), so the cost of Householder triangularization is

$$
T_{\mathrm{qr}} \sim \sum_{l=0}^{\lambda} p_{l} m_{l} n_{l}^{2} \sim \sum_{l=0}^{\lambda} p_{l} k_{l}^{3} \sim \begin{cases}M+N & \text { if } d=1,  \tag{5.3}\\ M+N^{3(1-1 / d)} & \text { if } d>1,\end{cases}
$$

following the structure of $\mathbf{R}$. The $M>N$ and $M<N$ cases are easily analyzed by noting that only the blocks at level $\lambda$ are rectangular (in the asymptotic sense), so any nonzero propagation during triangularization is limited and both essentially reduce to the square case above (Figure 5).

The complexities (5.2) and (5.3) of compression and factorization, respectively, together constitute the cost of the precomputation phase.
5.3. Compressed pseudoinverse application. We now examine the cost of applying the pseudoinverse to solve the weighted least squares problem (3.2) using the precomputed QR factors. For this, we suppose that the solution is determined via the equation $\mathbf{R x}=\mathbf{Q}^{*} \mathbf{b}$, which requires one application of $\mathbf{Q}^{*}$, assumed to be performed using elementary Householder transformations, and one backsolve with $\mathbf{R}$, whose cost is clearly on the same order as multiplying by $\mathbf{R}$. Then from the arguments above, it is evident that both operations have complexity

$$
T_{\mathrm{sv}} \sim \sum_{l=0}^{\lambda} p_{l} m_{l} n_{l} \sim \sum_{l=0}^{\lambda} p_{l} k_{l}^{2} \sim \begin{cases}M+N & \text { if } d=1 \\ M+N \log N & \text { if } d=2 \\ M+N^{2(1-1 / d)} & \text { if } d>2\end{cases}
$$

Since the total number of such problems to be solved is constant for each outer problem (2.6), assuming fast convergence (section 3.2), this is also the complexity of the solution phase. As with classical matrix factorizations, the prefactor for $T_{\mathrm{sv}}$ is typically far smaller than that for $T_{\mathrm{cm}}$ or $T_{\mathrm{qr}}$.

Remark. One can also use the seminormal equations $\mathbf{R}^{*} \mathbf{R} x=\mathbf{A}^{*} \mathbf{b}$, which do not require the orthogonal matrix $\mathbf{Q}[8]$. However, one step of iterative refinement is necessary for stability, so the total cost is three applications of $\mathbf{A}$ or $\mathbf{A}^{*}$ (one each for the original and refinement solves, plus another to compute the residual), and four solves with $\mathbf{R}$ or $\mathbf{R}^{*}$. In practice, we found this approach to be slower than that involving $\mathbf{Q}$ by a factor of about four.
5.4. Some remarks. For all complexities above, the constants implicit in the estimates are of the form $\mathcal{O}\left(2^{d} \log ^{\alpha} \epsilon\right)$ for modest $\alpha$, i.e., they are exponential in the dimension and polylogarithmic in the precision [27, 28].

In the special case that the source and target data are separated, $k_{l}=\mathcal{O}(1)$ for all $l$, so $T_{\mathrm{cm}}, T_{\mathrm{qr}}$, and $T_{\mathrm{sv}}$ all have optimal complexity $\mathcal{O}(M+N)$ in any dimension. This describes, for example, the fitting of atomic partial charges to reproduce electrostatic potential values on "shells" around a molecule [6, 23], the computation of equivalent densities in the kernel-independent FMM [58], and even the calculation of the ID in RS [24, 39, 44], which requires a least squares solve [17].
6. Updating and downdating. We now discuss an important feature of our solver: its capacity for efficient updating and downdating in response to dynamically changing data. Our methods are based on the augmented system approach of [26], extended to the least squares setting, and exploit the ability to rapidly apply $A^{+}$ via the solution phase of our algorithm, which we hereafter take as a computational primitive. Thus, suppose that we are given some base linear system (1.1), focusing for simplicity on the overdetermined case, for which we have precomputed a compressed QR factorization of $A$. We consider the addition and deletion of both rows and columns, corresponding to the modification of observations and regression variables, respectively. Furthermore, we assume that such modifications are small, in particular so that the system remains overdetermined, and accommodate each case within the framework of the general equality-constrained least squares problem

$$
\begin{equation*}
\min _{\mathbf{C x}=\mathbf{g}}\|\mathbf{E x}-\mathbf{f}\| . \tag{6.1}
\end{equation*}
$$

6.1. Adding and deleting rows. To add $p_{\mathrm{r}}$ rows to the matrix $A=\left(a_{i j}\right)$, and correspondingly to the vector $b=\left(b_{1}, \ldots, b_{M}\right)^{\top}$, in (1.1), we simply use (6.1) with

$$
\mathbf{E}=\left[\begin{array}{l}
A  \tag{6.2}\\
C_{+}^{\mathrm{r}}
\end{array}\right], \quad \mathbf{C}=0, \quad \mathbf{x}=x, \quad \mathbf{f}=\left[\begin{array}{l}
b \\
b_{+}
\end{array}\right], \quad \mathbf{g}=0,
$$

where $C_{+}^{\mathrm{r}} \in \mathbb{C}^{p_{\mathrm{r}} \times N}$ describes the influence of the variables $x$ on the new data $b_{+}$. To delete $q_{\mathrm{r}}$ rows with indices $k_{1}, \ldots, k_{q_{\mathrm{r}}}$, we add $q_{\mathrm{r}}$ degrees of freedom to those rows to be deleted in order to enforce strict agreement with those observations as follows:

$$
\mathbf{E}=\left[\begin{array}{ll}
A & B_{-}^{\mathrm{r}}
\end{array}\right], \quad \mathbf{C}=\left[\begin{array}{ll}
C_{-}^{\mathrm{r}} & I
\end{array}\right], \quad \mathbf{x}=\left[\begin{array}{l}
x \\
x_{-}^{\mathrm{r}}
\end{array}\right], \quad \mathbf{f}=b, \quad \mathbf{g}=d,
$$

where $B_{-}^{\mathrm{r}}=\left(\delta_{k_{i} j}\right) \in \mathbb{C}^{M \times q_{\mathrm{r}}}, C_{-}^{\mathrm{r}}=\left(a_{k_{i} j}\right) \in \mathbb{C}^{q_{\mathrm{r}} \times N}$, and $d=\left(b_{k_{1}}, \ldots, b_{k_{q_{\mathrm{r}}}}\right)^{\mathrm{T}}$; here,

$$
\delta_{i j}= \begin{cases}1 & \text { if } i=j, \\ 0 & \text { if } i \neq j\end{cases}
$$

is the Kronecker delta. The simultaneous addition and deletion of rows can be achieved via a straightforward combination of the above:

$$
\mathbf{E}=\left[\begin{array}{ll}
A & B_{-}^{\mathrm{r}} \\
C_{+}^{\mathrm{r}} &
\end{array}\right], \quad \mathbf{C}=\left[\begin{array}{ll}
C_{-}^{\mathrm{r}} & I
\end{array}\right], \quad \mathbf{x}=\left[\begin{array}{l}
x \\
x_{-}^{\mathrm{r}}
\end{array}\right], \quad \mathbf{f}=\left[\begin{array}{l}
b \\
b_{+}
\end{array}\right], \quad \mathbf{g}=d .
$$

6.2. Adding and deleting columns. To add $p_{c}$ columns to $A$ and hence to the vector $x=\left(x_{1}, \ldots, x_{N}\right)^{\top}$, we let

$$
\mathbf{E}=\left[\begin{array}{ll}
A & B_{+}^{\mathrm{c}}
\end{array}\right], \quad \mathbf{C}=0, \quad \mathbf{x}=\left[\begin{array}{l}
x \\
x_{+}^{c}
\end{array}\right], \quad \mathbf{f}=b, \quad \mathbf{g}=0
$$

where $B_{+}^{\mathrm{c}} \in \mathbb{C}^{M \times p_{c}}$ describes the influence of the new variables $x_{+}^{\mathrm{c}}$ on the data. To delete $q_{\mathrm{c}}$ columns with indices $l_{1}, \ldots, l_{q_{\mathrm{c}}}$, we add $q_{\mathrm{c}}$ "antivariables" annihilating their effects:

$$
\mathbf{E}=\left[\begin{array}{ll}
A & B_{-}^{\mathrm{c}}
\end{array}\right], \quad \mathbf{C}=\left[\begin{array}{ll}
C_{-}^{\mathrm{c}} & I
\end{array}\right], \quad \mathbf{x}=\left[\begin{array}{l}
x \\
x_{-}^{\mathrm{c}}
\end{array}\right], \quad \mathbf{f}=b, \quad \mathbf{g}=0,
$$

where $B_{-}^{\mathrm{c}}=\left(a_{i l_{j}}\right) \in \mathbb{C}^{M \times q_{\mathrm{c}}}$ and $C_{-}^{\mathrm{c}}=\left(\delta_{i l_{j}}\right) \in \mathbb{C}^{q_{\mathrm{c}} \times N}$. Finally, to add and delete columns simultaneously, we take

$$
\mathbf{E}=\left[\begin{array}{lll}
A & B_{+}^{\mathrm{c}} & B_{-}^{\mathrm{c}}
\end{array}\right], \quad \mathbf{C}=\left[\begin{array}{lll}
C_{-}^{\mathrm{c}} & 0 & I
\end{array}\right], \quad \mathbf{x}=\left[\begin{array}{c}
x \\
x_{+}^{\mathrm{c}} \\
x_{-}^{\mathrm{c}}
\end{array}\right], \quad \mathbf{f}=b, \quad \mathbf{g}=0 .
$$

6.3. Simultaneous modification of rows and columns. The general case of modifying both rows and columns can be treated using (6.1) with

$$
\mathbf{E}=\left[\begin{array}{cccc}
A & B_{+}^{\mathrm{c}} & B_{-}^{\mathrm{r}} & B_{-}^{\mathrm{c}} \\
C_{+}^{\mathrm{r}} & D_{+} & 0 & D_{-}
\end{array}\right], \quad \mathbf{C}=\left[\begin{array}{cccc}
C_{-}^{\mathrm{r}} & 0 & I & 0 \\
C_{-}^{\mathrm{c}} & 0 & 0 & I
\end{array}\right]
$$

and

$$
\mathbf{x}=\left[\begin{array}{l}
x \\
x_{+}^{c} \\
x_{-}^{r} \\
x_{-}^{c}
\end{array}\right], \quad \mathbf{f}=\left[\begin{array}{l}
b \\
b_{+}
\end{array}\right], \quad \mathbf{g}=\left[\begin{array}{l}
d \\
0
\end{array}\right],
$$

where $D_{+} \in \mathbb{C}^{p_{\mathrm{r}} \times p_{c}}$ describes the influence of $x_{+}^{\mathrm{c}}$ on $b_{+}, D_{-}=\left(c_{i l_{j}}\right) \in \mathbb{C}^{p_{\mathrm{r}} \times q_{\mathrm{c}}}$ for $C_{+}^{\mathrm{r}}=\left(c_{i j}\right)$ accounts for the effect of column deletion on $b_{+}$(alternatively, one can zero out the relevant columns in $C_{+}^{\mathrm{r}}$ ), and all other quantities are as defined previously.
6.4. Solution methods. The augmented system (6.1) can be solved by deferred correction [5], where the matrix to be considered at each step is

$$
\mathbf{A}=\left[\begin{array}{c}
\mathbf{E} \\
\tau \mathbf{C}
\end{array}\right] \equiv\left[\begin{array}{ll}
A & B \\
C & D
\end{array}\right] \in \mathbb{C}^{(M+m) \times(N+n)}
$$

for $B \in \mathbb{C}^{M \times n}, C \in \mathbb{C}^{m \times N}$, and $D \in \mathbb{C}^{m \times n}$, where $m=p_{\mathrm{r}}+q_{\mathrm{r}}+q_{\mathrm{c}}$ and $n=$ $p_{\mathrm{c}}+q_{\mathrm{r}}+q_{\mathrm{c}}$. If $m$ and $n$ are small, then an efficient approach is to compute $\mathbf{A}^{+}$ by using $A^{+}$in various block pseudoinverse formulas (see, e.g., [11]) or by invoking Greville's method [12, 29], which can construct $\mathbf{A}^{+}$via a sequence of rank-one updates to $A^{+}$. Alternatively, one can appeal to iterative methods like GMRES [49], using $A^{+}$as a preconditioner. In this approach, instead of solving

$$
\min _{\mathbf{x}}\|\mathbf{A x}-\mathbf{b}\|, \quad \mathbf{b}=\left[\begin{array}{c}
\mathbf{f} \\
\tau \mathbf{g}
\end{array}\right],
$$

we consider instead, say, the left preconditioned system

$$
\min _{\mathrm{x}}\|\mathbf{B A x}-\mathbf{B b}\|
$$

for an appropriate choice of the preconditioner $\mathbf{B} \in \mathbb{C}^{(N+n) \times(M+m)}$. Hayami, Yin, and Ito $[37]$ showed that GMRES converges provided that range $(\mathbf{A})=\operatorname{range}\left(\mathbf{B}^{*}\right)$ and $\operatorname{range}\left(\mathbf{A}^{*}\right)=\operatorname{range}(\mathbf{B})$. Therefore, suitable choices of $\mathbf{B}$ include, e.g.,

$$
\left[\begin{array}{cc}
A^{+} & C^{*} \\
B^{*} & D^{*}
\end{array}\right], \quad\left[\begin{array}{ll}
A^{+} & C^{*} \\
B^{*} & D^{+}
\end{array}\right]
$$

(the latter if $D$ is not rank deficient), for which only one application of $A^{+}$is required per iteration. Such methods also have the possible advantage of being more flexible and robust. If the total number of iterations is small, the cost of updating is therefore only $\mathcal{O}\left(T_{\mathrm{sv}}\right)$ instead of $\mathcal{O}\left(T_{\mathrm{cm}}+T_{\mathrm{qr}}\right)$ for computing the QR factorization anew, which is typically much larger (section 5 ).

Remark. If $M+m>N+n$, then it is more efficient to solve the left preconditioned system of dimension $N+n$. Similarly, if $M+m<N+n$, then it is more efficient to solve the right preconditioned system of dimension $M+m$.
7. Numerical results. In this section, we report some numerical results for our fast semidirect solver, compared against LAPACK/ATLAS $[1,54]$ and an accelerated GMRES solver $[37,49]$ using an FMM-type scheme. We considered problems in both two and three dimensions. All matrices were block partitioned using quadtrees in two dimensions and octrees in three dimensions, uniformly subdivided until all leaf nodes contained no more than a fixed number of combined rows and columns (cf. sections 2.1 and 5), while adaptively pruning all empty nodes during the refinement process. The RS algorithm was implemented in Fortran and employed as described in [39]. Sparse QR factorizations were computed using SuiteSparseQR [20] through a MATLAB R2012b interface, keeping all orthogonal matrices in compact Householder form. The deferred correction procedure [5] was implemented in MATLAB. All calculations were performed in double-precision real arithmetic on a single 3.10 GHz processor with 4 GB of RAM.

For each case, where appropriate, we report the following data:

- $M, N$ : the uncompressed row and column dimensions, respectively;
- $K_{\mathrm{r}}, K_{\mathrm{c}}$ : the final row and column skeleton dimensions, respectively;
- $T_{\mathrm{cm}}$ : the matrix compression time ( s );
- $T_{\mathrm{qr}}$ : the sparse QR factorization time (s);
- $T_{\mathrm{sv}}$ : the pseudoinverse application time ( s );
- $n_{\text {iter }}$ : the number of iterations required for deferred correction;
- $E$ : the relative error $\left\|x-x_{\epsilon}\right\| /\|x\|$ with respect to the solution $x$ produced by LAPACK/ATLAS (if the problem is small enough) or FMM/GMRES;
- $R$ : the relative residual $\left\|A x_{\epsilon}-b\right\| /\|b\|$ with respect to the true operator.

Note that $E$ and $R$ are not the true relative error and residual but nevertheless provide a useful measure of accuracy.
7.1. Laplace's equation. For benchmarking purposes, we first applied our method to Laplace's equation

$$
\Delta u=0 \quad \text { in } \Omega \in \mathbb{R}^{d}, \quad u=f \quad \text { on } \partial \Omega
$$

in a simply connected interior domain with Dirichlet boundary conditions, which can be solved by writing the solution in the form of a double-layer potential

$$
u(\vec{x})=\int_{\partial \Omega} \frac{\partial G}{\partial \nu_{\vec{y}}}(\|\vec{x}-\vec{y}\|) \sigma(\vec{y}) d \vec{y} \quad \text { for } \vec{x} \in \Omega
$$



Fig. 6. CPU times for solving Laplace's equation in two and three dimensions using $L A P A C K / A T L A S(L P), F M M / G M R E S$ (FMM), and $R S$ as a function of the system size $N$. For $L P$ and $R S$, the computation is split into two parts: precomputation (pc), for LP consisting of matrix formation and factorization, and for $R S$ of matrix compression and factorization; and system solution (sv), consisting of matrix (pseudo) inverse application via precomputed $Q R$ factors. The precision of FMM and $R S$ was set at $\epsilon=10^{-9}$ in two dimensions and $10^{-6}$ in three dimensions. Dotted lines indicate extrapolated data; for $R S$ in three dimensions, only factorization and inversion (as executed through MATLAB) are extrapolated.
where

$$
G(r)= \begin{cases}-1 /(2 \pi) \log r & \text { if } d=2 \\ 1 /(4 \pi r) & \text { if } d=3\end{cases}
$$

is the free-space Green's function, $\nu_{\vec{y}}$ is the unit outer normal at $\vec{y} \in \partial \Omega$, and $\sigma$ is an unknown surface density. Letting $\vec{x}$ approach the boundary, standard results from potential theory [31] yield the second-kind Fredholm boundary integral equation

$$
\begin{equation*}
-\frac{1}{2} \sigma(\vec{x})+\int_{\partial \Omega} \frac{\partial G}{\partial \nu_{\vec{y}}}(\|\vec{x}-\vec{y}\|) \sigma(\vec{y}) d \vec{y}=f(\vec{x}) \tag{7.1}
\end{equation*}
$$

for $\sigma$, assuming that $\partial \Omega$ is smooth. This is not a least squares problem, but it allows us to compare the performance of the sparse QR approach with our previous sparse LU results [39]. Of course, since the system (7.1) is square, the corresponding sparse embedding (2.4) can be solved without iteration.

In two dimensions, we took as the problem geometry a 2:1 ellipse, discretized via the trapezoidal rule, while in three dimensions, we used the unit sphere, discretized as a collection of flat triangles with piecewise constant densities. We also compared our algorithm against an FMM-accelerated GMRES solver driven by the open-source FMMLIB software package [25], which is a fairly efficient implementation (but not optimized using the plane wave representations of [28]). Timing results for each case are shown in Figure 6, with detailed data for the RS scheme given in Tables 3 and 4. The precision was set to $\epsilon=10^{-9}$ in two dimensions and $10^{-6}$ in three dimensions.

It is evident that our method scales as predicted, with precomputation and solution complexities of $\mathcal{O}(N)$ in two dimensions $(d=1)$, and $\mathcal{O}\left(N^{3 / 2}\right)$ and $\mathcal{O}(N \log N)$, respectively, in three dimensions $(d=2)$. In two dimensions, both phases are very fast, easily beating the uncompressed LAPACK/ATLAS solver in both time $\left(\mathcal{O}\left(N^{3}\right)\right.$ and $\mathcal{O}\left(N^{2}\right)$ for precomputation and solution, respectively) and memory, and coming quite close to the FMM/GMRES solver as well. The same is essentially true in three

Table 3
Numerical results for solving Laplace's equation in two dimensions at precision $\epsilon=10^{-9}$.

| $N$ | $K_{\mathrm{r}}$ | $K_{\mathrm{c}}$ | $T_{\mathrm{cm}}$ | $T_{\mathrm{qr}}$ | $T_{\mathrm{sv}}$ | $E$ |
| ---: | :---: | :---: | :---: | :---: | :---: | :---: |
| 1024 | 30 | 30 | $3.1 \mathrm{E}-2$ | $9.4 \mathrm{E}-2$ | $5.8 \mathrm{E}-3$ | $1.1 \mathrm{E}-9$ |
| 2048 | 29 | 30 | $6.5 \mathrm{E}-2$ | $5.8 \mathrm{E}-2$ | $1.8 \mathrm{E}-2$ | $4.5 \mathrm{E}-9$ |
| 4096 | 30 | 30 | $1.3 \mathrm{E}-1$ | $1.2 \mathrm{E}-1$ | $3.7 \mathrm{E}-2$ | $1.5 \mathrm{E}-8$ |
| 8192 | 30 | 31 | $2.6 \mathrm{E}-1$ | $2.6 \mathrm{E}-1$ | $5.7 \mathrm{E}-2$ | $1.4 \mathrm{E}-8$ |
| 16384 | 31 | 31 | $5.2 \mathrm{E}-1$ | $6.0 \mathrm{E}-1$ | $9.2 \mathrm{E}-2$ | $1.7 \mathrm{E}-8$ |
| 32768 | 30 | 30 | $1.1 \mathrm{E}+0$ | $1.0 \mathrm{E}+0$ | $2.0 \mathrm{E}-1$ | $1.2 \mathrm{E}-8$ |
| 65536 | 30 | 30 | $2.1 \mathrm{E}+0$ | $2.0 \mathrm{E}+0$ | $3.4 \mathrm{E}-1$ | $1.6 \mathrm{E}-8$ |
| 131072 | 29 | 29 | $4.1 \mathrm{E}+0$ | $4.7 \mathrm{E}+0$ | $6.8 \mathrm{E}-1$ | $2.2 \mathrm{E}-8$ |

Table 4
Numerical results for solving Laplace's equation in three dimensions at precision $\epsilon=10^{-6}$. Parentheses denote extrapolated values.

| $N$ | $K_{\mathrm{r}}$ | $K_{\mathrm{c}}$ | $T_{\mathrm{cm}}$ | $T_{\mathrm{qr}}$ | $T_{\mathrm{sv}}$ | $E$ |
| ---: | ---: | ---: | :---: | :---: | :---: | :---: |
| 320 | 320 | 320 | $2.3 \mathrm{E}-1$ | $5.1 \mathrm{E}-2$ | $4.5 \mathrm{E}-3$ | $1.5 \mathrm{E}-10$ |
| 720 | 628 | 669 | $1.1 \mathrm{E}+0$ | $1.6 \mathrm{E}-1$ | $9.3 \mathrm{E}-3$ | $5.1 \mathrm{E}-07$ |
| 1280 | 890 | 913 | $3.7 \mathrm{E}+0$ | $5.0 \mathrm{E}-1$ | $3.2 \mathrm{E}-2$ | $1.0 \mathrm{E}-06$ |
| 2880 | 1393 | 1400 | $1.7 \mathrm{E}+1$ | $1.9 \mathrm{E}+0$ | $9.1 \mathrm{E}-2$ | $1.2 \mathrm{E}-06$ |
| 5120 | 1886 | 1850 | $4.7 \mathrm{E}+1$ | $6.0 \mathrm{E}+0$ | $1.6 \mathrm{E}-1$ | $2.2 \mathrm{E}-06$ |
| 11520 | 2750 | 2754 | $1.4 \mathrm{E}+2$ | $(1.9 \mathrm{E}+1)$ | $(3.9 \mathrm{E}-1)$ |  |
| 20480 | 3592 | 3551 | $3.1 \mathrm{E}+2$ | $(4.6 \mathrm{E}+1)$ | $(7.4 \mathrm{E}-1)$ |  |

dimensions over the range of problem sizes tested, though it should be emphasized that FMM/GMRES has optimal $\mathcal{O}(N)$ complexity and so should prevail asymptotically. However, as observed previously [26, 39, 44], the solve time using recursive skeletonization following precomputation (comprising one application of $\mathbf{Q}^{*}$ and one backsolve with $\mathbf{R}$ ) is much faster than an individual FMM/GMRES solve: e.g., in two dimensions at $N=131072, T_{\mathrm{FMM}}=3.9 \mathrm{~s}$, while $T_{\mathrm{sv}}=0.7 \mathrm{~s}$. This is significantly slower when compared with our UMFPACK-based sparse LU solver ( $T_{\text {sv }} \sim 0.1 \mathrm{~s}$ ) [39]. The difference may be due, in part, both to a higher constant inherent in the QR approach and to the overhead from interfacing with MATLAB. Unfortunately, we were unable to perform the sparse QR factorizations in-core for the three-dimensional case beyond $N \sim 10^{4}$; the corresponding data are extrapolated from the results of section 5 .
7.2. Least squares fitting of thin plate splines. We next turned to an overdetermined problem involving 2D function interpolation using thin plate splines; see (1.3). More specifically, we sought to compute the coefficients $a_{j}$ of the interpolant

$$
g(\vec{x})=\sum_{j=1}^{N} a_{j} \phi_{\operatorname{TPS}}\left(\left\|\vec{x}-\vec{c}_{j}\right\|\right)
$$

that best matches a given function

$$
f(x, y)=\sin (4 \pi x)+\cos (2 \pi y) \sin (3 \pi x y) \quad \text { for } \vec{x} \equiv(x, y)
$$

in the least squares sense on some set of randomly chosen targets $\vec{x}_{i} \in[0,1]^{2}$ for $i=1, \ldots, M$. The points $\vec{c}_{j}$ for $j=1, \ldots, N$ denote the centers of the splines and lie on a uniform tensor product grid on $[0,1]^{2}$. This is an inconsistent linear system. Since the problem is somewhat ill-conditioned, we also add Tikhonov regularization with regularization parameter $\mu=0.1$ as indicated in (4.1).

Timing results for various $M$ and $N$ at a fixed ratio of $M / N=4$ with $\epsilon=10^{-6}$ are shown in Figure 7, with detailed data in Table 5. The results are in line with our


FIG. 7. $C P U$ times for overdetermined thin plate spline fitting in two dimensions at precision $\epsilon=10^{-6}$ using LAPACK/ATLAS, FMM/GMRES, and $R S$ as a function of the system row dimension $M$, with the column dimension fixed proportionally at $N=M / 4$; all other notation as in Figure 6.

Table 5
Numerical results for overdetermined thin plate spline fitting in two dimensions at precision $\epsilon=10^{-6}$.

| $M$ | $N$ | $K_{\mathrm{r}}$ | $K_{\mathrm{C}}$ | $T_{\mathrm{cm}}$ | $T_{\mathrm{qr}}$ | $T_{\mathrm{sv}}$ | $n_{\text {iter }}$ | $E$ | $R$ |
| ---: | ---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: |
| 1024 | 256 | 174 | 148 | $7.7 \mathrm{E}-2$ | $1.7 \mathrm{E}-1$ | $5.3 \mathrm{E}-2$ | 1 | $4.1 \mathrm{E}-5$ | $1.4 \mathrm{E}-1$ |
| 4096 | 1024 | 260 | 247 | $5.7 \mathrm{E}-1$ | $3.1 \mathrm{E}-1$ | $1.6 \mathrm{E}-1$ | 1 | $8.3 \mathrm{E}-5$ | $4.4 \mathrm{E}-2$ |
| 16384 | 4096 | 399 | 391 | $3.1 \mathrm{E}+0$ | $1.5 \mathrm{E}+0$ | $1.0 \mathrm{E}+0$ | 1 | $3.9 \mathrm{E}-4$ | $1.6 \mathrm{E}-2$ |
| 65536 | 16384 | 564 | 574 | $1.5 \mathrm{E}+1$ | $7.0 \mathrm{E}+0$ | $4.7 \mathrm{E}+0$ | 1 | $2.0 \mathrm{E}-6$ | $6.7 \mathrm{E}-3$ |

complexity estimates of $\mathcal{O}\left(M+N^{3 / 2}\right)$ and $\mathcal{O}(M+N \log N)$ for precomputation and solution, respectively. This compares favorably with the uncompressed complexities of $\mathcal{O}\left(M N^{2}\right)$ and $\mathcal{O}(M N)$, respectively, for LAPACK/ATLAS. We also tested an iterative GMRES solver, which required from 33 up to 130 iterations on the largest problem considered using $A^{\top}$ as a left preconditioner. Direct timings are unavailable since we did not have an FMM to apply the thin plate spline kernel (or its transpose). The results are instead estimated using RS and an established benchmark FMM rate of about $10^{5}$ points per second in two dimensions. It is immediate that our fast solver outperforms FMM/GMRES due to the rapidly growing iteration count. Note also the convergence in the relative residual of roughly second order. In all cases, the deferred correction procedure converged with just one iteration.
7.3. Underdetermined charge fitting. We then considered an underdetermined problem: seeking a minimum-norm charge distribution in two dimensions. The setup is as follows. Let $\vec{x}_{j}$ for $j=1, \ldots, N$ be uniformly spaced points on the unit circle, each associated with a random charge $q_{j}$. We measure their induced field

$$
f(\vec{x} ; q)=-\frac{1}{2 \pi} \sum_{j=1}^{N} q_{j} \log \left\|\vec{x}-\vec{x}_{j}\right\|
$$

on a uniformly sampled outer ring of radius $1+\delta$, and compute an equivalent set of charges $\tilde{q}_{j}$ with minimal Euclidean norm reproducing those measurements. Here, we set $\delta=10^{-4}$ and sampled at $M=N / 8$ observation points over a range of $N$.

Timing results at precision $\epsilon=10^{-9}$ are shown in Figure 8, with detailed data given in Table 6. Since the source and target points are separated (by an annular


| $\bigcirc$ LP pc | $\square-\square$ LP sv | $\diamond$ FMM | $\bullet$ RS pc | ■- RS sv |
| :---: | :---: | :---: | :---: | :---: |

Fig. 8. CPU times for underdetermined charge fitting in two dimensions at precision $\epsilon=10^{-9}$ using LAPACK/ATLAS, FMM/GMRES, and RS as a function of the system column dimension $N$, with the row dimension fixed proportionally at $M=N / 8$; all other notation as in Figure 6.

TABLE 6
Numerical results for underdetermined charge fitting in two dimensions at precision $\epsilon=10^{-9}$.

| $M$ | $N$ | $K_{\mathrm{r}}$ | $K_{\mathrm{C}}$ | $T_{\mathrm{cm}}$ | $T_{\mathrm{qr}}$ | $T_{\mathrm{sv}}$ | $n_{\text {iter }}$ | $E$ | $R$ |
| ---: | ---: | ---: | ---: | :---: | :---: | :---: | :---: | :---: | :---: |
| 128 | 1024 | 69 | 72 | $1.7 \mathrm{E}-2$ | $2.1 \mathrm{E}-2$ | $1.6 \mathrm{E}-2$ | 1 | $1.6 \mathrm{E}-9$ | $1.6 \mathrm{E}-15$ |
| 256 | 2048 | 80 | 80 | $4.4 \mathrm{E}-2$ | $4.3 \mathrm{E}-2$ | $5.8 \mathrm{E}-2$ | 2 | $1.3 \mathrm{E}-8$ | $1.1 \mathrm{E}-15$ |
| 512 | 4096 | 89 | 90 | $9.6 \mathrm{E}-2$ | $8.7 \mathrm{E}-2$ | $1.3 \mathrm{E}-1$ | 2 | $6.1 \mathrm{E}-8$ | $1.9 \mathrm{E}-15$ |
| 1024 | 8192 | 99 | 100 | $2.0 \mathrm{E}-1$ | $1.8 \mathrm{E}-1$ | $2.5 \mathrm{E}-1$ | 2 | $5.5 \mathrm{E}-8$ | $3.0 \mathrm{E}-15$ |
| 2048 | 16384 | 108 | 110 | $4.0 \mathrm{E}-1$ | $3.8 \mathrm{E}-1$ | $5.1 \mathrm{E}-1$ | 2 | $3.6 \mathrm{E}-8$ | $1.8 \mathrm{E}-15$ |
| 4096 | 32768 | 119 | 119 | $8.1 \mathrm{E}-1$ | $8.2 \mathrm{E}-1$ | $1.2 \mathrm{E}+0$ | 2 | $3.5 \mathrm{E}-8$ | $2.1 \mathrm{E}-15$ |
| 8192 | 65536 | 128 | 131 | $1.6 \mathrm{E}+0$ | $2.0 \mathrm{E}+0$ | $2.6 \mathrm{E}+0$ | 2 | $4.8 \mathrm{E}-9$ | $7.1 \mathrm{E}-09$ |
| 16384 | 131072 | 134 | 138 | $3.3 \mathrm{E}+0$ | $4.0 \mathrm{E}+0$ | $5.5 \mathrm{E}+0$ | 2 | $6.6 \mathrm{E}-9$ | $7.5 \mathrm{E}-09$ |

region of width $\delta$ ), our algorithm has optimal $\mathcal{O}(M+N)$ complexity, which is readily observed. Furthermore, as we have solved an approximate, compressed system, we cannot in general fit the data exactly (with respect to the true operator). Indeed, we see relative residuals of order $\mathcal{O}(\epsilon)$ as predicted by the compression tolerance. Thus, our algorithm is especially suitable in the event that observations need to be matched only to a specified precision. Our semidirect method vastly outperformed both LAPACK/ATLAS and FMM/GMRES, which required from 42 up to 880 iterations using $A^{\top}$ as a right preconditioner. Deferred correction was successful in all cases within two steps.
7.4. Thin plate splines with updating. In our final example, we demonstrate the efficiency of our updating and downdating methods in the typical setting of fitting additional observations to an already specified overdetermined system. For this, we employed the thin plate spline approximation problem of section 7.2 with $M=16384$ and $N=4096$, followed by the addition of 50 new random target points. From section 6 , the perturbed system can be written as (6.1) with (6.2), i.e.,

$$
\mathbf{E x} \simeq \mathbf{f}, \quad \text { where } \mathbf{E}=\left[\begin{array}{l}
A \\
C_{+}^{\mathrm{r}}
\end{array}\right]
$$

We used GMRES with the left preconditioner $\mathbf{B}=\left[A^{+},\left(C_{+}^{\mathrm{r}}\right)^{*}\right]$, which, since $A$ has full column rank, gives $\mathbf{B E}=I+\left(C_{+}^{\mathrm{r}}\right)^{*} C_{+}^{\mathrm{r}}$; hence the preconditioned system is

$$
\begin{equation*}
\left[I+\left(C_{+}^{\mathrm{r}}\right)^{*} C_{+}^{\mathrm{r}}\right] \mathbf{x} \simeq \mathbf{B f} \tag{7.2}
\end{equation*}
$$

Note that only one application of $A^{+}$is necessary, independent of the number of iterations required. Solving this in MATLAB took 17 iterations and a total of 1.9 s , with 1.6 s going towards setting up (7.2). The relative residual on the new data was $7.7 \times 10^{-3}$. This should be compared with the roughly 6 s required to solve the problem without updating, treating it instead as a new system via our compressed algorithm (Table 5). Although this difference is perhaps not very dramatic, it is worth emphasizing that the complexity here scales as $\mathcal{O}(M+N \log N)$ with updating versus $\mathcal{O}\left(M+N^{3 / 2}\right)$ without, as the former needs only to apply $A^{+}$while the latter needs also to compress and factor $\mathbf{E}$. Therefore, the asymptotics for updating are much improved.
8. Generalizations and conclusions. In this paper, we have presented a fast semidirect algorithm for overdetermined and underdetermined least squares problems involving HBS matrices, and exhibited its efficiency and practical performance in a variety of situations including RBF interpolation and dynamic updating. In one dimension (including boundary problems in two dimensions and problems with separated data in all dimensions), the solver achieves optimal $\mathcal{O}(M+N)$ complexity and is extremely fast, but it falters somewhat in higher dimensions, due primarily to the growing ranks of the compressed matrices as expressed by (5.1). Developments for addressing this growth are now underway for square linear systems [18, 40], and we expect these ideas to carry over to the present setting. Significantly, the term involving the larger matrix dimension is linear in all complexities (i.e., only $\mathcal{O}(M)$ instead of $\mathcal{O}\left(M N^{2}\right)$ as for classical direct methods), which makes our algorithm ideally suited to large, rectangular systems where both $M$ and $N$ increase with refinement.

Remark. If only one dimension is large so that the matrix is strongly rectangular, then standard methods are usually sufficient; see also [45, 48, 51].

Although we have not explicitly considered least squares problems with HBS equality constraints (we have only done so implicitly through our treatment of underdetermined systems), it is evident that our methods generalize. However, our complexity estimates can depend on the structure of the system matrix. In particular, if it is sparse, e.g., a diagonal weighting matrix, then our estimates are preserved. We can also, in principle, handle HBS least squares problems with HBS constraints simply by expanding out both matrices in sparse form.

This flexibility is one of our method's main advantages, though it can also create some difficulties. In particular, the fundamental problem is no longer the unconstrained least squares system (1.1) but the more complicated equality-constrained system (3.1). Accordingly, more sophisticated iterative techniques $[3,5,53]$ are used, but these can fail if the problem is too ill-conditioned. This is perhaps the greatest drawback of the proposed scheme. Still, our numerical results suggest that the algorithm remains effective for moderately ill-conditioned problems that are already quite challenging for standard iterative solvers. For severely ill-conditioned problems, other methods may be preferred.

Finally, it is worth noting that fast direct solvers can be leveraged for other least squares techniques as well. This is straightforward for the normal equations, which are subject to well-known conditioning issues, and for the somewhat better behaved augmented system version $[2,8]$ :

$$
\left[\begin{array}{cc}
I & A \\
A^{*} &
\end{array}\right]\left[\begin{array}{l}
r \\
x
\end{array}\right]=\left[\begin{array}{l}
b \\
0
\end{array}\right] .
$$

This approach has the advantage of being immediately amenable to fast inversion techniques but at the cost of "squaring" and enlarging the system. Thus, all com-
plexity estimates involve $M+N$ instead of $M$ and $N$ separately. In particular, the current generation of fast direct solvers would require, e.g., $\mathcal{O}\left((M+N)^{3(1-1 / d)}\right)$ instead of $\mathcal{O}\left(M+N^{3(1-1 / d)}\right)$ work. With the development of a next generation of linear or nearly linear time solvers $[18,33,40]$, this distinction may become less critical. Memory usage and high-performance computing hardware issues will also play important roles in determining which methods are most competitive. We expect these issues to become settled in the near future.

Acknowledgment. We would like to thank the anonymous referees for their careful reading and insightful remarks, which have improved the paper tremendously.

## REFERENCES

[1] E. Anderson, Z. Bai, C. Bischof, L. S. Blackford, J. Demmel, J. Dongarra, J. Du Croz, A. Greenbaum, S. Hammarling, A. McKenney, and D. Sorensen, LAPACK Users' Guide, 3rd ed., SIAM, Philadelphia, 1999.
[2] M. Arioli, I. S. Duff, and P. P. M. de Rijk, On the augmented system approach to sparse least-squares problems, Numer. Math., 55 (1989), pp. 667-684.
[3] J. L. Barlow, Error analysis and implementation aspects of deferred correction for equality constrained least squares problems, SIAM J. Numer. Anal., 25 (1988), pp. 1340-1358.
[4] J. L. Barlow and S. L. Handy, The direct solution of weighted and equality constrained least-squares problems, SIAM J. Sci. Stat. Comput., 9 (1988), pp. 704-716.
[5] J. L. Barlow and U. B. Vemulapati, A note on deferred correction for equality constrained least squares problems, SIAM J. Numer. Anal., 29 (1992), pp. 249-256.
[6] C. I. Bayly, P. Cieplak, W. D. Cornell, and P. A. Kollman, A well-behaved electrostatic potential based method using charge restraints for deriving atomic charges: The RESP model, J. Phys. Chem., 97 (1993), pp. 10269-10280.
[7] P. Benner and T. Mach, On the $Q R$ decomposition of $\mathcal{H}$-matrices, Computing, 88 (2010), pp. 111-129.
[8] A. Bנörck, Numerical Methods for Least Squares Problems, SIAM, Philadelphia, 1996.
[9] S. BÖRM, Data-sparse approximation of non-local operators by $\mathscr{H}^{2}$-matrices, Linear Algebra Appl., 422 (2007), pp. 380-403.
[10] M. D. Buhmann, Radial Basis Functions: Theory and Implementation, Cambridge University Press, Cambridge, 2003.
[11] F. Burns, D. Carlson, E. Haynsworth, and T. Markham, Generalized inverse formulas using the Schur complement, SIAM J. Appl. Math., 26 (1974), pp. 254-259.
[12] S. L. Campbell and C. D. Meyer Jr., Generalized Inverses of Linear Transformations, Pitman, London, 1979.
[13] J. C. Carr, R. K. Beatson, J. B. Cherrie, T. J. Mitchell, W. R. Fright, B. C. McCallum, and T. R. Evans, Reconstruction and representation of $3 D$ objects with radial basis functions, in Proceedings of the 28th Annual Conference on Computer Graphics and Interactive Techniques, Los Angeles, CA, ACM, New York, 2001, pp. 67-76.
[14] S. Chandrasekaran, P. Dewilde, M. Gu, T. Pals, X. Sun, A. J. Van Der Veen, and D. White, Some fast algorithms for sequentially semiseparable representations, SIAM J. Matrix Anal. Appl., 27 (2005), pp. 341-364.
[15] S. Chandrasekaran, P. Dewilde, M. Gu, W. Lyons, and T. Pals, A fast solver for HSS representations via sparse matrices, SIAM J. Matrix Anal. Appl., 29 (2006), pp. 67-81.
[16] S. Chandrasekaran, M. Gu, and T. Pals, A fast ULV decomposition solver for hierarchically semiseparable representations, SIAM J. Matrix Anal. Appl., 28 (2006), pp. 603-622.
[17] H. Cheng, Z. Gimbutas, P.-G. Martinsson, and V. Rokhlin, On the compression of low rank matrices, SIAM J. Sci. Comput., 26 (2005), pp. 1389-1404.
[18] E. Corona, P.-G. Martinsson, and D. Zorin, An $O(N)$ direct solver for integral equations on the plane, Appl. Comput. Hamon. Anal., to appear.
[19] T. A. DAVIS, Algorithm 832: UMFPACK V4.3-an unsymmetric-pattern multifrontal method, ACM Trans. Math. Software, 30 (2004), pp. 196-199.
[20] T. A. Davis, Algorithm 915, SuiteSparseQR: Multifrontal multithreaded rank-revealing sparse $Q R$ factorization, ACM Trans. Math. Software, 38 (2011), 8.
[21] P. Dewilde and S. Chandrasekaran, A hierarchical semi-separable Moore-Penrose equation solver, in Wavelets, Multiscale Systems and Hypergeometric Analysis, Oper. Theory Adv. Appl. 167, D. Alpay, A. Luger, and H. Woracek, eds., Birkhäuser, Basel, 2006, pp. 69-85.
[22] J. Duchon, Splines minimizing rotation-invariant semi-norms in Sobolev spaces, in Constructive Theory of Functions of Several Variables, Lecture Notes in Math. 571, W. Schempp and K. Zeller, eds., Springer-Verlag, Berlin, 1977, pp. 85-100.
[23] M. M. Francl, C. Carey, L. E. Chirlian, and D. M. Gange, Charges fit to electrostatic potentials. II. Can atomic charges be unambiguously fit to electrostatic potentials?, J. Comput. Chem., 17 (1996), pp. 367-383.
[24] A. Gillman, P. M. Young, and P.-G. Martinsson, A direct solver with $O(N)$ complexity for integral equations on one-dimensional domains, Front. Math. China., 7 (2012), pp. 217247.
[25] Z. Gimbutas and L. Greengard, FMMLIB: Fast Multipole Methods for Electrostatics, Elastostatics, and Low Frequency Acoustic Modeling, manuscript.
[26] L. Greengard, D. Gueyffier, P.-G. Martinsson, and V. Rokhlin, Fast direct solvers for integral equations in complex three-dimensional domains, Acta Numer., 18 (2009), pp. 243275.
[27] L. Greengard and V. Rokhlin, A fast algorithm for particle simulations, J. Comput. Phys., 73 (1987), pp. 325-348.
[28] L. Greengard and V. Rokhlin, A new version of the fast multipole method for the Laplace equation in three dimensions, Acta Numer., 6 (1997), pp. 229-269.
[29] T. N. E. Greville, Some applications of the pseudoinverse of a matrix, SIAM Rev., 2 (1960), pp. 15-22.
[30] M. Gu, New fast algorithms for structured linear least squares problems, SIAM J. Matrix Anal. Appl., 20 (1998), pp. 244-269.
[31] R. B. Guenther and J. W. Lee, Partial Differential Equations of Mathematical Physics and Integral Equations, Prentice-Hall, Englewood Cliffs, NJ, 1988.
[32] W. Hackbusch, A sparse matrix arithmetic based on $\mathcal{H}$-matrices. Part I: Introduction to $\mathcal{H}$ matrices, Computing, 62 (1999), pp. 89-108.
[33] W. Hackbusch and S. Börm, Data-sparse approximation by adaptive $\mathscr{H}^{2}$-matrices, Computing, 69 (2002), pp. 1-35.
[34] W. Hackbusch and B. N. Khoromskij, A sparse $\mathscr{H}$-matrix arithmetic. Part II: Application to multi-dimensional problems, Computing, 64 (2000), pp. 21-47.
[35] W. Hackbusch, B. N. Khoromskij, and S. Sauter, On $\mathcal{H}^{2}$-matrices, in Lectures on Applied Mathematics, H.-J. Bungartz, R. W. Hoppe, and C. Zenger, eds., Springer-Verlag, Berlin, 2000, pp. 9-29.
[36] R. L. Hardy, Multiquadric equations of topography and other irregular surfaces, J. Geophys. Res., 76 (1971), pp. 1905-1915.
[37] K. Hayami, J.-F. Yin, and T. Ito, GMRES methods for least squares problems, SIAM J. Matrix Anal. Appl., 31 (2010), pp. 2400-2430.
[38] N. J. Higham, Accuracy and Stability of Numerical Algorithms, 2nd. ed., SIAM, Philadelphia, 2002.
[39] K. L. Ho and L. Greengard, A fast direct solver for structured linear systems by recursive skeletonization, SIAM J. Sci. Comput., 34 (2012), pp. A2507-A2532.
[40] K. L. Ho and L. Ying, Hierarchical Interpolative Factorization for Elliptic Operators: Integral Equations, preprint, arXiv:1307.2666, 2013.
[41] T. Kailath and A. H. Sayed, Fast Reliable Algorithms for Matrices with Structure, SIAM, Philadelphia, 1999.
[42] C. L. Lawson and R. J. Hanson, Solving Least Squares Problems, Prentice-Hall, Englewood Cliffs, NJ, 1974.
[43] E. Liberty, F. Woolfe, P.-G. Martinsson, V. Rokhlin, and M. Tygert, Randomized algorithms for the low-rank approximation of matrices, Proc. Natl. Acad. Sci. USA, 104 (2007), pp. 20167-20172.
[44] P.-G. Martinsson and V. Rokhlin, A fast direct solver for boundary integral equations in two dimensions, J. Comput. Phys., 205 (2005), pp. 1-23.
[45] X. Meng, M. A. Saunders, and M. W. Mahoney, LSRN: A parallel iterative solver for strongly over- or under-determined systems, SIAM J. Sci. Comput., 36 (2014), pp. C95C118.
[46] C. C. Paige and M. A. Saunders, $L S Q R$ : An algorithm for sparse linear equations and sparse least squares, ACM Trans. Math. Software, 8 (1982), pp. 43-71.
[47] M. J. D. Powell, Radial basis functions for multivariable interpolation: A review, in Algorithms for Approximation, J. C. Mason and M. G. Cox, eds., Clarendon Press, Oxford, 1987, pp. 143-167.
[48] V. Rokhlin and M. Tygert, A fast randomized algorithm for overdetermined linear leastsquares regression, Proc. Natl. Acad. Sci. USA, 105 (2008), pp. 13212-13217.
[49] Y. SaAd and M. H. Schultz, GMRES: A generalized minimal residual algorithm for solving nonsymmetric linear systems, SIAM J. Sci. Stat. Comput., 7 (1986), pp. 856-869.
[50] H. Samet, The quadtree and related hierarchical data structures, ACM Comp. Surv., 16 (1984), pp. 187-260.
[51] M. Tygert, A Fast Algorithm for Computing Minimal-Norm Solutions to Underdetermined Systems of Linear Equations, preprint, arXiv:0905.4745, 2009.
[52] M. Van Barel, G. Heinig, and P. Kravanja, A superfast method for solving Toeplitz linear least squares problems, Linear Algebra Appl., 366 (2003), pp. 441-457.
[53] C. VAN LoAN, On the method of weighting for equality-constrained least-squares problems, SIAM J. Numer. Anal., 22 (1985), pp. 851-864.
[54] R. C. Whaley, A. Petitet, and J. J. Dongarra, Automated empirical optimization of software and the ATLAS project, Parallel Comput., 27 (2001), pp. 3-35.
[55] F. Woolfe, E. Liberty, V. Rokhlin, and M. Tygert, A fast randomized algorithm for the approximation of matrices, Appl. Comput. Harmon. Anal., 25 (2008), pp. 335-366.
[56] J. Xia, S. Chandrasekaran, M. Gu, and X. S. Li, Fast algorithms for hierarchically semiseparable matrices, Numer. Linear Algebra Appl., 17 (2010), pp. 953-976.
[57] J. Xia, Y. Xi, and M. Gu, A superfast structured solver for Toeplitz linear systems via randomized sampling, SIAM J. Matrix Anal. Appl., 33 (2012), pp. 837-858.
[58] L. Ying, G. Biros, and D. Zorin, A kernel-independent adaptive fast multipole algorithm in two and three dimensions, J. Comput. Phys., 196 (2004), pp. 591-626.


[^0]:    *Received by the editors December 17, 2012; accepted for publication (in revised form) by M. Van Barel April 7, 2014; published electronically June 5, 2014. This work was supported in part by the National Science Foundation under awards DGE-0333389 and DMS-1203554, by the U.S. Department of Energy under contract DEFG0288ER25053, and by the Air Force Office of Scientific Research under NSSEFF Program Award FA9550-10-1-0180.
    http://www.siam.org/journals/simax/35-2/90267.html
    ${ }^{\dagger}$ Courant Institute of Mathematical Sciences and Program in Computational Biology, New York University, New York, NY 10012. Current address: Department of Mathematics, Stanford University, Stanford, CA 94305-2125 (klho@stanford.edu).
    ${ }^{\ddagger}$ Courant Institute of Mathematical Sciences, New York University, New York, NY 10012 (greengard@courant.nyu.edu).

