CHAPTER 1

A simple direct solver for integral equations

1.1. Broader context of solvers for discretized integral equations

In the past several chapters, we have investigated techniques for formulating a physical problem such as electro-statics or acoustic scattering in the form of an integral equation such as

$$\alpha\, q(\boldsymbol{x}) + \int_\Gamma K(\boldsymbol{x},\boldsymbol{y})\, q(\boldsymbol{y})\, ds(\boldsymbol{y}) = f(\boldsymbol{x}), \qquad \boldsymbol{x} \in \Gamma. \tag{1.1}$$

This is in contrast to the perhaps more common methodology of formulating the problem as a partial differential equation. Having obtained an integral equation, we saw in Chapter ? how to discretize the problem to convert a continuum equation such as (1.1) to a linear system such as

$$\mathbf{A}\mathbf{q} = \mathbf{f}. \tag{1.2}$$

Since the coefficient matrix $\mathbf{A}$ will be dense, simplistic solvers such as Gaussian elimination will have cubic complexity, which limits their use to very small problem sizes. Standard practice in applications is then to switch to an iterative solver, whose cost will typically scale as

$$T_{\text{solve}} \approx N_{\text{iter}} \times T_{\text{matvec}},$$

where N_{iter} is the number of iterations required for convergence and T_{matvec} is the time required for applying $\mathbf{A}$ to a vector. When the underlying BIE is a Fredholm equation of the second kind, N_{iter} is often small, since the matrix $\mathbf{A}$ in this case tends to be well conditioned. The cost T_{matvec} of applying $\mathbf{A}$ can in most common environments be accelerated from the $O(N^2)$ cost of forming and applying $\mathbf{A}$ directly to the linear or close to linear complexity of fast summation schemes such as the FMM or the $\mathcal{H}$-matrix framework. (In practice, BIE formulations often give accurate results for N small enough that acceleration schemes are not used.)

The use of iterative methods to solve (1.2) is well established at this point, and we refer to the textbooks [**4**] and [**1**], and the survey [**5**] for additional details. We remark that in order to attain fast convergence in practice, it is often necessary to introduce preconditioners, which may complicate the implementation substantially.

As an alternative to iterative solvers accelerated by fast summation methods, there also exist accelerated *direct* solvers whose complexity is in many environments also linear. Such direct solvers have an advantage in that their execution time is not held hostage to the speed of convergence, and that they work without the need for specialized pre-conditioners. As we discussed in the introduction, a direct solver can typically be viewed as consisting of a "build stage" where an approximate inverse $\mathbf{B}$ of the coefficient matrix $\mathbf{A}$ is built, and then for each given data vector $\mathbf{f}$, a "solve stage" where the approximate solution $\mathbf{u}_{\text{approx}} = \mathbf{B}\mathbf{f}$ is computed. As the solve stage is typically very fast, direct solvers are particularly compelling in situations where a sequence of linear systems involving the same coefficient matrix need to be solved. On the other hand, the build stage is for problems in 3D typically quite slow, which tends to make iterative solvers run faster in situations where only a single solve is performed for each coefficient matrix, and where convergence is fast.

Over the next few chapters, we will engage with some of the core ideas underlying accelerated direct solvers for boundary integral equations. In this first chapter (Chapter 1), we will develop a simple single-level

scheme that requires minimal data-structures while introducing the essential concepts. Chapter 2 describes how the complexity of the solver can be improved dramatically by moving to a full hierarchical domain decomposition. (In Chapters 1 and 2, we closely follow the presentation in [**2**].) Chapter 3 explores some issues related to the particular rank structured data format we introduce in Chapters 1 and 2, and describe how it relates to the HODLR format of the introduction and to other popular formats. Chapter 4 describes how the interpolative decomposition provides a powerful tool for accelerating and simplifying direct solvers for BIEs. In Chapter 5, we describe how to build the data-sparse representation of the coefficient matrix in the first place.

A reader who wishes to focus on the high level concepts and does not intend to actually implement the methods may want to start with Chapters 1 and 2, and then Sections 4.3 and 5.1.

Throughout this chapter, we consider the BIE (1.1) for the case where Γ is the boundary of a domain Ω in two or three dimensions. We typically have in mind a linear system (1.2) obtained from a Nyström discretization based on a set of interpolation points $\{\boldsymbol{x}_i\}_{i=1}^N$ and weights $\{w_i\}_{i=1}^N$, as described in Chapter ?. However, many of the techniques described apply equally well to other discretization schemes (collocation, BEM, etc.).

1.2. Block separable matrices

To introduce a simple direct solver, let $\mathbf{A}$ denote the $N \times N$ dense coefficient matrix in (1.2). Let us tessellate it into $p \times p$ blocks, so that

$$\mathbf{A} = \begin{bmatrix} \mathbf{D}_1 & \mathbf{A}_{1,2} & \mathbf{A}_{1,3} & \cdots & \mathbf{A}_{1,p} \\ \mathbf{A}_{2,1} & \mathbf{D}_2 & \mathbf{A}_{2,3} & \cdots & \mathbf{A}_{2,p} \\ \vdots & \vdots & \vdots & & \vdots \\ \mathbf{A}_{p,1} & \mathbf{A}_{p,2} & \mathbf{A}_{p,3} & \cdots & \mathbf{D}_p \end{bmatrix}. \tag{1.3}$$

For simplicity, let us suppose that each block is of the same size $n \times n$. (There is no difficulty in letting the block sizes vary, other than the need to introduce a more unwieldy notational machinery.) In this section, we will demonstrate that when a matrix $\mathbf{A}$ can be tesselated as in (1.3) in such a way that all off-diagonal blocks satisfy a certain low-rank condition, then there is a simple formula that allows us to compute the inverse of $\mathbf{A}$ efficiently. To be precise, the first condition that we need is that there should be some "small" number k that serves as an upper bound for the ranks of all the off-diagonal blocks. We additionally assume that there exist some basis matrices $\{\mathbf{U}_\kappa\}_{\kappa=1}^p$ and $\{\mathbf{V}_\kappa\}_{\kappa=1}^p$ such that each off-diagonal block $\mathbf{A}_{\sigma,\tau}$ of $\mathbf{A}$ admits the factorization

$$\underset{n\times n}{\mathbf{A}_{\sigma,\tau}} = \underset{n\times k}{\mathbf{U}_\sigma} \; \underset{k\times k}{\tilde{\mathbf{A}}_{\sigma,\tau}} \; \underset{k\times n}{\mathbf{V}_\tau^*}, \qquad \sigma,\tau \in \{1,\,2,\,\ldots,\,p\}, \quad \sigma \neq \tau. \tag{1.4}$$

Note that this is stronger than merely assuming that the rank of each matrix is bounded by k. Specifically, the columns of $\mathbf{U}_\sigma$ must form a basis for the columns of *every* off-diagonal block in row σ, and analogously, the columns of $\mathbf{V}_\tau$ must form a basis for the rows in every the off-diagonal blocks in column τ.

When (1.4) holds, the matrix $\mathbf{A}$ admits a block factorization

$$\underset{pn\times pn}{\mathbf{A}} = \underset{pn\times pk}{\mathbf{U}} \; \underset{pk\times pk}{\tilde{\mathbf{A}}} \; \underset{pk\times pn}{\mathbf{V}^*} + \underset{pn\times pn}{\mathbf{D}}, \tag{1.5}$$

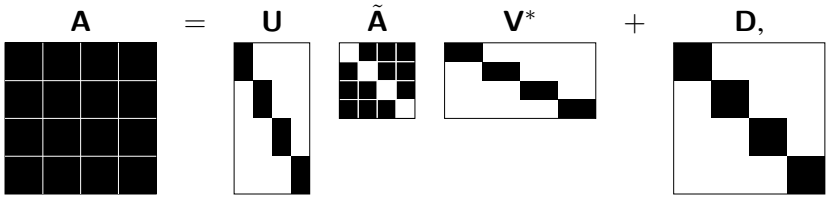
where

$$\mathbf{U} = \mathrm{diag}(\mathbf{U}_1,\,\mathbf{U}_2,\,\ldots,\,\mathbf{U}_p), \quad \mathbf{V} = \mathrm{diag}(\mathbf{V}_1,\,\mathbf{V}_2,\,\ldots,\,\mathbf{V}_p), \quad \mathbf{D} = \mathrm{diag}(\mathbf{D}_1,\,\mathbf{D}_2,\,\ldots,\,\mathbf{D}_p),$$

and

$$\tilde{\mathbf{A}} = \begin{bmatrix} \mathbf{0} & \tilde{\mathbf{A}}_{12} & \tilde{\mathbf{A}}_{13} & \cdots \\ \tilde{\mathbf{A}}_{21} & \mathbf{0} & \tilde{\mathbf{A}}_{23} & \cdots \\ \tilde{\mathbf{A}}_{31} & \tilde{\mathbf{A}}_{32} & \mathbf{0} & \cdots \\ \vdots & \vdots & \vdots & \end{bmatrix}.$$

The block structure of formula (1.5) for $p = 4$ is illustrated below:



The point here is that (1.5) expresses $\mathbf{A}$ as a sum of two terms, where the first term (the product $\mathbf{U}\tilde{\mathbf{A}}\mathbf{V}^*$) is of low rank, and the second term (the block diagonal matrix $\mathbf{D}$) is easy to invert. This is exactly the framing for the well-known Woodbury formula for inverting a low rank perturbation of a matrix. For the specific split in (1.5), the relevant formula takes the following form:

LEMMA 1.1 (Variation of Woodbury). *Suppose that $\mathbf{A}$ is an invertible $N \times N$ matrix, that K is a positive integer smaller than N, and that $\mathbf{A}$ admits the factorization*

$$\underset{N\times N}{\mathbf{A}} = \underset{N\times K}{\mathbf{U}}\ \underset{K\times K}{\tilde{\mathbf{A}}}\ \underset{K\times N}{\mathbf{V}^*} + \underset{N\times N}{\mathbf{D}}. \tag{1.6}$$

Then

$$\underset{N\times N}{\mathbf{A}^{-1}} = \underset{N\times K}{\mathbf{E}}\ \underset{K\times K}{\left(\tilde{\mathbf{A}} + \hat{\mathbf{D}}\right)^{-1}}\ \underset{K\times N}{\mathbf{F}^*} + \underset{N\times N}{\mathbf{G}}, \tag{1.7}$$

where

$$\hat{\mathbf{D}} = \left(\mathbf{V}^*\,\mathbf{D}^{-1}\,\mathbf{U}\right)^{-1}, \tag{1.8}$$

$$\mathbf{E} = \mathbf{D}^{-1}\,\mathbf{U}\,\hat{\mathbf{D}}, \tag{1.9}$$

$$\mathbf{F} = (\hat{\mathbf{D}}\,\mathbf{V}^*\,\mathbf{D}^{-1})^*, \tag{1.10}$$

$$\mathbf{G} = \mathbf{D}^{-1} - \mathbf{D}^{-1}\,\mathbf{U}\,\hat{\mathbf{D}}\,\mathbf{V}^*\,\mathbf{D}^{-1}, \tag{1.11}$$

provided all inverses that appear in the formulas exist.

PROOF. Fix an $\mathbf{f} \in \mathbb{R}^N$ and set $\mathbf{q} = \mathbf{A}^{-1}\mathbf{f}$. Then

$$\mathbf{A}\mathbf{q} = \mathbf{f}. \tag{1.12}$$

Furthermore, set

$$\tilde{\mathbf{q}} = \mathbf{V}^*\mathbf{q}. \tag{1.13}$$

Combining (1.6), (1.12), and (1.13) we get a linear system

$$\begin{bmatrix} \mathbf{D} & \mathbf{U}\tilde{\mathbf{A}} \\ -\mathbf{V}^* & \mathbf{I} \end{bmatrix} \begin{bmatrix} \mathbf{q} \\ \tilde{\mathbf{q}} \end{bmatrix} = \begin{bmatrix} \mathbf{f} \\ \mathbf{0} \end{bmatrix}.$$

Eliminating $\tilde{\mathbf{q}}$ via a Schur complement, we find an equation for $\tilde{\mathbf{q}}$ as follows:

$$\left(\mathbf{I} + \mathbf{V}^*\mathbf{D}^{-1}\mathbf{U}\tilde{\mathbf{A}}\right)\tilde{\mathbf{q}} = \mathbf{V}^*\mathbf{D}^{-1}\mathbf{f}. \tag{1.14}$$

Multiply both sides of (1.14) by $\hat{\mathbf{D}} = (\mathbf{V}^*\mathbf{D}^{-1}\mathbf{U})^{-1}$ to get

$$\left(\hat{\mathbf{D}} + \tilde{\mathbf{A}}\right)\tilde{\mathbf{q}} = \hat{\mathbf{D}}\mathbf{V}^*\mathbf{D}^{-1}\mathbf{f}. \tag{1.15}$$

Solving (1.15) for $\tilde{\mathbf{q}}$, we get

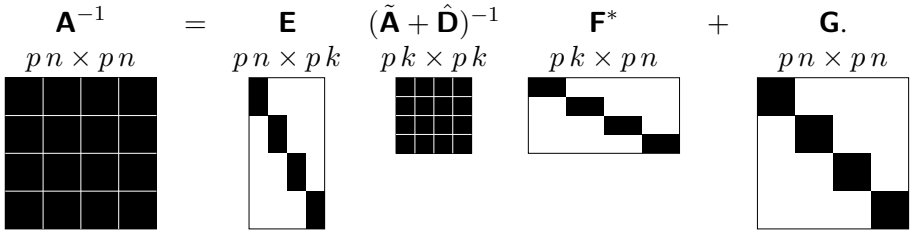
$$\tilde{\mathbf{q}} = \left(\hat{\mathbf{D}} + \tilde{\mathbf{A}}\right)^{-1} \hat{\mathbf{D}}\mathbf{V}^*\mathbf{D}^{-1}\mathbf{f}. \tag{1.16}$$

Finally, we can express $\mathbf{q}$ as follows

$$\begin{aligned}
\mathbf{q} &= \mathbf{D}^{-1}\mathbf{f} - \mathbf{D}^{-1}\mathbf{U}\tilde{\mathbf{A}}\tilde{\mathbf{q}} && \textit{(use relation (1.15) to eliminate } \tilde{\mathbf{A}}\tilde{\mathbf{q}}\textit{)} \\
&= \mathbf{D}^{-1}\mathbf{f} - \mathbf{D}^{-1}\mathbf{U}\left(\hat{\mathbf{D}}\mathbf{V}^*\mathbf{D}^{-1}\mathbf{f} - \hat{\mathbf{D}}\tilde{\mathbf{q}}\right) && \textit{(use relation (1.16) to eliminate } \tilde{\mathbf{q}}\textit{)} \\
&= \underbrace{\left(\mathbf{D}^{-1} - \mathbf{D}^{-1}\mathbf{U}\hat{\mathbf{D}}\mathbf{V}^*\mathbf{D}^{-1}\right)}_{=\mathbf{G}}\mathbf{f} + \underbrace{\mathbf{D}^{-1}\mathbf{U}\hat{\mathbf{D}}}_{=\mathbf{E}}\left(\hat{\mathbf{D}} + \tilde{\mathbf{A}}\right)^{-1}\underbrace{\hat{\mathbf{D}}\mathbf{V}^*\mathbf{D}^{-1}}_{=\mathbf{F}^*}\mathbf{f}.
\end{aligned}$$

Since $\mathbf{q}$ was defined via $\mathbf{q} = \mathbf{A}^{-1}\mathbf{f}$, this completes the proof. □

When Lemma 1.1 is applied to (1.5), the computed inverse takes the form



Now observe that the matrices $\hat{\mathbf{D}}$, $\mathbf{E}$, $\mathbf{F}$, and $\mathbf{G}$ are cheap to form since all matrices used to construct them are block diagonal. This means that Lemma 1.1 reduces the task of inverting the large (size $pn \times pn$) matrix $\mathbf{A}$ to the task of inverting the small (size $pk \times pk$) matrix $\tilde{\mathbf{A}} + \hat{\mathbf{D}}$.

Let us observe that Lemma 1.1 does not say anything about the *stability* of the inversion formula. When $\mathbf{A}$ is positive symmetric definite and $\mathbf{U}$ is orthonormal, it is possible to show that the intermediate matrices are no worse conditioned than $\mathbf{A}$ itself, see Section 3.2, or [**2**, Sec. 3]. For general $\mathbf{A}$, no such guarantees can hold (indeed, one can construct arbitrarily ill-conditioned counter examples).

REMARK 1.1. *The version of the Woodbury formula we use here is slightly unusual. A more formulation is*

$$\left(\mathbf{D} + \mathbf{U}\tilde{\mathbf{A}}\mathbf{V}^*\right)^{-1} = \mathbf{D}^{-1} - \mathbf{D}^{-1}\mathbf{U}\left(\tilde{\mathbf{A}}^{-1} + \mathbf{V}^*\mathbf{D}^{-1}\mathbf{U}\right)^{-1}\mathbf{V}^*\mathbf{D}^{-1}. \tag{1.17}$$

The problem with the formula (1.17) is that it involves the inverse of $\tilde{\mathbf{A}}$, which is not block diagonal, and is in our applications often not invertible. To get around this difficulty, we use the matrix identity

$$\left(\mathbf{X}^{-1} + \mathbf{Y}^{-1}\right)^{-1} = \mathbf{Y} - \mathbf{Y}\left(\mathbf{X} + \mathbf{Y}\right)^{-1}\mathbf{Y}, \tag{1.18}$$

which is valid for any two equi-sized square matrices $\mathbf{X}$ and $\mathbf{Y}$ for which all inverses exist. Lemma 1.1 now results if we set $\mathbf{X} = \tilde{\mathbf{A}}$ and with $\mathbf{Y} = \hat{\mathbf{D}} = \left(\mathbf{V}^*\mathbf{D}^{-1}\mathbf{U}\right)^{-1}$ in (1.18), and insert the result in the classic Woodbury formula (1.17).

1.3. Asymptotic complexity of the Woodbury formula for a block separable matrix

Suppose that we are given a block separable matrix $\mathbf{A}$ consisting of $p \times p$ blocks, each of size $n \times n$, with block rank k. Let us further assume that we are given the factors in a BS factorization. Then in order to apply the formula (1.7), we need to first compute the block diagonal matrices $\hat{\mathbf{D}}$, $\mathbf{E}$, $\mathbf{F}$, and $\mathbf{G}$. Each matrix holds p diagonal blocks, and in order to form each diagonal block operations on matrices of size at most

$n \times n$ are carried out, so this cost scales as pn^3. Then the $pk \times pk$ matrix $\tilde{\mathbf{A}} + \hat{\mathbf{D}}$ needs to be inverted. Since this matrix is dense, the total cost T works out to be

$$T \sim pn^3 + (pk)^3.$$

In applications, we are typically given a fixed matrix $\mathbf{A}$, and are free to choose an optimal tessellation. Finding the optimal choice is a slightly complicated question since the rank k in general depends on the size of the blocks. However, the dependence is sometimes weak, as happens for instance for matrices arising from discretizing a BIE on a curve in the plane. Let us for simplicity suppose that k is fixed. Then what is the optimal choice of p? First observe that in this scenario, $n = N/p$, so we would have

$$T \sim p\,(N/p)^3 + (pk)^3 \sim p^{-2}N^3 + p^3k^3.$$

Differentiating with respect to p, and setting the derivative to zero, we find that the optimal choice is $p \sim (N/k)^{3/5}$, which then leads to

$$T \sim N^3p^{-2} + k^3p^3 \sim N^3(N/k)^{-6/5} + k^3(N/k)^{9/5} \sim N^{9/5}k^{6/5}.$$

We see that the complexity is slightly better than quadratic, which is a substantial improvement over the cubic complexity of standard Gaussian elimination of a dense system.

1.4. How to compute a block separable representation of a matrix

Let us close this chapter by briefly discussing how to obtain the factors in a factorization such as (1.4). To do this, we first need to write down a precise definition for the type of rank-structured matrix we have encountered in this section:

DEFINITION 1. Let $\mathbf{A}$ be an $N \times N$ matrix, let $I = 1 : N$ be its full index vector, and let

$$I = I_1 \cup I_2 \cup \cdots \cup I_p \tag{1.19}$$

be a disjoint partition of I. We then say that $\mathbf{A}$ has *block rank* k with respect to this partition if there exist matrices $\{\mathbf{U}_\kappa\}_{\kappa=1}^p$ and $\{\mathbf{V}_\kappa\}_{\kappa=1}^p$ such that each off-diagonal block of $\mathbf{A}$ admits a factorization

$$\begin{array}{cccccc} \mathbf{A}(I_\sigma, I_\tau) & = & \mathbf{U}_\sigma & \tilde{\mathbf{A}}_{\sigma,\tau} & \mathbf{V}_\tau^*, & \sigma, \tau \in \{1,\, 2,\, \ldots,\, p\}, \quad \sigma \neq \tau. \\ n_\sigma \times n_\tau & & n_\sigma \times k & k \times k & k \times n_\tau & \end{array} \tag{1.20}$$

where n_τ is the length of I_τ.

We informally say that a matrix is *block separable (BS)* if its block rank is "small". In practical applications, most matrices that we are interested in have off-diagonal blocks that are only of low-rank to within some preset tolerance ε. We will allow ourselves to refer to such a matrix as being block-separable, although one should perhaps to be strict refer to such a matrix as "block separable to precision ε".

Let us now consider the question of how to compute the basis matrices $\{\mathbf{U}_\tau\}$ and $\{\mathbf{V}_\tau\}$, given a matrix $\mathbf{A}$ and a partition (1.19) of the index vector. For any given τ, the columns of the basis matrix $\mathbf{U}_\tau$ must span the columns of every block $\mathbf{A}_{\tau,\sigma}$ for which $\sigma \neq \tau$. In other words, the columns of $\mathbf{U}_\tau$ must span all the columns in the matrix

$$\mathbf{A}(I_\tau, I_\tau^{\mathrm{c}}) = \big[\mathbf{A}_{\tau,1},\, \mathbf{A}_{\tau,2},\, \ldots,\, \mathbf{A}_{\tau,\tau-1},\, \mathbf{A}_{\tau,\tau+1},\, \ldots,\, \mathbf{A}_{\tau,p}\big],$$

where $I_\tau^{\mathrm{c}} = (1 : N) \backslash I_\tau$. We can solve this task using an SVD as follows:

$$[\mathbf{U}_\tau, \mathbf{X}_\tau, \mathbf{Y}_\tau] = \mathtt{svd}(\mathbf{A}(I_\tau, I_\tau^{\mathrm{c}}), k). \tag{1.21}$$

We could then in principle compute all the $\mathbf{V}_\tau$ matrices in a similar fashion, and then compute $\tilde{\mathbf{A}}_{\sigma,\tau} = \mathbf{U}_\sigma^*\mathbf{A}(I_\sigma, I_\tau)\mathbf{V}_\tau$ for every off-diagonal block. This would be a bit wasteful, however. It turns out to be possible to use the output of the SVD in (1.21) as input for a compression procedure that computes both the $\mathbf{V}_\tau$ matrices, and all matrices $\tilde{\mathbf{A}}_{\sigma,\tau}$ in one go. We leave the details as an exercise.

In Chapter 5 we describe compression techniques that apply specifically when a BS matrix arises from the discretization of a boundary integral operator. In this case, there turns out to be no need to form the entire matrix, and the compression procedure ends up having complexity that scales linearly with N (assuming that the block size n is kept fixed while the number of blocks $p \to \infty$).

CHAPTER 2

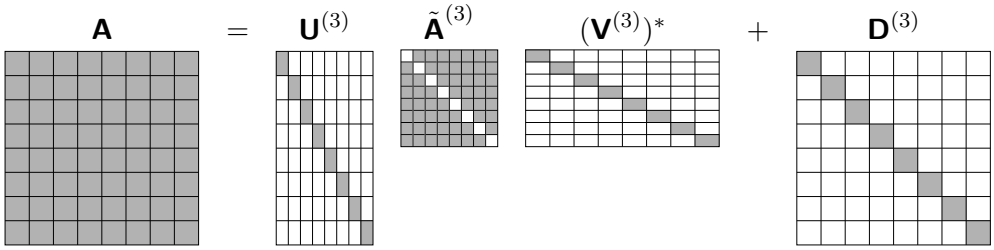
A multilevel scheme

The scheme described in Chapter 1 is quite a powerful numerical tool in its own right, but it cannot reach optimal computational complexity. In this chapter, we will show how to apply the method recursively to attain linear, or close to linear, complexity for many problems of high importance in applications. The key observation is that the matrix $\tilde{\mathbf{A}} + \hat{\mathbf{D}}$ that needs to be inverted in the Woodbury formula (1.7) in Lemma 1.1 is itself often block separable, and can be compressed again. We refer to a matrix that admits a hierarchial structure of block separable matrices as a *Hierarchically Block Separable (HBS)* matrix.

2.1. Heuristic description

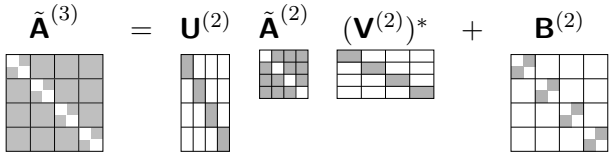
For a matrix to be hierarchically block separable (HBS), it must first of all be block separable, as defined in Section 1.2. To informally introduce the key concept, let us consider a matrix $\mathbf{A}$ that consists of 8×8 blocks, so that *cf.* (1.5):

(2.1)



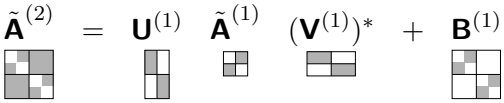
The superscripts on the right-hand side of (2.1) indicate that the factorization is at what we call "level 3" (the motivation for this labeling should become clear once we reach equation (2.4)). We next require the smaller matrix $\tilde{\mathbf{A}}^{(3)}$ to be BS, and to admit the analogous factorization:

(2.2)



In forming (2.2), we reblocked the matrix $\tilde{\mathbf{A}}^{(3)}$ by merging blocks in groups of four. The purpose is to "reintroduce" rank deficiencies in the off-diagonal blocks. In the final step in the hierarchy, we require that upon reblocking, $\tilde{\mathbf{A}}^{(2)}$ is BS and admits a factorization:

(2.3)



Combining (2.1), (2.2), and (2.3), we find that $\mathbf{A}$ can be expressed as

$$\mathbf{A} = \mathbf{U}^{(3)}\big(\mathbf{U}^{(2)}\big(\mathbf{U}^{(1)}\,\mathbf{B}^{(0)}\,(\mathbf{V}^{(1)})^* + \mathbf{B}^{(1)}\big)(\mathbf{V}^{(2)})^* + \mathbf{B}^{(2)}\big)(\mathbf{V}^{(3)})^* + \mathbf{D}^{(3)}. \tag{2.4}$$

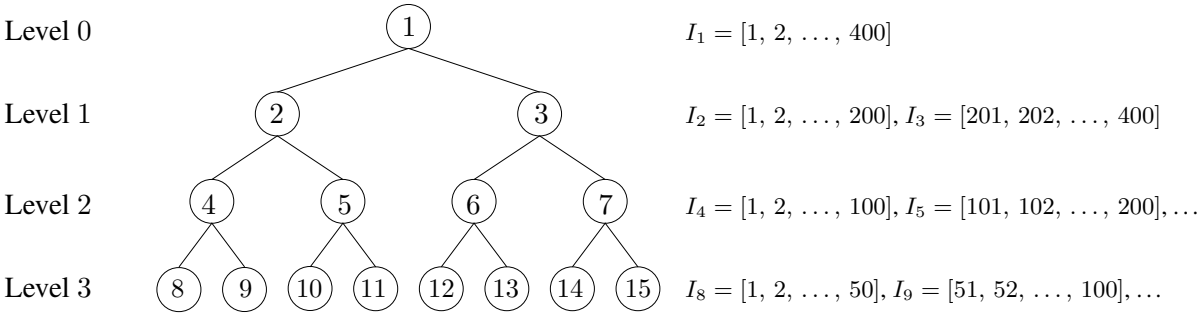
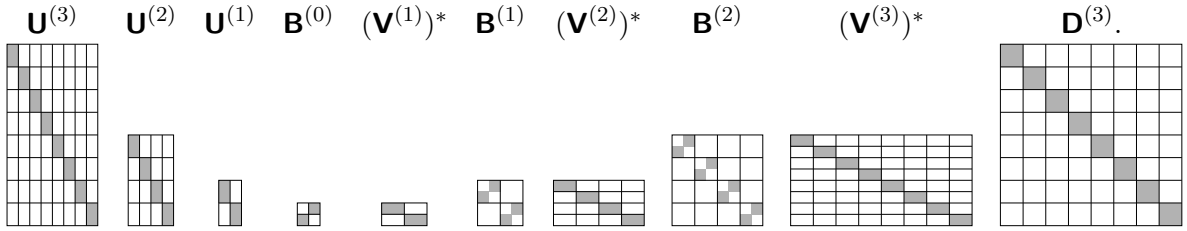


FIGURE 2.1. Numbering of nodes in a fully populated binary tree with $L = 3$ levels. The root is the original index vector $I = I_1 = [1, 2, \ldots, 400]$.

The block structure of the right hand side of (2.4) is:



In other words, the HBS property lets us completely represent a matrix in terms of certain block diagonal factors. The total cost of storing these factors is $O(Nk)$, and the format is an example of a so-called "data-sparse" representation of the matrix.

2.2. Tree structure

The HBS representation of an $N \times N$ matrix $\mathbf{A}$ is based on a partition of the index vector $I = [1, 2, \ldots, N]$ into a binary tree structure. For simplicity, we limit attention to binary tree structures in which every level is fully populated. We let I form the root of the tree, and give it the index 1, $I_1 = I$. We next split the root into two roughly equi-sized vectors I_2 and I_3 so that $I_1 = I_2 \cup I_3$. The full tree is then formed by continuing to subdivide any interval that holds more than some preset fixed number n of indices. We use the integers $\ell = 0, 1, \ldots, L$ to label the different levels, with 0 denoting the coarsest level. A *leaf* is a node corresponding to a vector that never got split. For a non-leaf node τ, its *children* are the two boxes α and β such that $I_\tau = I_\alpha \cup I_\beta$, and τ is then the *parent* of α and β. Two boxes with the same parent are called *siblings*. These definitions are illustrated in Figure 2.1

REMARK 2.1. *The HBS format works with a broad range of different tree structures. It is permissible to split a node into more than two children if desirable, to distribute the points in an index set unevenly among its children, to split only some nodes on a given level, etc. This flexibility is essential when* $\mathbf{A}$ *approximates a non-uniformly discretized integral operator; in this case, the partition tree it constructed based on a geometric subdivision of the domain of integration. The spatial geometry of a box then dictates whether and how it is to be split.*

2.3. Definition of the HBS property

We are now prepared to rigorously define what it means for an $N \times N$ matrix $\mathbf{A}$ to be *hierarchically block seperable* with respect to a given binary tree $\mathcal{T}$ that partitions the index vector $J = [1, 2, \ldots, N]$. We suppose that the tree has L fully populated levels, and that for every leaf node τ, the index vector I_τ holds precisely n points, so that $N = n\,2^L$. Then $\mathbf{A}$ is HBS with block rank k if the following two conditions hold:

	Name:	Size:	Function:
For each leaf node τ:	$\mathbf{D}_\tau$	$n \times n$	The diagonal block $\mathbf{A}(I_\tau, I_\tau)$.
	$\mathbf{U}_\tau$	$n \times k$	Basis for the columns in the blocks in row τ.
	$\mathbf{V}_\tau$	$n \times k$	Basis for the rows in the blocks in column τ.
For each parent node τ:	$\mathbf{B}_\tau$	$2k \times 2k$	Interactions between the children of τ.
	$\mathbf{U}_\tau$	$2k \times k$	Basis for the columns in the (reduced) blocks in row τ.
	$\mathbf{V}_\tau$	$2k \times k$	Basis for the rows in the (reduced) blocks in column τ.

FIGURE 2.2. An HBS matrix $\mathbf{A}$ associated with a tree $\mathcal{T}$ is fully specified if the factors listed above are provided.

(1) Assumption on ranks of off-diagonal blocks at the finest level: For each pair of distinct leaf nodes τ and τ', let us define the matrix $\mathbf{A}_{\tau,\tau'}$ via

$$\mathbf{A}_{\tau,\tau'} = \mathbf{A}(I_\tau, I_{\tau'}). \tag{2.5}$$

We then require that each such matrix has rank k, and can be factorized as

$$\begin{array}{ccccc} \mathbf{A}_{\tau,\tau'} & = & \mathbf{U}_\tau & \tilde{\mathbf{A}}_{\tau,\tau'} & \mathbf{V}^*_{\tau'}. \\ n \times n & & n \times k & k \times k & k \times n \end{array} \tag{2.6}$$

(2) Assumption on ranks of off-diagonal blocks on level $\ell = L-1, L-2, \ldots, 1$: The rank assumption at level ℓ is defined in terms of the blocks constructed on the next finer level $\ell+1$: For any distinct nodes τ and τ' on level ℓ with children α, β and α', β', respectively, define

$$\mathbf{A}_{\tau,\tau'} = \begin{bmatrix} \tilde{\mathbf{A}}_{\alpha,\alpha'} & \tilde{\mathbf{A}}_{\alpha,\beta'} \\ \tilde{\mathbf{A}}_{\beta,\alpha'} & \tilde{\mathbf{A}}_{\beta,\beta'} \end{bmatrix}. \tag{2.7}$$

We then require that each such matrix has rank k, and can be factorized as

$$\begin{array}{ccccc} \mathbf{A}_{\tau,\tau'} & = & \mathbf{U}_\tau & \tilde{\mathbf{A}}_{\tau,\tau'} & \mathbf{V}^*_{\tau'}. \\ 2k \times 2k & & 2k \times k & k \times k & k \times 2k \end{array} \tag{2.8}$$

The two points above complete the definition. An HBS matrix is now fully described if the basis matrices $\mathbf{U}_\tau$ and $\mathbf{V}_\tau$ are provided for each node τ, and in addition, we are for each leaf τ given the $n \times n$ matrix

$$\mathbf{D}_\tau = \mathbf{A}(I_\tau, I_\tau), \tag{2.9}$$

and for each parent node τ with children α and β we are given the $2k \times 2k$ matrix

$$\mathbf{B}_\tau = \begin{bmatrix} \mathbf{0} & \tilde{\mathbf{A}}_{\alpha,\beta} \\ \tilde{\mathbf{A}}_{\beta,\alpha} & \mathbf{0} \end{bmatrix}. \tag{2.10}$$

Observe in particular that the matrices $\tilde{\mathbf{A}}_{\alpha,\beta}$ are only required when $\{\alpha, \beta\}$ forms a sibling pair. Figure 2.2 summarizes the required matrices.

2.4. Telescoping factorizations

In the heuristic description of the HBS property in Section 2.1, we claimed that any HBS matrix can be expressed as a telescoping factorization with block diagonal factors. We will now formalize this claim.

Given the matrices defined in Section 2.3 (and summarized in Figure 2.2), we define the following block diagonal factors:

$$\mathbf{D}^{(\ell)} = \text{diag}(\mathbf{D}_\tau : \tau \text{ is a box on level } \ell), \qquad \ell = 0,\, 1,\, \ldots,\, L, \tag{2.11}$$

$$\mathbf{U}^{(\ell)} = \text{diag}(\mathbf{U}_\tau : \tau \text{ is a box on level } \ell), \qquad \ell = 1,\, 2,\, \ldots,\, L, \tag{2.12}$$

$$\mathbf{V}^{(\ell)} = \text{diag}(\mathbf{V}_\tau : \tau \text{ is a box on level } \ell), \qquad \ell = 1,\, 2,\, \ldots,\, L, \tag{2.13}$$

$$\mathbf{B}^{(\ell)} = \text{diag}(\mathbf{B}_\tau : \tau \text{ is a box on level } \ell), \qquad \ell = 0,\, 1,\, \ldots,\, L-1,. \tag{2.14}$$

Furthermore, we let $\tilde{\mathbf{A}}^{(\ell)}$ denote the block matrix whose diagonal blocks are zero, and whose off-diagonal blocks are the blocks $\tilde{\mathbf{A}}_{\tau,\tau'}$ for all distinct τ, τ' on level ℓ. With these definitions,

$$\underset{n\,2^L \times n\,2^L}{\mathbf{A}} = \underset{n\,2^L \times k\,2^L}{\mathbf{U}^{(L)}} \quad \underset{k\,2^L \times k\,2^L}{\tilde{\mathbf{A}}^{(L)}} \quad \underset{k\,2^L \times n\,2^L}{(\mathbf{V}^{(L)})^*} + \underset{n\,2^L \times n\,2^L}{\mathbf{D}^{(L)}}; \tag{2.15}$$

for $\ell = L-1,\, L-2,\, \ldots,\, 1$ we have

$$\underset{k\,2^{\ell+1} \times k\,2^{\ell+1}}{\tilde{\mathbf{A}}^{(\ell+1)}} = \underset{k\,2^{\ell+1} \times k\,2^{\ell}}{\mathbf{U}^{(\ell)}} \quad \underset{k\,2^{\ell} \times k\,2^{\ell}}{\tilde{\mathbf{A}}^{(\ell)}} \quad \underset{k\,2^{\ell} \times k\,2^{\ell+1}}{(\mathbf{V}^{(\ell)})^*} + \underset{k\,2^{\ell+1} \times k\,2^{\ell+1}}{\mathbf{B}^{(\ell)}}; \tag{2.16}$$

and finally

$$\tilde{\mathbf{A}}^{(1)} = \mathbf{B}^{(0)}. \tag{2.17}$$

REMARK 2.2 (Matrix-vector multiplication). *The telescoping factorizations in Section 2.4 easily translate into a formula for evaluating the matrix-vector product* $\mathbf{u} = \mathbf{A}\mathbf{q}$ *once all factors in an HBS representation have been provided. Under the assumption that the tree is refined all the way do where the number of nodes in each leaf is comparable to the interactions ranks (so that* $n = O(k)$*), the asymptotic complexity of the matrix-vector multiplication is* $O(Nk)$*. This operation is summarized in Figure 2.3.*

2.5. Inversion of hierarchically block separable matrices

We are now ready to describe in detail how to invert an HBS matrix. The inversion algorithm turns out to be remarkably simple, as can be seen by peeking ahead at Figure 2.4. Moreover, as long as all intermediate matrices that need to be inverted are non-singular, the inversion formula is exact up to floating point arithmetic.

Technically, the scheme consists of recursive application of Lemma 1.1, and its derivation is given in the proof of the following theorem, cf. [**2**, Thm. 5.1].

THEOREM 2.1. *Let* $\mathbf{A}$ *be an invertible* $N \times N$ *HBS matrix with block-rank* k*. Suppose that the associated tree structure is uniform, and that each leaf box holds at most* $O(k)$ *nodes. Then a data-sparse representation of* $\mathbf{A}^{-1}$ *that is exact up to rounding errors can be computed in* $O(Nk^2)$ *operations via the process outlined in Figure 2.4, provided that none of the matrices that need to be inverted is singular. The computed inverse can be applied to a vector in* $O(Nk)$ *operations via the algorithm in Figure 2.5.*

Proof: We can according to equation (2.15) express an HBS matrix as

$$\mathbf{A} = \mathbf{U}^{(L)}\, \tilde{\mathbf{A}}^{(L)}\, (\mathbf{V}^{(L)})^* + \mathbf{D}^{(L)}.$$

Lemma 1.1 immediately applies, and we find that

$$\mathbf{A}^{-1} = \mathbf{E}^{(L)} \left(\tilde{\mathbf{A}}^{(L)} + \hat{\mathbf{D}}^{(L)}\right)^{-1} (\mathbf{F}^{(L)})^* + \mathbf{G}^{(L)}, \tag{2.18}$$

where $\hat{\mathbf{D}}^{(L)}$, $\mathbf{E}^{(L)}$, $\mathbf{F}^{(L)}$, and $\mathbf{G}^{(L)}$ are defined via (1.8), (1.9), (1.10), and (1.11).

loop over all leaf boxes τ
    $\hat{\mathbf{q}}_\tau = \mathbf{V}_\tau^* \, \mathbf{q}(I_\tau)$.
end loop

loop over levels, finer to coarser, $\ell = L-1,\, L-2,\, \ldots, 1$
    loop over all parent boxes τ on level ℓ,
        Let α and β denote the children of τ.
        $\hat{\mathbf{q}}_\tau = \mathbf{V}_\tau^* \begin{bmatrix} \hat{\mathbf{q}}_\alpha \\ \hat{\mathbf{q}}_\beta \end{bmatrix}$.
    end loop
end loop

$\hat{\mathbf{u}}_1 = 0$
loop over all levels, coarser to finer, $\ell = 1,\, 2,\, \ldots,\, L-1$
    loop over all parent boxes τ on level ℓ
        Let α and β denote the children of τ.
        $\begin{bmatrix} \hat{\mathbf{u}}_\alpha \\ \hat{\mathbf{u}}_\beta \end{bmatrix} = \mathbf{U}_\tau \, \hat{\mathbf{u}}_\tau + \begin{bmatrix} 0 & \mathbf{B}_{\alpha,\beta} \\ \mathbf{B}_{\beta,\alpha} & 0 \end{bmatrix} \begin{bmatrix} \hat{\mathbf{q}}_\alpha \\ \hat{\mathbf{q}}_\beta \end{bmatrix}$.
    end loop
end loop

loop over all leaf boxes τ
    $\mathbf{u}(I_\tau) = \mathbf{U}_\tau \, \hat{\mathbf{u}}_\tau + \mathbf{D}_\tau \, \mathbf{q}(I_\tau)$.
end loop

FIGURE 2.3. The HBS matrix-vector multiplication. Given a vector $\mathbf{q}$ and a matrix $\mathbf{A}$ in HBS format, compute $\mathbf{u} = \mathbf{A}\mathbf{q}$.

To move to the next coarser level, we set $\ell = L-1$ in formula (2.16) whence

$$\tilde{\mathbf{A}}^{(L)} + \hat{\mathbf{D}}^{(L)} = \mathbf{U}^{(L-1)} \, \tilde{\mathbf{A}}^{(L-1)} \, (\mathbf{V}^{(L-1)})^* + \mathbf{B}^{(L-1)} + \hat{\mathbf{D}}^{(L)}. \tag{2.19}$$

We define

$$\tilde{\mathbf{D}}^{(L-1)} = \mathbf{B}^{(L-1)} + \hat{\mathbf{D}}^{(L)} = \left[\begin{array}{cc|cc|cc|c} \hat{\mathbf{D}}_{\tau_1} & \mathbf{B}_{\tau_1\tau_2} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \cdots \\ \mathbf{B}_{\tau_1\tau_2} & \hat{\mathbf{D}}_{\tau_2} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \cdots \\ \hline \mathbf{0} & \mathbf{0} & \hat{\mathbf{D}}_{\tau_3} & \mathbf{B}_{\tau_3\tau_4} & \mathbf{0} & \mathbf{0} & \cdots \\ \mathbf{0} & \mathbf{0} & \mathbf{B}_{\tau_3\tau_4} & \hat{\mathbf{D}}_{\tau_4} & \mathbf{0} & \mathbf{0} & \cdots \\ \hline \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \hat{\mathbf{D}}_{\tau_5} & \mathbf{B}_{\tau_5\tau_6} & \cdots \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{B}_{\tau_5\tau_6} & \hat{\mathbf{D}}_{\tau_6} & \cdots \\ \hline \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \end{array}\right],$$

where $\{\tau_1,\, \tau_2,\, \ldots,\, \tau_{2^L}\}$ is a list of the boxes on level L. Equation (2.19) then takes the form

$$\tilde{\mathbf{A}}^{(L)} + \hat{\mathbf{D}}^{(L)} = \mathbf{U}^{(L-1)} \, \tilde{\mathbf{A}}^{(L-1)} \, (\mathbf{V}^{(L-1)})^* + \tilde{\mathbf{D}}^{(L-1)}. \tag{2.20}$$

(We note that in (2.20), the terms on the left hand side are block matrices with $2^L \times 2^L$ blocks, each of size $k \times k$, whereas the terms on the right hand side are block matrices with $2^{L-1} \times 2^{L-1}$ blocks, each of size $2k \times 2k$.) Now Lemma 1.1 applies to (2.20) and we find that

$$\big(\tilde{\mathbf{A}}^{(L)} + \hat{\mathbf{D}}^{(L)}\big)^{-1} = \mathbf{E}^{(L-1)} \, \big(\tilde{\mathbf{A}}^{(L-1)} + \hat{\mathbf{D}}^{(L-1)}\big)^{-1} \, (\mathbf{F}^{(L-1)})^* + \mathbf{G}^{(L-1)},$$

loop over all levels, finer to coarser, $\ell = L,\ L-1,\ \ldots, 1$
  loop over all boxes τ on level ℓ,
    if τ is a leaf node
      $\tilde{\mathbf{D}}_\tau = \mathbf{D}_\tau$
    else
      Let α and β denote the children of τ.
      $\tilde{\mathbf{D}}_\tau = \begin{bmatrix} \hat{\mathbf{D}}_\alpha & \mathbf{B}_{\alpha,\beta} \\ \mathbf{B}_{\beta,\alpha} & \hat{\mathbf{D}}_\beta \end{bmatrix}$
    end if
    $\hat{\mathbf{D}}_\tau = \left(\mathbf{V}_\tau^* \, \tilde{\mathbf{D}}_\tau^{-1} \, \mathbf{U}_\tau\right)^{-1}$.
    $\mathbf{E}_\tau = \tilde{\mathbf{D}}_\tau^{-1} \, \mathbf{U}_\tau \, \hat{\mathbf{D}}_\tau$.
    $\mathbf{F}_\tau^* = \hat{\mathbf{D}}_\tau \, \mathbf{V}_\tau^* \, \tilde{\mathbf{D}}_\tau^{-1}$.
    $\mathbf{G}_\tau = \hat{\mathbf{D}}_\tau - \tilde{\mathbf{D}}_\tau^{-1} \, \mathbf{U}_\tau \, \hat{\mathbf{D}}_\tau \, \mathbf{V}_\tau^* \, \tilde{\mathbf{D}}_\tau^{-1}$.
  end loop
end loop
$\mathbf{G}_1 = \begin{bmatrix} \hat{\mathbf{D}}_2 & \mathbf{B}_{2,3} \\ \mathbf{B}_{3,2} & \hat{\mathbf{D}}_3 \end{bmatrix}^{-1}$.

FIGURE 2.4. Inversion of an HBS matrix.

where $\hat{\mathbf{D}}^{(L-1)}$, $\mathbf{E}^{(L-1)}$, $\mathbf{F}^{(L-1)}$, and $\mathbf{G}^{(L-1)}$ are defined via (1.8), (1.9), (1.10), and (1.11).

The process by which we went from step L to step $L-1$ is then repeated to move up to coarser and coarser levels. With each step, the size of the matrix to be inverted is cut in half. Once we get to the top level, we are left with the task of inverting the matrix

$$\tilde{\mathbf{A}}^{(1)} + \hat{\mathbf{D}}^{(1)} = \begin{bmatrix} \hat{\mathbf{D}}_2 & \mathbf{B}_{2,3} \\ \mathbf{B}_{3,2} & \hat{\mathbf{D}}_3 \end{bmatrix} \tag{2.21}$$

The matrix in (2.21) is of size $2k \times 2k$, and we use brute force to evaluate

$$\mathbf{G}^{(0)} = \mathbf{G}_1 = \begin{bmatrix} \hat{\mathbf{D}}_2 & \mathbf{B}_{2,3} \\ \mathbf{B}_{3,2} & \hat{\mathbf{D}}_3 \end{bmatrix}^{-1}.$$

The proof that the asymptotic cost of the method is $O(Nk^2)$ is given in Section 2.6.1. □

REMARK 2.3. *The HBS inversion algorithm in Figure 2.4 provides four formulas for the matrices* $\{\mathbf{E}_\tau,\ \mathbf{F}_\tau,\ \mathbf{G}_\tau,\ \hat{\mathbf{D}}_\tau\}_\tau$. *The task of actually computing the matrices can be accelerated in two ways: (1) Several of the matrix-matrix products in the formulas are recurring, and the computation should be organized so that each matrix-matrix product is evaluated only once. (2) When interpolatory decompositions are used, multiplications involving the matrices* $\mathbf{U}_\tau$ *and* $\mathbf{V}_\tau$ *can be accelerated by exploiting that they each contain a* $k \times k$ *identity matrix.*

REMARK 2.4. *The assumption in Theorem 2.1 that none of the matrices to be inverted is singular is undesirable. When* $\mathbf{A}$ *is spd, this assumption can be done away with (cf. Corollary 3.1), and it can further be proved that the inversion process is numerically stable.*

2.6. Asymptotic complexity of the Woodbury formula for an HBS matrix

It is time to estimate the asymptotic cost of the HBS inversion scheme summarized in Figure 2.4. We start in Section 2.6.1 by analyzing the case where the interaction ranks are the same at every level in the

loop over all leaf boxes τ
    $\hat{\mathbf{u}}_\tau = \mathbf{F}_\tau^* \, \mathbf{u}(I_\tau)$.
end loop

loop over all levels, finer to coarser, $\ell = L,\, L-1,\, \ldots, 1$
    loop over all parent boxes τ on level ℓ,
        Let α and β denote the children of τ.
        $\hat{\mathbf{u}}_\tau = \mathbf{F}_\tau^* \begin{bmatrix} \hat{\mathbf{u}}_\alpha \\ \hat{\mathbf{u}}_\beta \end{bmatrix}$.
    end loop
end loop

$\begin{bmatrix} \hat{\mathbf{q}}_2 \\ \hat{\mathbf{q}}_3 \end{bmatrix} = \hat{\mathbf{G}}_1 \begin{bmatrix} \hat{\mathbf{u}}_2 \\ \hat{\mathbf{u}}_3 \end{bmatrix}$.

loop over all levels, coarser to finer, $\ell = 1,\, 2,\, \ldots,\, L-1$
    loop over all parent boxes τ on level ℓ
        Let α and β denote the children of τ.
        $\begin{bmatrix} \hat{\mathbf{q}}_\alpha \\ \hat{\mathbf{q}}_\beta \end{bmatrix} = \mathbf{E}_\tau \, \hat{\mathbf{u}}_\tau + \mathbf{G}_\tau \begin{bmatrix} \hat{\mathbf{u}}_\alpha \\ \hat{\mathbf{u}}_\beta \end{bmatrix}$.
    end loop
end loop

loop over all leaf boxes τ
    $\mathbf{q}(I_\tau) = \mathbf{E}_\tau \, \hat{\mathbf{q}}_\tau + \mathbf{G}_\tau \, \mathbf{u}(I_\tau)$.
end loop

FIGURE 2.5. Application of the inverse of an HBS matrix. Given a vector $\mathbf{u}$, compute $\mathbf{q} = \mathbf{A}^{-1} \, \mathbf{u}$ using the compressed representation of $\mathbf{A}^{-1}$ resulting from the algorithm in Figure 2.4.

hierarchy, which is the basic assumption underlying the entire discussion so far in this chapter. In Sections 2.6.2 and 2.6.3, we then analyze the cases where the interaction ranks increase as the sizes of the boxes grow.

Throughout this section, we stick with the assumption that the tree is uniform, that it has L levels, and that there are roughly equally many nodes in each leaf box.

2.6.1. Constant interaction ranks. Let us assume that there is a uniform upper bound k on the HBS rank of all boxes at all levels, conforming to the assumption in Theorem 2.1 Let us further assume that each leaf box holds at most $2k$ nodes (the argument of course remains unchanged for $O(k)$ nodes in each leaf box).

We first consider the cost of constructing the matrices $\hat{\mathbf{D}}_\tau$, $\mathbf{E}_\tau$, $\mathbf{F}_\tau$, $\mathbf{G}_\tau$ for all boxes τ on a fixed level ℓ. To construct one of these matrices, we must perform dense matrix algebra (matrix-matrix multiply, matrix inversion, etc.) on matrices of size at most $2k \times 2k$. Since there are 2^ℓ nodes on level ℓ, we find:

$$\text{cost of processing level } \ell \sim 2^\ell \, k^3.$$

Now sum over all the L levels to find

$$\text{total cost} \sim \sum_{\ell=1}^{L} 2^\ell \, k^3 \sim 2^L \, k^3.$$

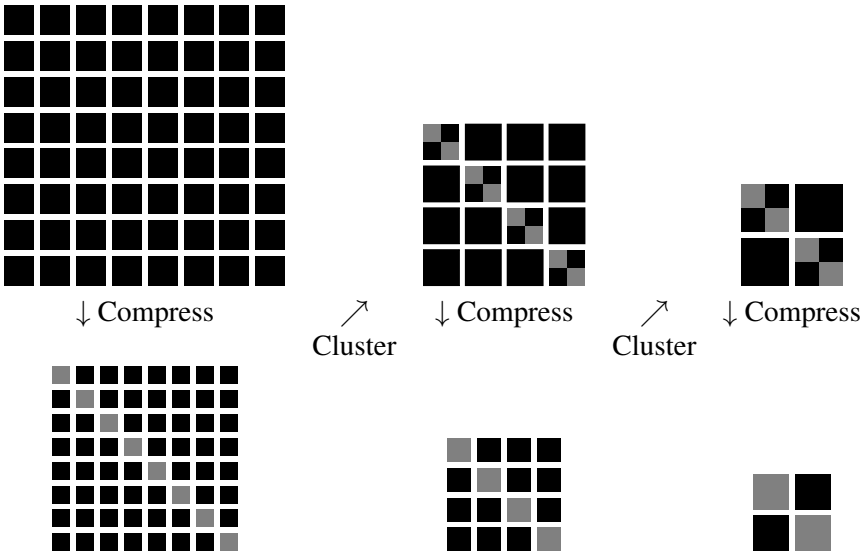


FIGURE 2.6. Cartoon of the hierarchical compression strategy described in Section 2.5. Gray blocks represent the matrices $\hat{\mathbf{D}}_\tau$. These must be explicitly constructed. All black blocks represent sub-matrices of the original matrix.

Finally observe that since there are 2^L leaf boxes, and each leaf box holds about $2k$ points, we have $N \approx 2^{L+1}k$. In consequence, the total cost of the inversion procedure is $O(Nk^2)$.

2.6.2. Logarithmic growth of interaction ranks. Let us next suppose that the rank of a block is proportional to the logarithm of the number of nodes in the block. This is often a realistic assumption for a matrix $\mathbf{A}$ arising from discretization of a boundary integral operator on a contour in the plane. If we let n denote the number of nodes in a leaf box, we find that there are $2^{(L-\ell)}n$ nodes in a box at level ℓ, and consequently the interaction rank k_ℓ at level ℓ would satisfy

$$k_\ell \sim \log(n2^{(L-\ell)}) = \log(n) + (L-\ell)\log(2).$$

Since there are $(N/n)2^{(L-\ell)}$ boxes at level ℓ, we then find a total cost

$$T \sim \sum_{\ell=1}^{L} \frac{N}{n} 2^{(\ell-L)} \big(\log(n) + (L-\ell)\log(2)\big)^3 \sim \frac{N}{n} \sum_{j=0}^{L-1} 2^{-j} \big(\log(n) + j\log(2)\big)^3 \sim \frac{N}{n}(\log(n))^3.$$

The last similarity holds since

$$\begin{aligned}\sum_{j=0}^{L-1} 2^{-j}\big(\log(n) + j\log(2)\big)^3 &\leq \sum_{j=0}^{\infty} 2^{-j}\big(\log(n) + j\log(2)\big)^3 \\ &= \sum_{j=0}^{\infty} 2^{-j}\big(\log(n)^3 + 3\log(n)^2 j\log(2) + 3\log(n)j^2\log(2)^2 + j^3\log(2)^3\big),\end{aligned}$$

and in each term, the polynomial growth in j does not prevent that fast exponential decay from making the sum converge. Since we may pick $n = O(1)$, it follows that the total cost is $O(N)$ in this case as well. See **[6, 2, 3]** for details.

2.6.3. Algebraic growth of interaction ranks. As our final case, let us suppose that the ranks grow as a power of the number of nodes in a box. To be precise, let m_ℓ denote the number of nodes in a box at level

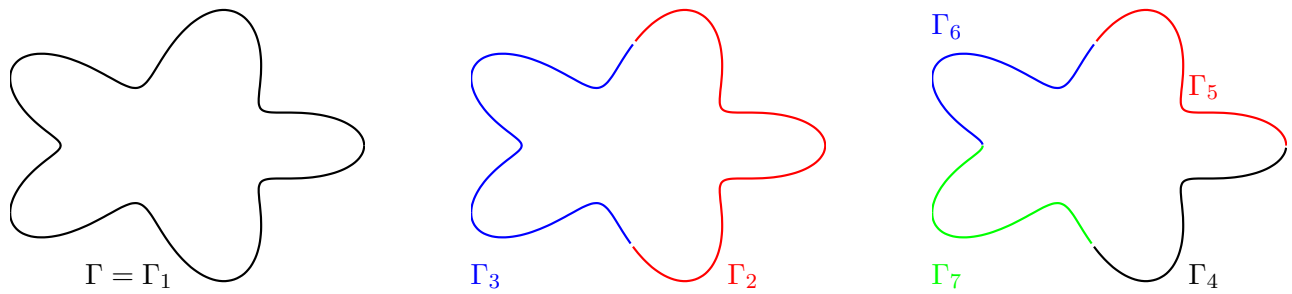


FIGURE 2.7. Hej

ℓ so that $m_\ell \sim N2^{-\ell}$, and then suppose that the interaction ranks k_ℓ at level ℓ satisfy

$$k_\ell \sim m_\ell^\alpha,$$

for some $\alpha \in (0, 1)$. The total computational cost is then

$$T \sim \sum_{\ell=0}^{L} 2^\ell m_\ell^3 \sim \sum_{\ell=0}^{L} 2^\ell (N2^{-\ell})^3 \sim N^{3\alpha} \sum_{\ell=0}^{L} 2^{\ell(1-3\alpha)} \sim \{\text{Set } \beta = 2^{(1-3\alpha)}\} \sim N^{3\alpha} \sum_{\ell=0}^{L} \beta^\ell.$$

We see that the cost estimate is a geometric sum that is dominated either by the root computation or by the leaf computation, depending on whether β is smaller or larger than 1. We get three cases:

$\boxed{\alpha > 1/3:}$ Now $\beta > 1$, the sum is dominated by the root, and $T \sim N^{3\alpha}$.

$\boxed{\alpha = 1/3:}$ Now $\beta = 1$, and $T \sim NL \sim N \log N$.

$\boxed{\alpha < 1/3:}$ Now $\beta < 1$, the sum is dominated by the leaves, and $T \sim N$, since we may pick $n = O(1)$.

2.7. A numerical example

To close this chapter, let us check what the HBS ranks actually turn out to be for typical BIEs associated with a contour Γ in two dimension. We consider the simple geometry shown in Figure 2.7, and the two integral operators

$$[C\sigma](\boldsymbol{x}) = \int_\Gamma \big(d(\boldsymbol{x}, \boldsymbol{y}) + s(\boldsymbol{x}, \boldsymbol{y})\big)\, \sigma(\boldsymbol{y})\, ds(\boldsymbol{y}), \tag{2.22}$$

$$[C_\kappa\sigma](\boldsymbol{x}) = \int_\Gamma \big(d_\kappa(\boldsymbol{x}, \boldsymbol{y}) + i\kappa s_\kappa(\boldsymbol{x}, \boldsymbol{y})\big)\, \sigma(\boldsymbol{y})\, ds(\boldsymbol{y}), \tag{2.23}$$

where s and d are the single and double layer kernels associated with the Laplace equation, and where s_κ and d_κ are the corresponding kernels associated with the Helmholtz equation. We build a matrix $\mathbf{A}$ by discretizing either (2.22) or (2.23) using a Nyström scheme associated with a panel based quadrature with 40 panels, with 10 Lagrangian interpolation nodes on each panel, so that the matrix ends up being of size 400×400. (We use the Alpert method for accurately discretizing the singular kernels.)

Figure 2.8 shows the singular values of off-diagonal blocks of the matrix $\mathbf{A}$ associated with three different integral operators. Left column: The Laplace operator (2.22). Middle and right columns: The Helmholtz operator (2.23) for κ chosen so that the diameter of the contour is 3λ and 30λ respectively. The plots in the top row show the singular values of the 200×200 matrix $\mathbf{A}(I_2, I_2^c)$ while the bottom row shows the corresponding singular values of the 100×300 matrix $\mathbf{A}(I_4, I_4^c)$.

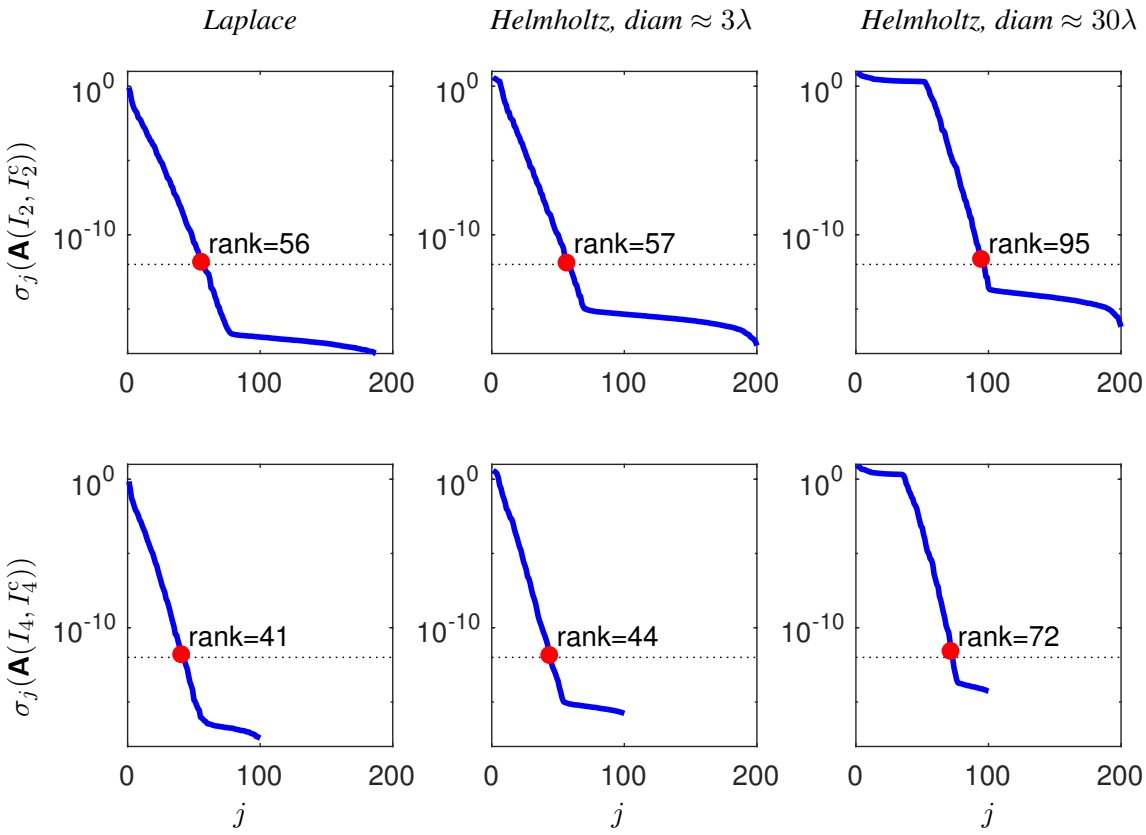


FIGURE 2.8. Singular values of off-diagonal blocks that arise in the compression of an HBS matrix $\mathbf{A}$, as described in Section 2.7. The matrix $\mathbf{A}$ is obtained from Nyström discretization of the boundary integral operators (2.22) and (2.23), on the contour Γ shown in Figure 2.7. The top row holds plots of the singular values of $\mathbf{A}(I_2, I_2^c)$ and the bottom row holds the corresponding plots for $\mathbf{A}(I_4, I_4^c)$. The ranks shown are to absolute precision 10^{-12}.

For small wave numbers, we observe first of all that there is not much difference between the Laplace case and the Helmholtz case. Moreover, the ranks do not appear to change much with the size of the patch being compressed. When the wave-number increases, however, we see both that the ranks grow substantially, and also that there is a meaningful difference between the ranks of the blocks $\mathbf{A}(I_2, I_2^c)$ and $\mathbf{A}(I_4, I_4^c)$. This conforms with the discussion in Section ???, where we claimed that the "knee" that we see in two rightmost graphs in Figure 2.8 is located at $k_{\text{knee}} \sim \kappa D$, where D is the diameter of the patch.

CHAPTER 3

Additional topics on HBS matrices

Our objective in Chapter 2 was to provide a complete description of an $O(N)$ algorithm for inverting an HBS matrix that was not unduly burdened by technical minutiae. In this chapter, we return to some of the topics that were introduced in Chapter 2 for a more detailed discussion. The issues here are important when implementing a direct solver in an actual application, but can be skipped by a reader looking for a more conceptual introduction to direct solvers for integral equations.

3.1. Error estimation

When the direct solver described in Chapters 1 and 2 is used to solve a linear system

$$\mathbf{A}\mathbf{q} = \mathbf{f},$$

one goes through the following steps:

(1) Given the input matrix $\mathbf{A}$, one by some means find an HBS approximation $\mathbf{A}_{\text{approx}}$ such that

$$\mathbf{A}_{\text{approx}} \approx \mathbf{A}.$$

(2) The HBS matrix $\mathbf{A}_{\text{approx}}$ is inverted using the method in Figure 2.4 to build a matrix $\mathbf{B}$ such that

$$\mathbf{B} = \mathbf{A}_{\text{approx}}^{-1}.$$

(3) An approximate solution is computed using the formula

$$\mathbf{q}_{\text{approx}} = \mathbf{B}\mathbf{f}.$$

After the process has completed, what is the accuracy of the computed solution $\mathbf{q}_{\text{approx}}$? There are several paths whereby errors may by introduced. First of all, many techniques for compressing a matrix are based on heuristics to some degree, and it can be difficult to know exactly how accurately the matrix $\mathbf{A}_{\text{approx}}$ captures $\mathbf{A}$. Second, even if we know that, say

$$\|\mathbf{A} - \mathbf{A}_{\text{approx}}\| \leq \varepsilon$$

for some small ε, it does not necessarily follow that $\mathbf{A}_{\text{approx}}^{-1}$ is within distance ε of $\mathbf{A}^{-1}$. If $\mathbf{A}$ is well-conditioned, then a relation such as

$$\mathbf{A}^{-1} - \mathbf{A}_{\text{approx}}^{-1} = \mathbf{A}^{-1}\big(\mathbf{A}_{\text{approx}} - \mathbf{A}\big)\mathbf{A}_{\text{approx}}^{-1}$$

may help us bound the error, but we often face a situation where $\mathbf{A}^{-1}$ may be large. Finally, while the HBS inversion procedure in Figure 2.4 is in principle mathematically exact, round-off errors can potentially throw things off, in particular when some of the intermediate matrices that need to be inverted are ill-conditioned. Given the many sources of errors, it is challenging to attempt to bound the resulting error *á priori,* and to the extent that guaranteed bounds can be derived, they tend to be very pessimistic. For these reasons, it is often more productive to estimate the error in the actual computed solution. The easiest technique for doing this is of course to simply compute the residual $\mathbf{A}\mathbf{q}_{\text{approx}} - \mathbf{f}$ and check if it is small enough. We often have techniques available for applying $\mathbf{A}$ rapidly, using, for instance an FMM executed at high accuracy.

To get a handle on the error in the direct solver itself (as opposed to the error for a specific data vector $\mathbf{f}$), it can be useful to estimate the quantity

$$e = \|\mathbf{I} - \mathbf{BA}\|$$

that provides a measure of how good of an approximate inverse the computed object $\mathbf{B}$ is to $\mathbf{A}$. The number e can often be estimated at moderate cost by executing a power iteration on the matrix $\big(\mathbf{I} - \mathbf{BA}\big)\big(\mathbf{I} - \mathbf{A}^*\mathbf{B}^*\big)$, exploiting the fact that we can very rapidly apply both $\mathbf{B}$ and $\mathbf{B}^*$ to vectors since we have them available in an HBS format, and using a standard fast summation scheme such as the FMM to apply $\mathbf{A}$ and $\mathbf{A}^*$. Once e has been computed, we immediately get bounds such as

$$\frac{\|\mathbf{A} - \mathbf{B}\|}{\|\mathbf{A}^{-1}\|} = \frac{\|\big(\mathbf{I} - \mathbf{BA}\big)\mathbf{A}^{-1}\|}{\|\mathbf{A}^{-1}\|} \leq \frac{\|\mathbf{I} - \mathbf{BA}\|\,\|\mathbf{A}^{-1}\|}{\|\mathbf{A}^{-1}\|} = e.$$

Or, analogously,

$$\|\mathbf{q} - \mathbf{q}_{\text{approx}}\| = \|\mathbf{q} - \mathbf{Bf}\| = \|\mathbf{q} - \mathbf{BAq}\| = \|\big(\mathbf{I} - \mathbf{BA}\big)\mathbf{q}\| \leq \|\mathbf{I} - \mathbf{BA}\|\,\|\mathbf{q}\| = e\,\|\mathbf{q}\|.$$

3.2. Symmetric positive definite matrices

The algorithm for inversion of an HBS matrix in Figure 2.4 requires the inversion of a large number of small intermediate matrices. It is in general not easy to prove that these are necessarily non-singular, or to say anything about their condition numbers. One exception is the case where $\mathbf{A}$ is symmetric positive definite, in which case we have the following theorem:

COROLLARY 3.1. *Let $\mathbf{A}$ be a symmetric positive definite (spd) matrix that admits a factorization*

$$\underset{N\times N}{\mathbf{A}} = \underset{N\times K}{\mathbf{U}}\ \underset{K\times K}{\tilde{\mathbf{A}}}\ \underset{K\times N}{\mathbf{U}^*} + \underset{N\times N}{\mathbf{D}}, \tag{3.1}$$

where $\ker(\mathbf{U}) = \{\mathbf{0}\}$ *and $\mathbf{D}$ is a block diagonal submatrix of $\mathbf{A}$. Then the matrices $\mathbf{D}$, $\big(\mathbf{U}^*\,\mathbf{D}^{-1}\,\mathbf{U}\big)$, and $\tilde{\mathbf{A}} + (\mathbf{U}^*\,\mathbf{D}^{-1}\mathbf{U})^{-1}$ are spd (and hence invertible).*

PROOF. That $\mathbf{D}$ is spd is an immediate consequence of the fact that it is a block diagonal submatrix of a spd matrix. To prove that $\mathbf{U}^*\,\mathbf{D}^{-1}\,\mathbf{U}$ is spd, let us pick any $\mathbf{x} \neq \mathbf{0}$. Setting $\mathbf{y} = \mathbf{D}^{-1}\,\mathbf{U}\,\mathbf{x}$, we then find that

$$\mathbf{x}^*\big(\mathbf{U}^*\mathbf{D}^{-1}\mathbf{U}\big)\mathbf{x} = \mathbf{x}^*\mathbf{U}^*\mathbf{D}^{-1}\mathbf{D}\mathbf{D}^{-1}\mathbf{U}\mathbf{x} = \big(\mathbf{D}^{-1}\mathbf{U}\mathbf{x}\big)^*\mathbf{D}\big(\mathbf{D}^{-1}\mathbf{U}\mathbf{x}\big) = \mathbf{y}^*\mathbf{D}\mathbf{y} > 0$$

since $\mathbf{D}$ is spd. (Observe that $\mathbf{y} \neq \mathbf{0}$ since $\ker(\mathbf{U}) = \{\mathbf{0}\}$.)

It remains only to prove that $\tilde{\mathbf{A}} + (\mathbf{U}^*\mathbf{D}^{-1}\mathbf{U})^{-1}$ is spd. To this end, define $\hat{\mathbf{D}}$ and $\mathbf{E}$ via

$$\hat{\mathbf{D}} = (\mathbf{U}^*\mathbf{D}^{-1}\mathbf{U})^{-1}$$
$$\mathbf{E} = \mathbf{D}^{-1}\mathbf{U}\hat{\mathbf{D}}.$$

Then

$$\begin{aligned}\tilde{\mathbf{A}} + \hat{\mathbf{D}} &= \hat{\mathbf{D}}\,\Big(\hat{\mathbf{D}}^{-1}\,\tilde{\mathbf{A}}\,\hat{\mathbf{D}}^{-1} + \hat{\mathbf{D}}^{-1}\Big)\,\hat{\mathbf{D}} = \hat{\mathbf{D}}\,\Big(\mathbf{U}^*\,\mathbf{D}^{-1}\,\mathbf{U}\,\tilde{\mathbf{A}}\,\mathbf{U}^*\mathbf{D}^{-1}\,\mathbf{U} + \hat{\mathbf{D}}^{-1}\Big)\,\hat{\mathbf{D}} \\ &= \hat{\mathbf{D}}\,\big(\mathbf{U}^*\,\mathbf{D}^{-1}\,(\mathbf{A} - \mathbf{D})\mathbf{D}^{-1}\,\mathbf{U} + \mathbf{U}^*\mathbf{D}^{-1}\mathbf{U}\big)\,\hat{\mathbf{D}} = \hat{\mathbf{D}}\,\mathbf{U}^*\,\mathbf{D}^{-1}\,\mathbf{A}\mathbf{D}^{-1}\,\mathbf{U}\hat{\mathbf{D}} = \mathbf{E}^*\mathbf{A}\mathbf{E}.\end{aligned}$$

That $\tilde{\mathbf{A}} + (\mathbf{U}^*\mathbf{D}^{-1}\mathbf{U})^{-1}$ is spd now follows since $\ker(\mathbf{E}) = \{\mathbf{0}\}$ and $\mathbf{A}$ is spd. □

The proof of Corollary 3.1 demonstrates that for an spd matrix $\mathbf{A}$, the stability of the inversion formula (1.7) can be assessed by tracking the conditioning of the matrices $\mathbf{E}$. If these matrices are well-conditioned,

then the condition number of $\tilde{\mathbf{A}} + \hat{\mathbf{D}}$ cannot be substantially larger than the condition number of $\mathbf{A}$. To make this statement precise, let $\nu_{\min}$ and $\nu_{\max}$ denote the smallest and largest singular values of $\mathbf{E}$, so that

$$\nu_{\min} = \inf_{\|\mathbf{x}\|=1} \|\mathbf{E}\mathbf{x}\|, \qquad \text{and} \qquad \nu_{\max} = \sup_{\|\mathbf{x}\|=1} \|\mathbf{E}\mathbf{x}\|.$$

Moreover, let $\lambda_{\min}$ and $\lambda_{\max}$ denote the smallest and the largest eigenvalues of $\mathbf{A}$, so that

$$\lambda_{\min} = \inf_{\|\mathbf{x}\|=1} \mathbf{x}^*\mathbf{A}\mathbf{x} \qquad \text{and} \qquad \lambda_{\max} = \sup_{\|\mathbf{x}\|=1} \mathbf{x}^*\mathbf{A}\mathbf{x}.$$

We then find that

$$\mathbf{x}^*\big(\tilde{\mathbf{A}} + \hat{\mathbf{D}}\big)\mathbf{x} = \mathbf{x}^*\mathbf{E}^*\mathbf{A}\mathbf{E}\mathbf{x} \geq \lambda_{\min} \|\mathbf{E}\mathbf{x}\|^2 \geq \lambda_{\min}\, \nu_{\min}^2 \|\mathbf{x}\|^2,$$

which shows that the smallest eigenvalue of $\tilde{\mathbf{A}} + \hat{\mathbf{D}}$ can be no smaller than $\nu_{\min}^2\, \lambda_{\min}$. Analogously,

$$\|\tilde{\mathbf{A}} + \hat{\mathbf{D}}\| = \|\mathbf{E}^*\mathbf{A}\mathbf{E}\| \leq \|\mathbf{E}\|^2 \|\mathbf{A}\| \leq \nu_{\max}^2\, \lambda_{\max},$$

which shows that the largest eigenvalue of $\tilde{\mathbf{A}} + \hat{\mathbf{D}}$ is bounded by $\nu_{\max}^2\, \lambda_{\max}$. To summarize, the spectrum of $\tilde{\mathbf{A}} + \hat{\mathbf{D}}$ is constrained to an interval:

$$\sigma(\tilde{\mathbf{A}} + \hat{\mathbf{D}}) \subset \big[\nu_{\min}^2 \lambda_{\min},\, \nu_{\max}^2 \lambda_{\max}\big].$$

It follows that the condition number of $\tilde{\mathbf{A}} + \hat{\mathbf{D}}$ satisfies

$$\kappa(\tilde{\mathbf{A}} + \hat{\mathbf{D}}) \leq \kappa(\mathbf{E})^2 \kappa(\mathbf{A}).$$

In the particular case where the singular values of $\mathbf{E}$ are all of roughly the same size, we have $\kappa(\mathbf{E}) \approx 1$. This substantiates our claim that in this case, the condition number of $\tilde{\mathbf{A}} + \hat{\mathbf{D}}$ is approximately bounded by that of $\mathbf{A}$.

3.3. Orthonormal basis matrices

The definition of the HBS property in Section 2.3 is flexible in the sense that we did not enforce any conditions on the factors $\mathbf{U}_\tau$, $\mathbf{V}_\tau$, and $\tilde{\mathbf{A}}_{\tau,\tau'}$ other than that (2.6) and (2.8) must hold. From a purely algebraic point of view, the conditions we imposed are sufficient in themselves. However, for numerical stability it is as always very helpful to work with basis matrices that are well-conditioned. This is useful to maintain high accuracy when performing a matrix-vector multiplication involving an HBS matrix, and is essential when inverting them [**7, 9, 8**].

The most stringent definition of a well-conditioned basis is that the columns should be orthonormal. If we are given any data-sparse representation of an HBS matrix, it turns out to be relatively straight-forward to convert it to one that involves orthonormal matrices $\mathbf{U}_\tau$ and $\mathbf{V}_{\tau'}$ in (2.6) and (2.8), as shown in the algorithm in Figure 3.1. This algorithm also exploits the fact that the HBS format has some amount of redundancies to convert all sibling interaction matrices $\mathbf{B}_{\tau,\tau'}$ to diagonal matrices, in effect turning (2.6) and (2.8) into singular value decompositions. This improves the compression rate of the HBS representation, and also provides a slight boost in computational speed since multiplication by a diagonal matrix is very fast.

While orthonormality is ideal from the point of view of conditioning of a basis, it is often convenient to relax this condition and allow other well-conditioned bases to be used. In particular, we will in Chapter 4 demonstrate that using the interpolatory decomposition (the ID) in the factorizations (2.6) and (2.8) is often very effective, in particular for gaining high speed when computing the compressed representation in the first place. When the ID is used, every $\mathbf{U}_\tau$ and $\mathbf{V}_\tau$ contains a $k \times k$ identity matrix, which mildly accelerates computations, and reduces storage requirements. Moreover, each $\tilde{\mathbf{A}}_{\tau,\tau'}$ is a submatrix of the original matrix $\mathbf{A}$. In situations where matrix entries can easily be computed on the fly, this means that we only need to store the index vectors that identify the submatrix, rather than the matrix entries themselves.

loop over levels, finer to coarser, $\ell = L-1,\, L-2,\, \ldots,\, 0$
  loop over all parent boxes τ on level ℓ,
    Let α and β denote the children of τ.
    $[\mathbf{U}_1,\, \mathbf{R}_1] = \mathtt{qr}(\mathbf{U}_\alpha)$. $[\mathbf{U}_2,\, \mathbf{R}_2] = \mathtt{qr}(\mathbf{U}_\beta)$.
    $[\mathbf{V}_1,\, \mathbf{S}_1] = \mathtt{qr}(\mathbf{V}_\alpha)$. $[\mathbf{V}_2,\, \mathbf{S}_2] = \mathtt{qr}(\mathbf{V}_\beta)$.
    $[\mathbf{X}_1,\, \mathbf{\Sigma}_{12},\, \mathbf{Y}_2] = \mathtt{svd}(\mathbf{R}_1\, \tilde{\mathbf{A}}_{\alpha,\beta}\, \mathbf{S}_2^*)$. $[\mathbf{X}_2,\, \mathbf{\Sigma}_{21},\, \mathbf{Y}_1] = \mathtt{svd}(\mathbf{R}_2\, \tilde{\mathbf{A}}_{\beta,\alpha}\, \mathbf{S}_1^*)$.
    $\mathbf{U}_\alpha \leftarrow \mathbf{U}_1\, \mathbf{X}_1$. $\mathbf{U}_\beta \leftarrow \mathbf{U}_2\, \mathbf{X}_2$.
    $\mathbf{V}_\alpha \leftarrow \mathbf{V}_1\, \mathbf{Y}_1$. $\mathbf{V}_\beta \leftarrow \mathbf{V}_2\, \mathbf{Y}_2$.
    $\mathbf{B}_{\alpha\beta} \leftarrow \mathbf{\Sigma}_{12}$. $\mathbf{B}_{\beta\alpha} \leftarrow \mathbf{\Sigma}_{21}$.
    if $\ell > 0$
      $\mathbf{U}_\tau \leftarrow \begin{bmatrix} \mathbf{X}_1^*\, \mathbf{R}_1 & 0 \\ 0 & \mathbf{X}_2^*\, \mathbf{R}_2 \end{bmatrix} \mathbf{U}_\tau$. $\mathbf{V}_\tau \leftarrow \begin{bmatrix} \mathbf{Y}_1^*\, \mathbf{S}_1 & 0 \\ 0 & \mathbf{Y}_2^*\, \mathbf{S}_2 \end{bmatrix} \mathbf{V}_\tau$.
    end if
  end loop
end loop

FIGURE 3.1. Reformating an HBS matrix: Given the factors $\mathbf{U}_\tau$, $\mathbf{V}_\tau$, $\tilde{\mathbf{A}}_{\alpha,\beta}$, $\mathbf{D}_\tau$ of an HBS matrix in general format, this algorithm computes new factors (that overwrite the old) such that all $\mathbf{U}_\tau$ and $\mathbf{V}_\tau$ are orthonormal, and all $\tilde{\mathbf{A}}_{\alpha,\beta}$ are diagonal.

loop over all levels, coarser to finer, $\ell = 0,\, 1,\, 2,\, \ldots, L-1$
  loop over all parent boxes τ on level ℓ,
    Let α and β denote the children of τ.
    Define the matrices $\mathbf{H}_{1,1}$, $\mathbf{B}_{\alpha,\beta}$, $\mathbf{B}_{\beta,\alpha}$, $\mathbf{H}_{2,2}$ so that
    $\mathbf{G}_\tau = \begin{bmatrix} \mathbf{H}_{1,1} & \mathbf{B}_{\alpha,\beta} \\ \mathbf{B}_{\beta,\alpha} & \mathbf{H}_{2,2} \end{bmatrix}$.
    $\mathbf{G}_\alpha \leftarrow \mathbf{G}_\alpha + \mathbf{E}_\alpha\, \mathbf{H}_{1,1}\, \mathbf{F}_\alpha^*$.
    $\mathbf{G}_\beta \leftarrow \mathbf{G}_\beta + \mathbf{E}_\beta\, \mathbf{H}_{2,2}\, \mathbf{F}_\beta^*$.
  end loop
end loop

loop over all leaf boxes τ
  $\mathbf{D}_\tau = \mathbf{G}_\tau$.
end loop

FIGURE 3.2. Reformatting of an HBS inverse. The method described here takes as input the factors in the data-sparse representation of an inverse resulting from the method in Figure 2.4 to obtain an inverse in the standard HBS format.

3.4. Reformatting the inverse of an HBS matrix to an HBS matrix

The HBS inversion algorithm in Figure 2.4 produces a representation of $\mathbf{A}^{-1}$ that is not exactly in HBS form since the matrices $\mathbf{G}_\tau$ do not have zero diagonal blocks like the matrices $\mathbf{B}_\tau$, *cf.* (2.10). However, a simple technique given in Figure 3.2 converts the factorization provided by the HBS inversion algorithm into a standard HBS factorization. If a factorization in which the expansion matrices are all orthonormal is sought, then further post-processing via Algorithm 3.1 will do the job.

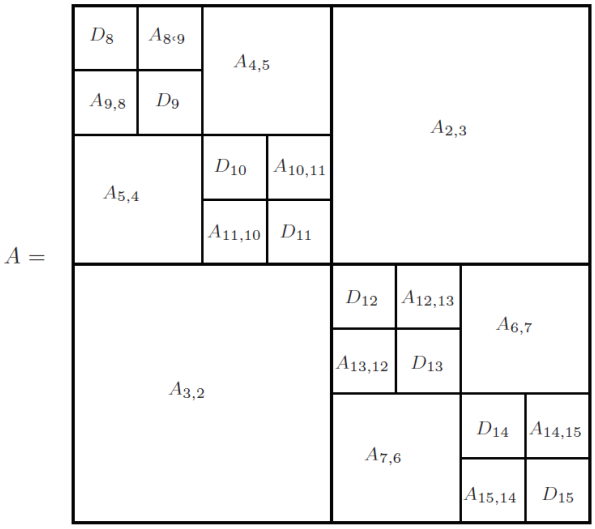


FIGURE 3.3. A matrix $\mathbf{A}$ tessellated according to a simple binary tree with three levels.

3.5. Extended basis matrices

When we introduced the HBS format in Section 2.3, the basis matrices $\mathbf{U}_\tau$ and $\mathbf{V}_\tau$ were defined in a way that is easy to interpret whenever τ is a leaf node. Let us recall that in this case, for any sibling pair $\{\alpha, \beta\}$, the corresponding off-diagonal block should admit a factorization

$$\begin{array}{cccccc} \mathbf{A}(I_\alpha, I_\beta) & = & \mathbf{U}_\alpha & \tilde{\mathbf{A}}_{\alpha,\beta} & \mathbf{V}_\beta^*. \\ n \times n & & n \times k & k \times k & k \times n \end{array} \tag{3.2}$$

So the columns of $\mathbf{U}_\tau$ should span the column space of $\mathbf{A}(I_\alpha, I_\beta)$, and the columns of $\mathbf{V}_\tau$ should span the row space. When τ is not a leaf, the definition (2.8) was recursive, and is perhaps harder to interpret. Our objective in this section is to clarify the meaning of these basis matrices, and to elucidate the connection between the HBS matrix format and the Hierarchically Off-Diagonal Low Rank (HODLR) matrix format introduced in Section ? in the introduction.

Let us start by discussing a simple example involving the tree with three levels shown in Figure 2.1. This tree of course corresponds to the tessellation of the corresponding matrix shown in Figure 3.3. When $\mathbf{A}$ has HBS rank k, it turns out that for every sibling pair $\{\alpha, \beta\}$ (not just the leaves), the corresponding off-diagonal block $\mathbf{A}(I_\alpha, I_\beta)$ as rank at most k. This means that each such block must admit a low-rank factorization

$$\begin{array}{cccccc} \mathbf{A}(I_\alpha, I_\beta) & = & \mathbf{U}_\alpha^{\text{long}} & \tilde{\mathbf{A}}_{\alpha,\beta} & (\mathbf{V}_\beta^{\text{long}})^*. \\ n_\ell \times n_\ell & & n_\ell \times k & k \times k & k \times n_\ell \end{array} \tag{3.3}$$

In (3.3), n_ℓ refers to length of the index vectors for boxes at level ℓ, so that for a perfectly balanced tree,

$$n_\ell = 2^{-\ell} N.$$

We see that at the higher levels, where ℓ is small, the basis matrices will start to become very long. The point of the HBS matrix format is that all the small matrices $\mathbf{U}_\tau$ and $\mathbf{V}_\tau$ efficiently encode the information needed to build the long basis matrices. To illustrate, let us consider the parent box 4 with children 8 and 9. Then $\mathbf{U}_4$ tells us how we can merge the information in the basis matrices $\mathbf{U}_8$ and $\mathbf{U}_9$ to build $\mathbf{U}_4^{\text{long}}$,

$$\begin{array}{cccc} \mathbf{U}_4^{\text{long}} & = & \begin{bmatrix} \mathbf{U}_8 & \mathbf{0} \\ \mathbf{0} & \mathbf{U}_9 \end{bmatrix} & \mathbf{U}_4. \\ 2n \times k & & 2n \times 2k & 2k \times k \end{array} \tag{3.4}$$

Analogously, $\mathbf{U}_2$ contains the information that tells us how to merge the information in $\mathbf{U}_4^{\text{long}}$ and $\mathbf{U}_5^{\text{long}}$ to form $\mathbf{U}_2^{\text{long}}$,

$$\underset{4n \times k}{\mathbf{U}_2^{\text{long}}} = \underset{4n \times 2k}{\begin{bmatrix} \mathbf{U}_4^{\text{long}} & \mathbf{0} \\ \mathbf{0} & \mathbf{U}_5^{\text{long}} \end{bmatrix}} \underset{2k \times k}{\mathbf{U}_2}. \tag{3.5}$$

Combining (3.5) with (3.4), and the analogous expression for $\mathbf{U}_5^{\text{long}}$ we find that

$$\underset{4n \times k}{\mathbf{U}_2^{\text{long}}} = \underset{4n \times 4k}{\begin{bmatrix} \mathbf{U}_8 & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{U}_9 & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{U}_{10} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{U}_{11} \end{bmatrix}} \underset{4k \times 2k}{\begin{bmatrix} \mathbf{U}_4 & \mathbf{0} \\ \mathbf{0} & \mathbf{U}_5 \end{bmatrix}} \underset{2k \times k}{\mathbf{U}_2}. \tag{3.6}$$

The point here is that by storing for each box τ only two small basis matrices $\mathbf{U}_\tau$ and $\mathbf{V}_\tau$ of size $2k \times k$, we can build the basis matrices for any sibling interaction matrix.

To be precise, let us for every node τ in the tree define two "long basis matrices" $\mathbf{U}_\tau^{\text{long}}$ and $\mathbf{V}_\tau^{\text{long}}$ as follows: When τ is a leaf, set

$$\mathbf{U}_\tau^{\text{long}} = \mathbf{U}_\tau, \qquad \text{and} \qquad \mathbf{V}_\tau^{\text{long}} = \mathbf{V}_\tau.$$

For a parent node τ with children α and β, set

$$\mathbf{U}_\tau^{\text{long}} = \begin{bmatrix} \mathbf{U}_\alpha^{\text{long}} & 0 \\ 0 & \mathbf{U}_\beta^{\text{long}} \end{bmatrix} \mathbf{U}_\tau, \qquad \text{and} \qquad \mathbf{V}_\tau^{\text{long}} = \begin{bmatrix} \mathbf{V}_\alpha^{\text{long}} & 0 \\ 0 & \mathbf{V}_\beta^{\text{long}} \end{bmatrix} \mathbf{V}_\tau.$$

One can then prove that for *any* distinct nodes τ and τ' on level ℓ (not just sibling pairs), we have

$$\underset{n\,2^{L-\ell} \times n\,2^{L-\ell}}{\mathbf{A}(I_\tau, I_{\tau'})} = \underset{n\,2^{L-\ell} \times k}{\mathbf{U}_\tau^{\text{long}}} \underset{k \times k}{\tilde{\mathbf{A}}_{\tau,\tau'}} \underset{k \times n\,2^{L-\ell}}{(\mathbf{V}_\tau^{\text{long}})^*}.$$

3.6. The connection between HBS and HODLR matrices

In Section 3.5, we established that every HBS matrix is block separable at every level. This in particular means that it is also compressible in the hierarchically off-diagonal low-rank (HODLR) format that we introduced in the introduction. The "long" basis matrices that appear in the HODLR representation can easily be built via telescoping representations such as (3.6). In this section, we will give an alternative definition of the HBS property that further elucidates precisely which HODLR matrices also admit the more compact HBS representation. In the course of proving the equivalency between the new definition and the old, we will also describe an explicit procedure for numerically building all the short basis matrices.

To start, let us first restate the definition of a HODLR matrix. As usual, we suppose that we are given an $N \times N$ matrix $\mathbf{A}$, with an associated binary tree $\mathcal{T}$ on the index vector $I = 1 : N$. Then $\mathbf{A}$ is compressible in the HODLR format with rank k if the following condition holds:

(HODLR) For every node τ with children α and β, the matrices $\mathbf{A}(I_\alpha, I_\beta)$ and $\mathbf{A}(I_\beta, I_\alpha)$ have rank at most k.

Now, let us consider an different condition:

(HBSnew) For every node α, the matrices $\mathbf{A}(I_\alpha, I_\alpha^{\text{c}})$ and $\mathbf{A}(I_\alpha^{\text{c}}, I_\alpha)$ have rank at most k.

Let us first observe that the condition (HBSnew) is clearly more restrictive than (HODLR), since for any parent node τ with children α and β, it is the case that $I_\beta \subseteq I_\alpha^{\mathrm{c}}$. Next, we will prove that condition (HBSnew) is equivalent to the definition of the HBS property that we gave in Section 2.3.

The fact that the HBS property implies the "HBSnew" property follows immediately from the discussion in Section 3.5: If $\mathbf{A}$ satisfies the HBS property, then use the procedure described to build the "long" basis matrices. this establishes that $\mathbf{A}$ is block separable at every level, which is precisely what HBSnew requires.

Next, let us suppose that $\mathbf{A}$ satisfies the HBSnew property. We will describe a constructive method for building all "short" basis matrices required in the HBS definition in Section 2.3. We start with the smallest boxes. For every leaf node α, form the corresponding long off-diagonal block $\mathbf{A}(I_\alpha, I_\alpha^{\mathrm{c}})$, and build a matrix $\mathbf{U}_\alpha$ whose columns form a basis for the column space of $\mathbf{A}(I_\alpha, I_\alpha^{\mathrm{c}})$, so that

$$\begin{array}{ccccc} \mathbf{A}(I_\alpha, I_\alpha^{\mathrm{c}}) & = & \mathbf{U}_\alpha & \mathbf{U}_\alpha^\dagger \mathbf{A}(I_\alpha, I_\alpha^{\mathrm{c}}), \\ n_\alpha \times (n - n_\alpha) & & n_\alpha \times k & k \times (n - n_\alpha) \end{array} \tag{3.7}$$

where $\mathbf{U}_\alpha^\dagger$ is the pseudo-inverse of $\mathbf{U}_\alpha$. To avoid pathologies, let us require the columns of $\mathbf{U}_\alpha$ to be linearly independent. Let us next consider a node τ whose children $\{\alpha, \beta\}$ are leaf nodes. Then we have a factorization (3.7) for node α, and an analogous one for node β:

$$\begin{array}{ccccc} \mathbf{A}(I_\beta, I_\beta^{\mathrm{c}}) & = & \mathbf{U}_\beta & \mathbf{U}_\beta^\dagger \mathbf{A}(I_\beta, I_\beta^{\mathrm{c}}). \\ n_\beta \times (n - n_\beta) & & n_\beta \times k & k \times (n - n_\beta) \end{array} \tag{3.8}$$

Figure 3.4 illustrates the two blocks being compressed in red and blue. Next, consider compressing the block $\mathbf{A}(I_\tau, I_\tau^{\mathrm{c}})$ associated with the parent τ (shown in green in Figure 3.4). Since $I_\tau^{\mathrm{c}} \subset I_\alpha^{\mathrm{c}}$ and $I_\tau^{\mathrm{c}} \subset I_\beta^{\mathrm{c}}$, we know that

$$\mathbf{A}(I_\alpha, I_\tau^{\mathrm{c}}) = \mathbf{U}_\alpha \mathbf{U}_\alpha^\dagger \mathbf{A}(I_\alpha, I_\tau^{\mathrm{c}}), \qquad \text{and} \qquad \mathbf{A}(I_\beta, I_\tau^{\mathrm{c}}) = \mathbf{U}_\beta \mathbf{U}_\beta^\dagger \mathbf{A}(I_\beta, I_\tau^{\mathrm{c}}).$$

We write this in blocked form as

$$\mathbf{A}(I_\tau, I_\tau^{\mathrm{c}}) = \begin{bmatrix} \mathbf{U}_\alpha & \mathbf{0} \\ \mathbf{0} & \mathbf{U}_\beta \end{bmatrix} \begin{bmatrix} \mathbf{U}_\alpha^\dagger \mathbf{A}(I_\alpha, I_\tau^{\mathrm{c}}) \\ \mathbf{U}_\beta^\dagger \mathbf{A}(I_\beta, I_\tau^{\mathrm{c}}) \end{bmatrix}.$$

Condition (HBSnew) tells us that the matrix $\mathbf{A}(I_\tau, I_\tau^{\mathrm{c}})$ has rank k, and since the factor $\begin{bmatrix} \mathbf{U}_\alpha & \mathbf{0} \\ \mathbf{0} & \mathbf{U}_\beta \end{bmatrix}$ has full rank, we conclude that the matrix $\begin{bmatrix} \mathbf{U}_\alpha^\dagger \mathbf{A}(I_\alpha, I_\tau^{\mathrm{c}}) \\ \mathbf{U}_\beta^\dagger \mathbf{A}(I_\beta, I_\tau^{\mathrm{c}}) \end{bmatrix}$ has rank (at most) k, and can be factored

$$\begin{bmatrix} \mathbf{U}_\alpha^\dagger \mathbf{A}(I_\alpha, I_\tau^{\mathrm{c}}) \\ \mathbf{U}_\beta^\dagger \mathbf{A}(I_\beta, I_\tau^{\mathrm{c}}) \end{bmatrix} = \mathbf{U}_\tau \, \mathbf{U}_\tau^\dagger \begin{bmatrix} \mathbf{U}_\alpha^\dagger \mathbf{A}(I_\alpha, I_\tau^{\mathrm{c}}) \\ \mathbf{U}_\beta^\dagger \mathbf{A}(I_\beta, I_\tau^{\mathrm{c}}) \end{bmatrix}.$$

Putting everything together, we now find that

$$\begin{array}{ccccccc} \mathbf{A}(I_\tau, I_\tau^{\mathrm{c}}) & = & \begin{bmatrix} \mathbf{U}_\alpha & \mathbf{0} \\ \mathbf{0} & \mathbf{U}_\beta \end{bmatrix} & \mathbf{U}_\tau & \mathbf{U}_\tau^\dagger & \begin{bmatrix} \mathbf{U}_\alpha^\dagger \mathbf{A}(I_\alpha, I_\tau^{\mathrm{c}}) \\ \mathbf{U}_\beta^\dagger \mathbf{A}(I_\beta, I_\tau^{\mathrm{c}}) \end{bmatrix}. \\ n_\tau \times (n - n_\tau) & & n_\tau \times 2k & 2k \times k & & k \times (n - n_\tau) \end{array}$$

The procedure for constructing the basis matrices $\{\mathbf{V}_\tau\}$ is entirely analogous.

Let us stress that the pseudo-inverses that appear in the formulas in this section are in practice never computed, they simply form a convenient way to avoid introducing extra notation. To illustrate, suppose that we are given an $m \times n$ matrix $\mathbf{X}$ of rank k, and we compute an $m \times k$ matrix $\mathbf{U}$ whose columns form a basis for the column space of $\mathbf{X}$. This results in a factorization that we can conveniently write $\mathbf{X} = \mathbf{U}\big(\mathbf{U}^\dagger \mathbf{X}\big)$. In practice, though, the factor $\mathbf{U}^\dagger \mathbf{X}$ will be formed automatically either as the "$\mathbf{R}$" factor in a QR factorization, or as the factor "$\mathbf{\Sigma V}^*$" when we use an SVD.

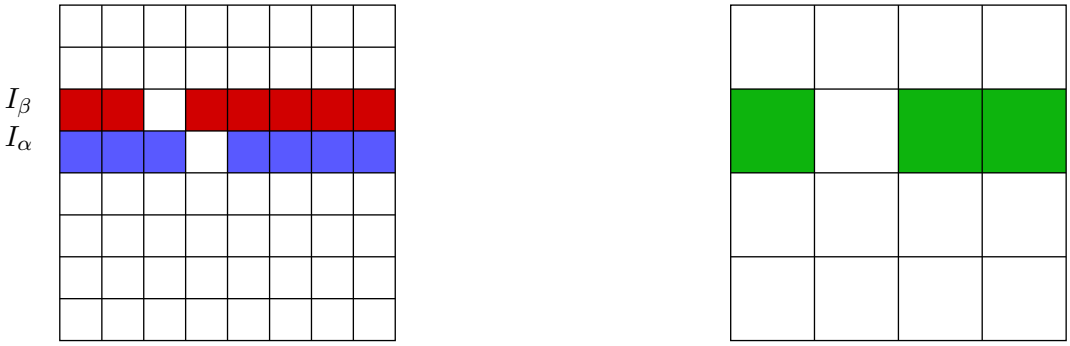


FIGURE 3.4. Compression of HBS matrices. Here, α and β are leaf nodes with a parent τ so that $I_\tau = I_\alpha \cup I_\beta$. The left figure shows $\mathbf{A}(I_\alpha, I_\alpha^c)$ in red and $\mathbf{A}(I_\beta, I_\beta^c)$ in blue. The right figure shows $\mathbf{A}(I_\tau, I_\tau^c)$ in green.

CHAPTER 4

Interpolative decompositions and skeletonization

The rank-structured formats described in Chapters 1 – 3 become particularly effective when the off-diagonal blocks in the matrix are factored using the interpolative decomposition (ID). When the ID is used, the off-diagonal blocks in the compressed matrices are represented simply as *sub-matrices* of the original matrix. This obviates the need to explicitly form them since they are uniquely determined by specifying only the index vectors that identify the sub-blocks. Perhaps even more importantly, when the ID is used, all long-range interactions are still expressed through the original kernel function. In other words, we in a sense preserve "the physics" of the underlying equation.

The discussion in this chapter is framed around the same model problem as in Chapters Chapters 1 – 3: We seek to solve a linear system

$$\mathbf{A}\mathbf{q} = \mathbf{f} \tag{4.1}$$

where $\mathbf{A}$ is an $N \times N$ dense coefficient matrix approximating a boundary integral operators such as in (1.1). As always, we let $I = 1 : N$ denote the global index vector.

4.1. The single level compression scheme revisited

Let us revisit the discussion in Section 1.2 of how to invert a block separable matrix, with now a slightly different approach to reducing the size of the system. Our starting point is still a partition $I = I_1 \cup I_2 \cup \cdots \cup I_p$ of the index vector, which induces a blocking (1.3) of the coefficient matrix. To keep the notation uncluttered, we suppose that every I_τ holds precisely n indices, so that $\mathbf{A}$ consists of $p \times p$ blocks, each of size $n \times n$. As in Section 1.2 we assume that each off-diagonal block admits a factorization

$$\underset{n\times n}{\mathbf{A}_{\sigma,\tau}} = \underset{n\times k}{\mathbf{U}_\sigma}\ \underset{k\times k}{\tilde{\mathbf{A}}_{\sigma,\tau}}\ \underset{k\times n}{\mathbf{V}_\tau^*}, \qquad \sigma,\tau \in \{1,\,2,\,\ldots,\,p\}, \quad \sigma \neq \tau. \tag{4.2}$$

Where we deviate from our previous treatment is that we next extend each basis matrix $\mathbf{U}_\tau$ by adding a matrix $\mathbf{U}_\tau^{\mathrm{r}}$ with $n - k$ columns to form a *square* change of basis matrix $\mathbb{U}_\tau$, and of course do the same for the $\mathbf{V}_\tau$ matrices. In other words, we form matrices

$$\underset{n\times n}{\mathbb{U}_\tau} = \left[\ \underset{n\times k}{\mathbf{U}_\tau}\ \ \underset{n\times(n-k)}{\mathbf{U}_\tau^{\mathrm{r}}}\ \right] \quad \text{and} \quad \underset{n\times n}{\mathbb{V}_\tau} = \left[\ \underset{n\times k}{\mathbf{V}_\tau}\ \ \underset{n\times(n-k)}{\mathbf{V}_\tau^{\mathrm{r}}}\ \right]$$

When $\mathbf{U}_\tau$ and $\mathbf{V}_\tau$ have orthonormal columns, we add columns to make $\mathbb{U}_\tau$ and $\mathbb{U}_\tau$ unitary. We more generally allow the basis matrices to be non-orthogonal, however, in which case we insist only that $\mathbb{U}_\tau$ and $\mathbb{V}_\tau$ should be well-conditioned. With the square change of basis matrices, the relation (4.2) takes the form

$$\underset{n\times n}{\mathbf{A}_{\sigma,\tau}} = \underset{n\times n}{\mathbb{U}_\sigma}\ \underset{n\times n}{\begin{bmatrix} \tilde{\mathbf{A}}_{\sigma,\tau} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} \end{bmatrix}}\ \underset{n\times n}{\mathbb{V}_\tau^*}, \qquad \sigma,\tau \in \{1,\,2,\,\ldots,\,p\}, \quad \sigma \neq \tau. \tag{4.3}$$

The diagonal blocks are full rank, so the factorization analogous to (4.3) for $\sigma = \tau$ becomes

$$\underset{n\times n}{\mathbf{A}_{\tau,\tau}} = \underset{n\times n}{\mathbb{U}_\tau}\ \underset{n\times n}{\begin{bmatrix} \tilde{\mathbf{A}}_{\tau,\tau}^{\mathrm{ss}} & \tilde{\mathbf{A}}_{\tau,\tau}^{\mathrm{sr}} \\ \tilde{\mathbf{A}}_{\tau,\tau}^{\mathrm{rs}} & \tilde{\mathbf{A}}_{\tau,\tau}^{\mathrm{rr}} \end{bmatrix}}\ \underset{n\times n}{\mathbb{V}_\tau^*}, \tag{4.4}$$

where the blocks are *defined* by the relationship

$$\begin{bmatrix} \mathbf{A}^{\mathrm{ss}}_{\tau,\tau} & \mathbf{A}^{\mathrm{sr}}_{\tau,\tau} \\ \mathbf{A}^{\mathrm{rs}}_{\tau,\tau} & \mathbf{A}^{\mathrm{rr}}_{\tau,\tau} \end{bmatrix} := \mathbb{U}_\tau^{-1} \mathbf{A}_{\tau,\tau} \mathbb{V}_\tau^{-*}. \tag{4.5}$$

The letters "s" and "r" stand for *skeleton* and *residual* variables, respectively. The skeleton variables involve global interactions, while the residual variables represent local interactions.

Using the new square change of basis matrices, we can write the coefficient matrix $\mathbf{A}$ in blocked form as

$$\mathbf{A} = \mathbb{U} \left[\begin{array}{cc|cc|cc} \tilde{\mathbf{A}}^{\mathrm{ss}}_{1,1} & \tilde{\mathbf{A}}^{\mathrm{sr}}_{1,1} & \tilde{\mathbf{A}}_{1,2} & \mathbf{0} & \tilde{\mathbf{A}}_{1,3} & \mathbf{0} \\ \tilde{\mathbf{A}}^{\mathrm{rs}}_{1,1} & \tilde{\mathbf{A}}^{\mathrm{rr}}_{1,1} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \hline \tilde{\mathbf{A}}_{2,1} & \mathbf{0} & \tilde{\mathbf{A}}^{\mathrm{ss}}_{2,2} & \tilde{\mathbf{A}}^{\mathrm{sr}}_{2,2} & \tilde{\mathbf{A}}_{2,3} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \tilde{\mathbf{A}}^{\mathrm{rs}}_{2,2} & \tilde{\mathbf{A}}^{\mathrm{rr}}_{2,2} & \mathbf{0} & \mathbf{0} \\ \hline \tilde{\mathbf{A}}_{3,1} & \mathbf{0} & \tilde{\mathbf{A}}_{3,2} & \mathbf{0} & \tilde{\mathbf{A}}^{\mathrm{ss}}_{3,3} & \tilde{\mathbf{A}}^{\mathrm{sr}}_{3,3} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \tilde{\mathbf{A}}^{\mathrm{rs}}_{3,3} & \tilde{\mathbf{A}}^{\mathrm{rr}}_{3,3} \end{array}\right] \mathbb{V}^*, \tag{4.6}$$

where we set $p = 3$ to keep things simple, and where $\mathbb{U}$, and $\mathbb{V}$ are defined by

$$\mathbb{U} = \begin{bmatrix} \mathbb{U}_1 & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbb{U}_2 & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbb{U}_3 \end{bmatrix}, \qquad \text{and} \qquad \mathbb{V} = \begin{bmatrix} \mathbb{V}_1 & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbb{V}_2 & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbb{V}_3 \end{bmatrix}.$$

In the new basis, the linear system (4.1) takes the form

$$\tilde{\mathbf{A}} \mathbb{V}^* \mathbf{q} = \mathbb{U}^{-1} \mathbf{f},$$

which we write out as

$$\left[\begin{array}{cc|cc|cc} \tilde{\mathbf{A}}^{\mathrm{ss}}_{1,1} & \tilde{\mathbf{A}}^{\mathrm{sr}}_{1,1} & \tilde{\mathbf{A}}_{1,2} & \mathbf{0} & \tilde{\mathbf{A}}_{1,3} & \mathbf{0} \\ \tilde{\mathbf{A}}^{\mathrm{rs}}_{1,1} & \tilde{\mathbf{A}}^{\mathrm{rr}}_{1,1} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \hline \tilde{\mathbf{A}}_{2,1} & \mathbf{0} & \tilde{\mathbf{A}}^{\mathrm{ss}}_{2,2} & \tilde{\mathbf{A}}^{\mathrm{sr}}_{2,2} & \tilde{\mathbf{A}}_{2,3} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \tilde{\mathbf{A}}^{\mathrm{rs}}_{2,2} & \tilde{\mathbf{A}}^{\mathrm{rr}}_{2,2} & \mathbf{0} & \mathbf{0} \\ \hline \tilde{\mathbf{A}}_{3,1} & \mathbf{0} & \tilde{\mathbf{A}}_{3,2} & \mathbf{0} & \tilde{\mathbf{A}}^{\mathrm{ss}}_{3,3} & \tilde{\mathbf{A}}^{\mathrm{sr}}_{3,3} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \tilde{\mathbf{A}}^{\mathrm{rs}}_{3,3} & \tilde{\mathbf{A}}^{\mathrm{rr}}_{3,3} \end{array}\right] \left[\begin{array}{c} \tilde{\mathbf{q}}^{\mathrm{s}}_1 \\ \tilde{\mathbf{q}}^{\mathrm{r}}_1 \\ \hline \tilde{\mathbf{q}}^{\mathrm{s}}_2 \\ \tilde{\mathbf{q}}^{\mathrm{r}}_2 \\ \hline \tilde{\mathbf{q}}^{\mathrm{s}}_3 \\ \tilde{\mathbf{q}}^{\mathrm{r}}_3 \end{array}\right] = \left[\begin{array}{c} \tilde{\mathbf{f}}^{\mathrm{s}}_1 \\ \tilde{\mathbf{f}}^{\mathrm{r}}_1 \\ \hline \tilde{\mathbf{f}}^{\mathrm{s}}_2 \\ \tilde{\mathbf{f}}^{\mathrm{r}}_2 \\ \hline \tilde{\mathbf{f}}^{\mathrm{s}}_3 \\ \tilde{\mathbf{f}}^{\mathrm{r}}_3, \end{array}\right] \tag{4.7}$$

where the new variables are defined by

$$\tilde{\mathbf{q}}_\tau = \begin{bmatrix} \tilde{\mathbf{q}}^{\mathrm{s}}_\tau \\ \tilde{\mathbf{q}}^{\mathrm{r}}_\tau \end{bmatrix} := \mathbb{V}_\tau^* \mathbf{q}_\tau \qquad \text{and} \qquad \tilde{\mathbf{f}}_\tau = \begin{bmatrix} \tilde{\mathbf{f}}^{\mathrm{s}}_\tau \\ \tilde{\mathbf{f}}^{\mathrm{r}}_\tau \end{bmatrix} := \mathbb{U}_\tau^{-1} \mathbf{f}_\tau. \tag{4.8}$$

Provided that each matrix $\tilde{\mathbf{A}}^{\mathrm{rr}}_{\tau,\tau}$ is invertible, we can eliminate the "residual" variables from (4.7), to obtain the reduced system

$$\underbrace{\begin{bmatrix} \tilde{\mathbf{D}}_1 & \tilde{\mathbf{A}}_{1,2} & \tilde{\mathbf{A}}_{1,3} \\ \tilde{\mathbf{A}}_{2,1} & \tilde{\mathbf{D}}_2 & \tilde{\mathbf{A}}_{2,3} \\ \tilde{\mathbf{A}}_{3,1} & \tilde{\mathbf{A}}_{3,2} & \tilde{\mathbf{D}}_3 \end{bmatrix}}_{=:\tilde{\mathbf{A}}} \begin{bmatrix} \tilde{\mathbf{q}}^{\mathrm{s}}_1 \\ \tilde{\mathbf{q}}^{\mathrm{s}}_2 \\ \tilde{\mathbf{q}}^{\mathrm{s}}_3 \end{bmatrix} = \begin{bmatrix} \tilde{\mathbf{f}}^{\mathrm{s}}_1 - \tilde{\mathbf{A}}^{\mathrm{sr}}_{1,1} \big(\tilde{\mathbf{A}}^{\mathrm{rr}}_{1,1}\big)^{-1} \tilde{\mathbf{f}}^{\mathrm{r}}_1 \\ \tilde{\mathbf{f}}^{\mathrm{s}}_2 - \tilde{\mathbf{A}}^{\mathrm{sr}}_{2,2} \big(\tilde{\mathbf{A}}^{\mathrm{rr}}_{2,2}\big)^{-1} \tilde{\mathbf{f}}^{\mathrm{r}}_2 \\ \tilde{\mathbf{f}}^{\mathrm{s}}_3 - \tilde{\mathbf{A}}^{\mathrm{sr}}_{3,3} \big(\tilde{\mathbf{A}}^{\mathrm{rr}}_{3,3}\big)^{-1} \tilde{\mathbf{f}}^{\mathrm{r}}_3 \end{bmatrix}, \tag{4.9}$$

where the diagonal blocks are defined by

$$\tilde{\mathbf{D}}_\tau = \tilde{\mathbf{A}}^{\mathrm{ss}}_{\tau,\tau} - \tilde{\mathbf{A}}^{\mathrm{sr}}_{\tau,\tau} \big(\tilde{\mathbf{A}}^{\mathrm{rr}}_{\tau,\tau}\big)^{-1} \tilde{\mathbf{A}}^{\mathrm{rs}}_{\tau,\tau}. \tag{4.10}$$

Once (4.9) has been solved, we can recover the residual variables from the formula

$$\tilde{\mathbf{q}}^{\mathrm{r}}_\tau = \big(\tilde{\mathbf{A}}^{\mathrm{rr}}_{\tau,\tau}\big)^{-1} \tilde{\mathbf{f}}^{\mathrm{r}}_\tau - \big(\tilde{\mathbf{A}}^{\mathrm{rr}}_{\tau,\tau}\big)^{-1} \tilde{\mathbf{A}}^{\mathrm{rs}}_{\tau,\tau} \tilde{\mathbf{q}}^{\mathrm{s}}_\tau. \tag{4.11}$$

Let us summarize our findings in a lemma that provides a formula for the inverse of a block separable matrix:

LEMMA 4.1. *Let $\mathbf{A}$ denote a matrix that consists of $p \times p$ blocks, each of size $n \times n$, and such that every off-diagonal block admits a low-rank factorization (4.3) for some rank $k < n$. We assume that each matrix $\mathbb{U}_\tau$ and $\mathbb{V}_\tau$ is non-singular, and define the blocks $\tilde{\mathbf{A}}^{\mathrm{ss}}_{\tau,\tau}$, $\tilde{\mathbf{A}}^{\mathrm{sr}}_{\tau,\tau}$, $\tilde{\mathbf{A}}^{\mathrm{rs}}_{\tau,\tau}$, and $\tilde{\mathbf{A}}^{\mathrm{rr}}_{\tau,\tau}$ via the formulas (4.5). Suppose that the matrix $\tilde{\mathbf{A}}$ defined in (4.9) in invertible, and also that every matrix $\tilde{\mathbf{A}}^{\mathrm{rr}}_{\tau,\tau}$ is invertible. Then $\mathbf{A}$ is invertible, and its inverse is given by the formula*

$$\underset{pn \times pn}{\mathbf{A}^{-1}} = \underset{pn \times pk}{\mathbf{E}} \underset{pk \times pk}{\begin{bmatrix} \tilde{\mathbf{D}}_1 & \tilde{\mathbf{A}}_{1,2} & \dots & \tilde{\mathbf{A}}_{1,p} \\ \tilde{\mathbf{A}}_{2,1} & \tilde{\mathbf{D}}_2 & \dots & \tilde{\mathbf{A}}_{2,p} \\ \vdots & \vdots & & \vdots \\ \tilde{\mathbf{A}}_{p,1} & \tilde{\mathbf{A}}_{p,2} & \dots & \tilde{\mathbf{D}}_p \end{bmatrix}^{-1}} \underset{pk \times pn}{\mathbf{F}^*} + \underset{pn \times pn}{\mathbf{G}} \tag{4.12}$$

where the diagonal blocks $\tilde{\mathbf{D}}_\tau$ are defined by (4.10), and where $\mathbf{E}, \mathbf{F}, \mathbf{G}$ are the block-diagonal matrices whose diagonal blocks are given by

$$\begin{aligned} \mathbf{E}_\tau &= \mathbb{V}_\tau^{-*} \begin{bmatrix} \mathbf{I} \\ -(\tilde{\mathbf{A}}^{\mathrm{rr}}_{\tau,\tau})^{-1} \tilde{\mathbf{A}}^{\mathrm{rs}}_{\tau,\tau} \end{bmatrix} \\ \mathbf{F}_\tau^* &= \begin{bmatrix} \mathbf{I} & -\tilde{\mathbf{A}}^{\mathrm{sr}}_{\tau,\tau}(\tilde{\mathbf{A}}^{\mathrm{rr}}_{\tau,\tau})^{-1} \end{bmatrix} \mathbb{U}_\tau^{-1} \\ \mathbf{G}_\tau &= \mathbb{V}_\tau^{-*} \begin{bmatrix} \mathbf{0} & \mathbf{0} \\ \mathbf{0} & (\tilde{\mathbf{A}}^{\mathrm{rr}}_{\tau,\tau})^{-1} \end{bmatrix} \mathbb{U}_\tau^{-1} \end{aligned}$$

Lemma 4.1 is of course just a slight variation of Lemma 1.1. It has the advantage that it makes explicit the fact that each matrix $\mathbf{G}_\tau$ is of rank $n-k$. Another interesting difference to Lemma 1.1 is that the matrices $\tilde{\mathbf{A}}^{\mathrm{rr}}_{\tau,\tau}$ that need to be inverted are of size $(n-k) \times (n-k)$, which is in contrast to the $n \times n$ matrices $\mathbf{D}_\tau$ in Lemma 1.1. In general, this apparent advantage is offset by the added cost of building the square change of basis matrices $\mathbb{U}_\tau$ and $\mathbb{V}_\tau$, and the need to apply these to the diagonal blocks. However, for the particular case where the interpolatory decomposition is used, all of these matrices, as well as their inverses, take on a particularly simple form that makes them economical to use.

PROOF. The core of the proof of the relation (4.12) consists of the reduction of the linear system (4.7) to (4.9), which we already described. All that remains is to work out the exact form of the matrices $\mathbf{E}_\tau$, $\mathbf{F}_\tau$, and $\mathbf{G}_\tau$. Let us start with the matrix $\mathbf{F}_\tau$ which maps the vector $\mathbf{f}(I_\tau)$ to the right side $\tilde{\mathbf{f}}^{\mathrm{s}}_1 - \tilde{\mathbf{A}}^{\mathrm{sr}}_{1,1}(\tilde{\mathbf{A}}^{\mathrm{rr}}_{1,1})^{-1}\tilde{\mathbf{f}}^{\mathrm{r}}_1$ in (4.9). We find that

$$\tilde{\mathbf{f}}^{\mathrm{s}}_1 - \tilde{\mathbf{A}}^{\mathrm{sr}}_{1,1}(\tilde{\mathbf{A}}^{\mathrm{rr}}_{1,1})^{-1}\tilde{\mathbf{f}}^{\mathrm{r}}_1 = \begin{bmatrix} \mathbf{I} & -\tilde{\mathbf{A}}^{\mathrm{sr}}_{1,1}(\tilde{\mathbf{A}}^{\mathrm{rr}}_{1,1})^{-1} \end{bmatrix} \begin{bmatrix} \tilde{\mathbf{f}}^{\mathrm{s}}_\tau \\ \tilde{\mathbf{f}}^{\mathrm{r}}_\tau \end{bmatrix} = \underbrace{\begin{bmatrix} \mathbf{I} & -\tilde{\mathbf{A}}^{\mathrm{sr}}_{1,1}(\tilde{\mathbf{A}}^{\mathrm{rr}}_{1,1})^{-1} \end{bmatrix} \mathbb{U}_\tau^{-1}}_{=:\mathbf{F}_\tau^*} \mathbf{f}(I_\tau).$$

The matrices $\mathbf{E}_\tau$ and $\mathbf{G}_\tau$ are the matrices that we need to recover the original unknowns $\mathbf{q}_\tau$ from the solution $\tilde{\mathbf{q}}^{\mathrm{s}}_\tau$ of the reduced system. Using the relation (4.11), we find that

$$\begin{aligned} \mathbf{q}_\tau = \mathbb{V}_\tau^{-*} \begin{bmatrix} \tilde{\mathbf{q}}^{\mathrm{s}}_\tau \\ \tilde{\mathbf{q}}^{\mathrm{r}}_\tau \end{bmatrix} &= \mathbb{V}_\tau^{-*} \begin{bmatrix} \tilde{\mathbf{q}}^{\mathrm{s}}_\tau \\ (\tilde{\mathbf{A}}^{\mathrm{rr}}_{\tau,\tau})^{-1}\tilde{\mathbf{f}}^{\mathrm{r}}_\tau - (\tilde{\mathbf{A}}^{\mathrm{rr}}_{\tau,\tau})^{-1}\tilde{\mathbf{A}}^{\mathrm{rs}}_{\tau,\tau}\tilde{\mathbf{q}}^{\mathrm{s}}_\tau \end{bmatrix} \\ &= \mathbb{V}_\tau^{-*} \begin{bmatrix} \mathbf{I} \\ -(\tilde{\mathbf{A}}^{\mathrm{rr}}_{\tau,\tau})^{-1}\tilde{\mathbf{A}}^{\mathrm{rs}}_{\tau,\tau} \end{bmatrix} \tilde{\mathbf{q}}^{\mathrm{s}}_\tau + \mathbb{V}_\tau^{-*} \begin{bmatrix} \mathbf{0} & \mathbf{0} \\ \mathbf{0} & (\tilde{\mathbf{A}}^{\mathrm{rr}}_{\tau,\tau})^{-1} \end{bmatrix} \begin{bmatrix} \tilde{\mathbf{f}}^{\mathrm{s}}_\tau \\ \tilde{\mathbf{f}}^{\mathrm{r}}_\tau \end{bmatrix} \\ &= \underbrace{\mathbb{V}_\tau^{-*} \begin{bmatrix} \mathbf{I} \\ -(\tilde{\mathbf{A}}^{\mathrm{rr}}_{\tau,\tau})^{-1}\tilde{\mathbf{A}}^{\mathrm{rs}}_{\tau,\tau} \end{bmatrix}}_{=:\mathbf{E}_\tau} \tilde{\mathbf{q}}^{\mathrm{s}}_\tau + \underbrace{\mathbb{V}_\tau^{-*} \begin{bmatrix} \mathbf{0} & \mathbf{0} \\ \mathbf{0} & (\tilde{\mathbf{A}}^{\mathrm{rr}}_{\tau,\tau})^{-1} \end{bmatrix} \mathbb{U}_\tau^{-1}}_{=:\mathbf{G}_\tau} \mathbf{f}(I_\tau). \end{aligned}$$

□

4.2. LU factorization of a block separable matrix

The formulation in Section 4.1 that is based on square change of basis matrices is particularly convenient for computing an LU decomposition of a block separable matrix, since it introduced large zero blocks that we can exploit, cf. (4.6). To demonstrate how this works, let us use the same framework as in Section 4.1, so that $\mathbf{A}$ is a blocked matrix, with the off-diagonal blocks satisfying (4.3). We split the diagonal blocks into skeleton and residual components as in (4.5). However, we will in this section reorder the compressed matrix $\tilde{\mathbf{A}}$ so that the residual variables all come first. To be precise, let us now define the global change of basis matrices $\mathbb{U}$ and $\mathbb{V}$ via

$$\mathbb{U} = \left[\begin{array}{ccc|ccc} \mathbf{U}_1^{\mathrm{r}} & \mathbf{0} & \mathbf{0} & \mathbf{U}_1 & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{U}_2^{\mathrm{r}} & \mathbf{0} & \mathbf{0} & \mathbf{U}_2 & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{U}_3^{\mathrm{r}} & \mathbf{0} & \mathbf{0} & \mathbf{U}_3 \end{array}\right] \quad \text{and} \quad \mathbb{V} = \left[\begin{array}{ccc|ccc} \mathbf{V}_1^{\mathrm{r}} & \mathbf{0} & \mathbf{0} & \mathbf{V}_1 & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{V}_2^{\mathrm{r}} & \mathbf{0} & \mathbf{0} & \mathbf{V}_2 & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{V}_3^{\mathrm{r}} & \mathbf{0} & \mathbf{0} & \mathbf{V}_3 \end{array}\right].$$

Then $\mathbf{A}$ admits the factorization

$$\mathbf{A} = \mathbb{U} \left[\begin{array}{ccc|ccc} \tilde{\mathbf{A}}_{1,1}^{\mathrm{rr}} & \mathbf{0} & \mathbf{0} & \tilde{\mathbf{A}}_{1,1}^{\mathrm{rs}} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \tilde{\mathbf{A}}_{2,2}^{\mathrm{rr}} & \mathbf{0} & \mathbf{0} & \tilde{\mathbf{A}}_{2,2}^{\mathrm{rs}} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \tilde{\mathbf{A}}_{3,3}^{\mathrm{rr}} & \mathbf{0} & \mathbf{0} & \tilde{\mathbf{A}}_{3,3}^{\mathrm{rs}} \\ \hline \tilde{\mathbf{A}}_{1,1}^{\mathrm{sr}} & \mathbf{0} & \mathbf{0} & \tilde{\mathbf{A}}_{1,1}^{\mathrm{ss}} & \tilde{\mathbf{A}}_{1,2} & \tilde{\mathbf{A}}_{1,3} \\ \mathbf{0} & \tilde{\mathbf{A}}_{2,2}^{\mathrm{sr}} & \mathbf{0} & \tilde{\mathbf{A}}_{2,1} & \tilde{\mathbf{A}}_{2,2}^{\mathrm{ss}} & \tilde{\mathbf{A}}_{2,3} \\ \mathbf{0} & \mathbf{0} & \tilde{\mathbf{A}}_{3,3}^{\mathrm{sr}} & \tilde{\mathbf{A}}_{3,1} & \tilde{\mathbf{A}}_{3,2} & \tilde{\mathbf{A}}_{3,3}^{\mathrm{ss}} \end{array}\right] \mathbb{V}^*.$$

We show the formulas for the particular case where $p = 3$, but the generalization should be obvious. Next, we simply compute the LU factorization of each diagonal block $\mathbf{A}_{\tau,\tau}^{\mathrm{rr}}$

$$\tilde{\mathbf{A}}_{\tau,\tau}^{\mathrm{rr}} = \mathbf{L}_\tau \mathbf{W}_\tau. \tag{4.13}$$

(In (4.13), we could not use the customary letter U for the upper triangular factor since this letter is already taken for the basis matrix for the column space of the block.) Provided that each $\tilde{\mathbf{A}}_{\tau,\tau}^{\mathrm{rr}}$ is non-singular, the partial LU factorization of $\mathbf{A}$ takes the form

$$\mathbf{A} = \mathbb{U}\mathbf{L} \left[\begin{array}{ccc|ccc} \mathbf{I} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{I} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{I} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \hline \mathbf{0} & \mathbf{0} & \mathbf{0} & \tilde{\mathbf{D}}_1 & \tilde{\mathbf{A}}_{1,2} & \tilde{\mathbf{A}}_{1,3} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \tilde{\mathbf{A}}_{2,1} & \tilde{\mathbf{D}}_2 & \tilde{\mathbf{A}}_{2,3} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \tilde{\mathbf{A}}_{3,1} & \tilde{\mathbf{A}}_{3,2} & \tilde{\mathbf{D}}_3 \end{array}\right] \mathbf{W}\mathbb{V}^*$$

where the matrices $\mathbf{L}$ and $\mathbf{W}$ are defined by

$$\mathbf{L} := \left[\begin{array}{ccc|ccc} \mathbf{L}_1 & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{L}_2 & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{L}_3 & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \hline \tilde{\mathbf{A}}_{1,1}^{\mathrm{sr}}\mathbf{W}_1^{-1} & \mathbf{0} & \mathbf{0} & \mathbf{I} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \tilde{\mathbf{A}}_{2,2}^{\mathrm{sr}}\mathbf{W}_2^{-1} & \mathbf{0} & \mathbf{0} & \mathbf{I} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \tilde{\mathbf{A}}_{3,3}^{\mathrm{sr}}\mathbf{W}_3^{-1} & \mathbf{0} & \mathbf{0} & \mathbf{I} \end{array}\right]$$

$$\mathbf{W} := \left[\begin{array}{ccc|ccc} \mathbf{W}_1 & \mathbf{0} & \mathbf{0} & \mathbf{L}_1^{-1}\tilde{\mathbf{A}}_{1,1}^{\mathrm{rs}} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{W}_2 & \mathbf{0} & \mathbf{0} & \mathbf{L}_2^{-1}\tilde{\mathbf{A}}_{2,2}^{\mathrm{rs}} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{W}_3 & \mathbf{0} & \mathbf{0} & \mathbf{L}_3^{-1}\tilde{\mathbf{A}}_{3,3}^{\mathrm{rs}} \\ \hline \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{I} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{I} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{I} \end{array}\right].$$

Local pivoting within a block can easily be introduced by simply inserting permutation matrices on the left, right, or both sides in the factorization (4.13), and then modifying the subsequent formulas as necessary. As always, pivoting *across* blocks would be far more challenging.

4.3. The ID and block separable matrices

Let us next explore the advantages of using the interpolatory decomposition for compressing the off-diagonal blocks in a block separable matrix. To start off the discussion let us revisit the situation where we compress a single block: We consider a simple partitioning of the index vector into two parts

$$I = I_1 \cup I_2,$$

which lets us write (4.1) as a 2×2 block matrix

$$\begin{bmatrix} \mathbf{A}_{11} & \mathbf{A}_{12} \\ \mathbf{A}_{21} & \mathbf{A}_{22} \end{bmatrix} \begin{bmatrix} \mathbf{q}_1 \\ \mathbf{q}_2 \end{bmatrix} = \begin{bmatrix} \mathbf{f}_1 \\ \mathbf{f}_2 \end{bmatrix}. \tag{4.14}$$

Let n_1 and n_2 denote the number of points in I_1 and I_2 respectively, so that $N = n_1 + n_2$. When $\mathbf{A}$ is a discretized integral operator on a domain Γ, we may think of I_1 as marking a set of nodes in a subregion Γ_1, as shown in Figure 4.1. We now use the ID to factorize $\mathbf{A}_{1,2}$ and $\mathbf{A}_{2,1}$ so that

$$\mathbf{A}_{1,2} = \mathbf{P}_1 \begin{bmatrix} \mathbf{I} \\ \mathbf{S}^* \end{bmatrix} \mathbf{A}(I_1^{\mathrm{s}}, I_2), \qquad \text{and} \qquad \mathbf{A}_{2,1} = \mathbf{A}(I_2, I_1^{\mathrm{s}}) \begin{bmatrix} \mathbf{I} & \mathbf{T} \end{bmatrix} \mathbf{P}_1^*, \tag{4.15}$$

where I_1^{s} denotes the index vector that lists the k "skeleton" indices chosen in the ID. (We for simplicity restrict ourselves to the case where the same skeleton indices are used for both the columns and the rows.) We refer to the indices in I_1 that were *not* picked as skeleton points as the "residual" indices; we gather these in the vector I_1^{r} so that

$$I_1 = I_1^{\mathrm{s}} \cup I_1^{\mathrm{r}}.$$

Introducing the basis matrices

$$\mathbf{U}_1 = \mathbf{P}_1 \begin{bmatrix} \mathbf{I} \\ \mathbf{S}^* \end{bmatrix}, \qquad \text{and} \qquad \mathbf{V}_1 = \mathbf{P}_1 \begin{bmatrix} \mathbf{I} \\ \mathbf{T}^* \end{bmatrix},$$

we can write (4.15) compactly as

$$\mathbf{A}_{1,2} = \mathbf{U}_1 \mathbf{A}(I_1^{\mathrm{s}}, I_2), \qquad \text{and} \qquad \mathbf{A}_{2,1} = \mathbf{A}(I_2, I_1^{\mathrm{s}}) \mathbf{V}_1^*. \tag{4.16}$$

The key observation is now that it becomes particularly easy to extend the basis matrices $\mathbf{U}_1$ and $\mathbf{V}_1$ to square well-conditioned "change of basis matrices"

$$\mathbb{U}_1 = \mathbf{P}_1 \begin{bmatrix} \mathbf{I} & \mathbf{0} \\ \mathbf{S}^* & \mathbf{I} \end{bmatrix} \qquad \text{and} \qquad \mathbb{V}_1 = \mathbf{P}_1 \begin{bmatrix} \mathbf{I} & \mathbf{0} \\ \mathbf{T}^* & \mathbf{I} \end{bmatrix}.$$

Moreover, the inverses of these matrices are given by

$$\mathbb{U}_1^{-1} = \begin{bmatrix} \mathbf{I} & \mathbf{0} \\ -\mathbf{S}^* & \mathbf{I} \end{bmatrix} \mathbf{P}_1^* \qquad \text{and} \qquad \mathbb{V}_1^{-1} = \begin{bmatrix} \mathbf{I} & \mathbf{0} \\ -\mathbf{T}^* & \mathbf{I} \end{bmatrix} \mathbf{P}_1^*.$$

The simple forms of the matrices $\mathbb{U}_1$ and $\mathbb{V}_1$ and their inverses lead to several simplification in the formulas in Sections 4.1 and 4.2. To start, let us revisit the formula for the four diagonal blocks in the transformed system, cf. (4.5),

$$\begin{aligned} \begin{bmatrix} \mathbf{A}_{1,1}^{\mathrm{ss}} & \mathbf{A}_{1,1}^{\mathrm{sr}} \\ \mathbf{A}_{1,1}^{\mathrm{rs}} & \mathbf{A}_{1,1}^{\mathrm{rr}} \end{bmatrix} := \mathbb{U}_1^{-1} \mathbf{A}_{1,1} \mathbb{V}_1^{-*} &= \begin{bmatrix} \mathbf{I} & \mathbf{0} \\ -\mathbf{S}^* & \mathbf{I} \end{bmatrix} \mathbf{P}_1^* \mathbf{A}(I_1, I_1) \mathbf{P}_1 \begin{bmatrix} \mathbf{I} & -\mathbf{T} \\ \mathbf{0} & \mathbf{I} \end{bmatrix} \\ &= \begin{bmatrix} \mathbf{I} & \mathbf{0} \\ -\mathbf{S}^* & \mathbf{I} \end{bmatrix} \begin{bmatrix} \mathbf{A}(I_1^{\mathrm{s}}, I_1^{\mathrm{s}}) & \mathbf{A}(I_1^{\mathrm{s}}, I_1^{\mathrm{r}}) \\ \mathbf{A}(I_1^{\mathrm{r}}, I_1^{\mathrm{s}}) & \mathbf{A}(I_1^{\mathrm{r}}, I_1^{\mathrm{r}}) \end{bmatrix} \begin{bmatrix} \mathbf{I} & -\mathbf{T} \\ \mathbf{0} & \mathbf{I} \end{bmatrix}. \end{aligned}$$

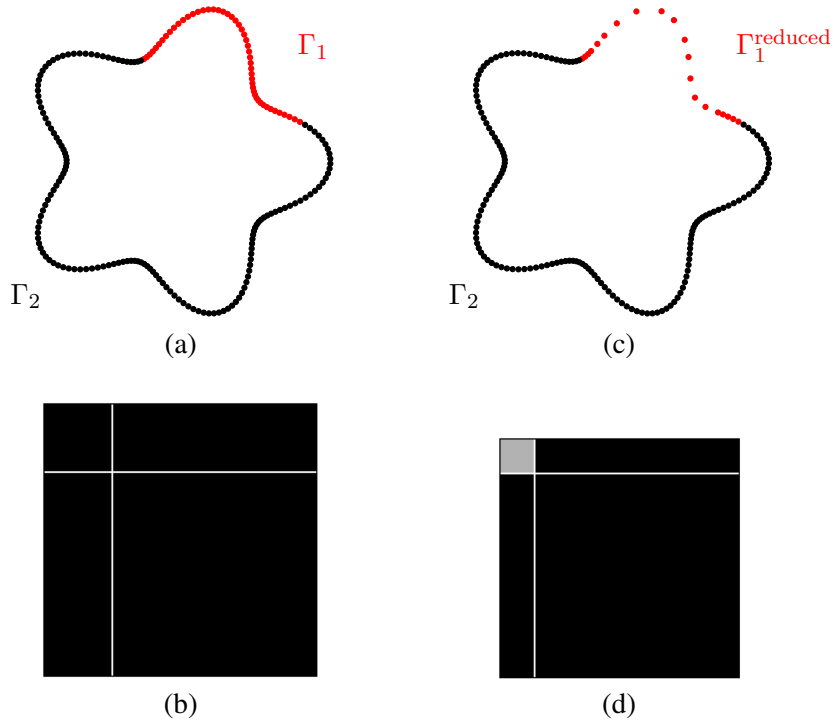


FIGURE 4.1. (a) The contour Γ is partitioned into pieces Γ_1 and Γ_2. (b) The linear system blocked in accordance with the partitioning of the contour, cf. (4.14). (c) The points remaining after compressing Γ_1. (d) The reduced linear system (4.17); observe that only the small gray block involves new matrix entries that need to be computed.

Multiplying the three matrices together, we obtain the formulas

$$\begin{aligned}
\tilde{\mathbf{A}}_{1,1}^{\mathrm{ss}} &= \mathbf{A}(I_1^{\mathrm{s}}, I_1^{\mathrm{s}}),\\
\tilde{\mathbf{A}}_{1,1}^{\mathrm{sr}} &= \mathbf{A}(I_1^{\mathrm{s}}, I_1^{\mathrm{r}}) - \mathbf{A}(I_1^{\mathrm{s}}, I_1^{\mathrm{s}})\mathbf{T},\\
\tilde{\mathbf{A}}_{1,1}^{\mathrm{rs}} &= \mathbf{A}(I_1^{\mathrm{r}}, I_1^{\mathrm{s}}) - \mathbf{S}^*\mathbf{A}(I_1^{\mathrm{s}}, I_1^{\mathrm{s}}),\\
\tilde{\mathbf{A}}_{1,1}^{\mathrm{rr}} &= \mathbf{A}(I_1^{\mathrm{r}}, I_1^{\mathrm{r}}) - \mathbf{S}^*\mathbf{A}(I_1^{\mathrm{s}}, I_1^{\mathrm{r}}) - \mathbf{A}(I_1^{\mathrm{r}}, I_1^{\mathrm{s}})\mathbf{T} + \mathbf{S}^*\mathbf{A}(I_1^{\mathrm{s}}, I_1^{\mathrm{s}})\mathbf{T}.
\end{aligned}$$

Eliminating the residual variables, we now end up with the compressed linear system

$$\begin{bmatrix} \tilde{\mathbf{D}}_1 & \mathbf{A}(I_1^{\mathrm{s}}, I_2) \\ \mathbf{A}(I_2, I_1^{\mathrm{s}}) & \mathbf{A}(I_2, I_2) \end{bmatrix} \begin{bmatrix} \tilde{\mathbf{q}}_1 \\ \mathbf{q}_2 \end{bmatrix} = \begin{bmatrix} \tilde{\mathbf{f}}_1^{\mathrm{s}} - \tilde{\mathbf{A}}_{1,1}^{\mathrm{sr}} (\tilde{\mathbf{A}}_{1,1}^{\mathrm{rr}})^{-1} \tilde{f}_1^{\mathrm{r}} \\ \mathbf{f}_2 \end{bmatrix}, \tag{4.17}$$

where the compressed diagonal block is defined by

$$\tilde{\mathbf{D}}_1 = \tilde{\mathbf{A}}_{1,1}^{\mathrm{ss}} - \tilde{\mathbf{A}}_{1,1}^{\mathrm{sr}} (\tilde{\mathbf{A}}_{1,1}^{\mathrm{rr}})^{-1} \tilde{\mathbf{A}}_{1,1}^{\mathrm{rs}}.$$

More generally, we can apply Lemma 4.1, to find that the inverse of the matrix $\mathbf{A}$ takes the form

$$\mathbf{A}^{-1} = \begin{bmatrix} \mathbf{E}_1 & \mathbf{0} \\ \mathbf{0} & \mathbf{I} \end{bmatrix} \left(\begin{bmatrix} \tilde{\mathbf{D}}_1 & \mathbf{A}(I_1^{\mathrm{s}}, I_2) \\ \mathbf{A}(I_2, I_1^{\mathrm{s}}) & \mathbf{A}(I_2, I_2) \end{bmatrix} \right)^{-1} \begin{bmatrix} \mathbf{F}_1^* & \mathbf{0} \\ \mathbf{0} & \mathbf{I} \end{bmatrix} + \begin{bmatrix} \mathbf{G}_1 & \mathbf{0} \\ \mathbf{0} & \mathbf{0} \end{bmatrix},$$

(1) **loop** over all nodes τ,
(2) Form the *complement* of the index vector: $L_\tau = I \backslash I_\tau$.
(3) Determine the row skeletons: $[\mathbf{U}_\tau, J_\tau^{\mathrm{row}}] = \mathtt{id\_row}(\mathbf{A}(I_\tau, L_\tau), k)$.
(4) Determine the column skeletons: $[\mathbf{V}_\tau, J_\tau^{\mathrm{col}}] = \mathtt{id\_row}(\mathbf{A}(L_\tau, I_\tau)^*, k)$.
(5) Extract the skeleton index vectors $I_\tau^{\mathrm{s,row}} = I_\tau(J_\tau^{\mathrm{row}})$, and $I_\tau^{\mathrm{s,col}} = I_\tau(J_\tau^{\mathrm{col}})$.
(6) **end loop**

FIGURE 4.2. This algorithm computes an ID based HBS decomposition of a matrix $\mathbf{A}$, for a specified rank k. It is a single level scheme, as described in Sections 1.4 and 4.4. The particular advantage of using the ID is that the sibling interaction matrices need not be computed explicitly, as they are identified by the skeleton index vectors, $\tilde{\mathbf{A}}_{\sigma,\tau} = \mathbf{A}(I_\sigma^{\mathrm{s,row}}, I_\tau^{\mathrm{s,col}})$.

where

$$\mathbf{E}_1 = \mathbb{V}_1^{-*} \begin{bmatrix} \mathbf{I} \\ -(\tilde{\mathbf{A}}_{1,1}^{\mathrm{rr}})^{-1}\tilde{\mathbf{A}}_{1,1}^{\mathrm{rs}} \end{bmatrix} = \mathbf{P}_1 \begin{bmatrix} \mathbf{I} & -\mathbf{T} \\ \mathbf{0} & \mathbf{I} \end{bmatrix} \begin{bmatrix} \mathbf{I} \\ -(\tilde{\mathbf{A}}_{1,1}^{\mathrm{rr}})^{-1}\tilde{\mathbf{A}}_{1,1}^{\mathrm{rs}} \end{bmatrix} = \mathbf{P}_1 \begin{bmatrix} \mathbf{I} + \mathbf{T}(\tilde{\mathbf{A}}_{1,1}^{\mathrm{rr}})^{-1}\mathbf{A}_{1,1}^{\mathrm{r,s}} \\ -(\tilde{\mathbf{A}}_{1,1}^{\mathrm{rr}})^{-1}\mathbf{A}_{1,1}^{\mathrm{r,s}} \end{bmatrix},$$

$$\begin{aligned} \mathbf{F}_1^* &= \begin{bmatrix} \mathbf{I} & -\tilde{\mathbf{A}}_{1,1}^{\mathrm{sr}}(\tilde{\mathbf{A}}_{1,1}^{\mathrm{rr}})^{-1} \end{bmatrix} \mathbb{U}_1^{-1} = \begin{bmatrix} \mathbf{I} & -\tilde{\mathbf{A}}_{1,1}^{\mathrm{sr}}(\tilde{\mathbf{A}}_{1,1}^{\mathrm{rr}})^{-1} \end{bmatrix} \begin{bmatrix} \mathbf{I} & \mathbf{0} \\ -\mathbf{S}^* & \mathbf{I} \end{bmatrix} \mathbf{P}_1^* \\ &= \begin{bmatrix} \mathbf{I} + \tilde{\mathbf{A}}_{1,1}^{\mathrm{sr}}(\tilde{\mathbf{A}}_{1,1}^{\mathrm{rr}})^{-1}\mathbf{S}^* & -\tilde{\mathbf{A}}_{1,1}^{\mathrm{sr}}(\tilde{\mathbf{A}}_{1,1}^{\mathrm{rr}})^{-1} \end{bmatrix} \mathbf{P}_1^*, \end{aligned}$$

$$\begin{aligned} \mathbf{G}_1 &= \mathbb{V}_1^{-*} \begin{bmatrix} \mathbf{0} & \mathbf{0} \\ \mathbf{0} & (\tilde{\mathbf{A}}_{1,1}^{\mathrm{rr}})^{-1} \end{bmatrix} \mathbb{U}_1^{-1} = \mathbf{P}_1 \begin{bmatrix} \mathbf{I} & -\mathbf{T} \\ \mathbf{0} & \mathbf{I} \end{bmatrix} \begin{bmatrix} \mathbf{0} & \mathbf{0} \\ \mathbf{0} & (\tilde{\mathbf{A}}_{1,1}^{\mathrm{rr}})^{-1} \end{bmatrix} \begin{bmatrix} \mathbf{I} & \mathbf{0} \\ -\mathbf{S}^* & \mathbf{I} \end{bmatrix} \mathbf{P}_1^* \\ &= \mathbf{P}_1 \begin{bmatrix} -\mathbf{T} \\ \mathbf{I} \end{bmatrix} (\tilde{\mathbf{A}}_{1,1}^{\mathrm{rr}})^{-1} \begin{bmatrix} -\mathbf{S}^* & \mathbf{I} \end{bmatrix} \mathbf{P}_1^*. \end{aligned}$$

4.4. Compression using the ID — "skeletonization"

In Section 4.3, we saw how the ID leads to more economical expressions for the matrices that need to be computed in order to invert or factorize an HBS matrix. Let us next address the question of how to find the compressed representation in the first place. In principle, the answer is straight forwards, we simply apply the procedure described in Section 1.4, but now use the ID to factorize each off-diagonal blocks. In other words, for any block τ, the basis matrices $\mathbf{U}_\tau$ and $\mathbf{V}_\tau$ and the associated skeleton index vectors are computed via the commands

$$[\mathbf{U}_\tau, J_\tau^{\mathrm{row}}] = \mathtt{id\_row}(\mathbf{A}(I_\tau, I_\tau^{\mathrm{c}}), k), \qquad \text{and} \qquad [\mathbf{V}_\tau, J_\tau^{\mathrm{col}}] = \mathtt{id\_row}(\mathbf{A}(I_\tau^{\mathrm{c}}, I_\tau)^*, k),$$

and then the index vectors are derived via the formulas

$$I_\tau^{\mathrm{s,row}} = I_\tau(J_\tau^{\mathrm{rol}}), \qquad \text{and} \qquad I_\tau^{\mathrm{s,col}} = I_\tau(J_\tau^{\mathrm{col}}).$$

If we seek a symmetric factorization in which $I_\tau^{\mathrm{s,col}} = I_\tau^{\mathrm{s,row}}$ and $\mathbf{U}_\tau = \mathbf{V}_\tau$, then we simply factorize the blocks jointly, so that

$$[\mathbf{U}_\tau, J_\tau] = \mathtt{id\_row}(\begin{bmatrix} \mathbf{A}(I_\tau, I_\tau^{\mathrm{c}}) & \mathbf{A}(I_\tau^{\mathrm{c}}, I_\tau)^* \end{bmatrix}, k), \qquad \text{and} \qquad I_\tau^{\mathrm{s}} = I_\tau(J_\tau).$$

Enforcing symmetry this way turns out to not only simplify formulas and reducing storage, but it can also improve the numerical stability of the inversion procedure. The price to pay is that the interaction ranks tends to grow slightly. The corresponding compression algorithms are for future reference summarized in Figures 4.2 and 4.3.

(1) **loop** over all nodes τ,
(2) Form the complement of the index vector: $L_\tau = I \backslash I_\tau$.
(3) Form the extended off-diagonal block: $\mathbf{W}_\tau = \big[\mathbf{A}(I_\tau, L_\tau) \quad \mathbf{A}(L_\tau, I_\tau)^*\big]$.
(4) Factorize the extended off-diagonal block: $[\mathbf{U}_\tau, J_\tau] = \texttt{id\_row}(\mathbf{W}_\tau, k)$.
(5) Extract the reduced ("skeletonized") index vector $I^{\mathrm{s}}_\tau = I_\tau(J_\tau)$.
(6) **end loop**

FIGURE 4.3. A single level compression scheme using the ID and the HBS format. The algorithm is identical to the one in Figure 4.2, except that we now enforce that the row and the column skeletons be identical.

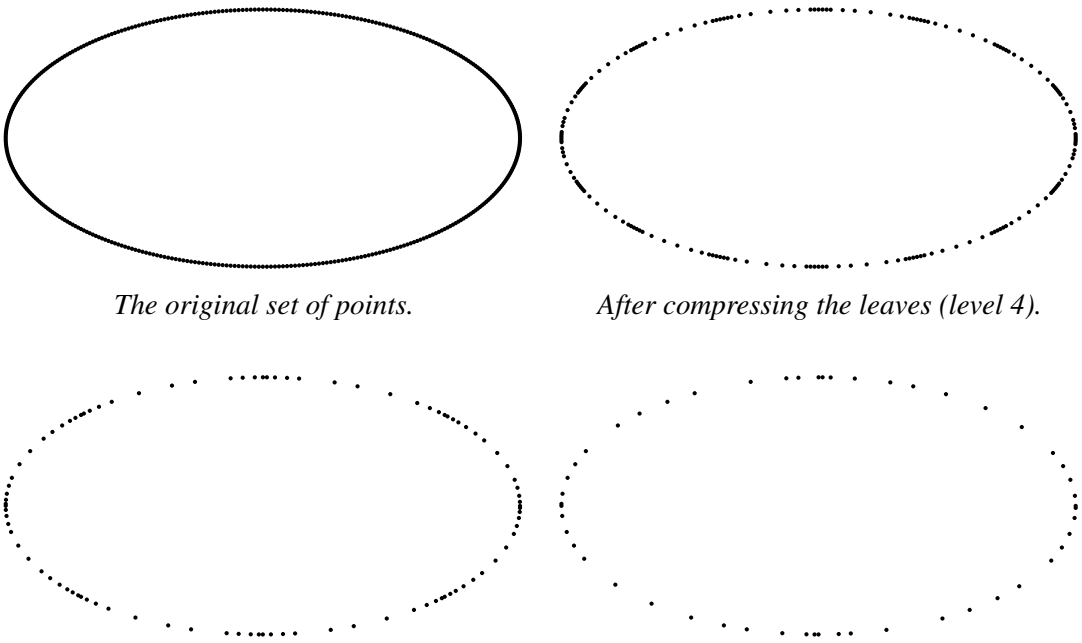
(1) **loop** over all levels, finer to coarser, $\ell = L,\ L-1,\ \ldots, 1$
(2) **loop** over all boxes τ on level ℓ,
(3) **if** τ is a leaf node
(4) Set $M_\tau = I_\tau$ and form its complement $L_\tau = I^{\mathrm{c}}_\tau$.
(5) **else**
(6) Let α and β denote the children of τ.
(7) Form the vector of skeleton points of the children: $M_\tau = [I^{\mathrm{s}}_\alpha,\ I^{\mathrm{s}}_\beta]$.
(8) Form the complement of the index vector $L_\tau = I^{(\ell+1)} \backslash M_\tau$.
(9) **end if**
(10) Form the off-diagonal block: $\mathbf{W}_\tau = \big[\mathbf{A}(M_\tau, L_\tau) \quad \mathbf{A}(L_\tau, M_\tau)^*\big]$.
(11) Factor the off-diagonal block: $[\mathbf{U}_\tau, J_\tau] = \texttt{id\_row}(\mathbf{W}_\tau, \varepsilon)$.
(12) Build the vector of "skeleton" points $I^{\mathrm{s}}_\tau = M_\tau(J_\tau)$.
(13) **end loop**
(14) Form the level ℓ compressed index vector: $I^{(\ell)} = \bigcup_{\tau \text{ is on level } \ell} I^{\mathrm{s}}_\tau$.
(15) **end loop**

FIGURE 4.4. Recursive skeletonization: Let $\mathbf{A}$ denote a given matrix, and let $\mathcal{T}$ be a tree on the index vector, as described in Section 2.2. Then given a tolerance ε, the algorithm described will build a symmetric skeletonization of $\mathbf{A}$. In other words, for every node τ, it builds the index vector I^{s}_τ that holds the skeleton nodes, and the matrix of basis vectors $\mathbf{U}_\tau$ needed in the HBS representation of $\mathbf{A}$. This is a brute force compression scheme with complexity $O(N^2)$. It is the foundation for the accelerated schemes described in Chapter 5.

4.5. Multilevel inversion: "recursive skeletonization"

Extending the single level skeletonization solver to a multi level solver is remarkably simple. The essential observation is that the off-diagonal blocks that need to be compressed are identified simply as submatrices of the original off-diagonal blocks, as identified by the skeleton index vectors computed on the next finer level. The resulting procedure is summarized in Figure 4.4.

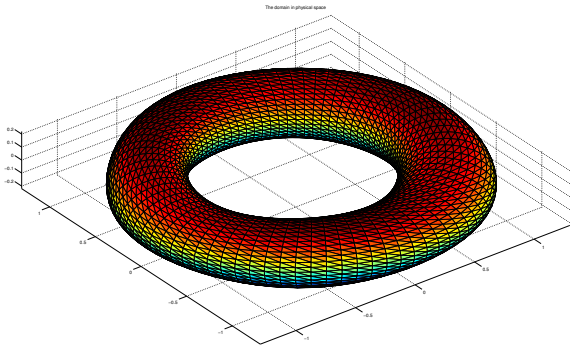
The recursive skeletonization procedure can easily by visualized by plotting the points remaining after processing each level. Figure 4.5 illustrates the procedure for a contour in two dimensions by plotting for each level ℓ the nodes in the compressed $\tilde{I}^{(\ell)}$ that "survived" compression. Figure 4.6 shows the analogous computation for a surface in 3D.



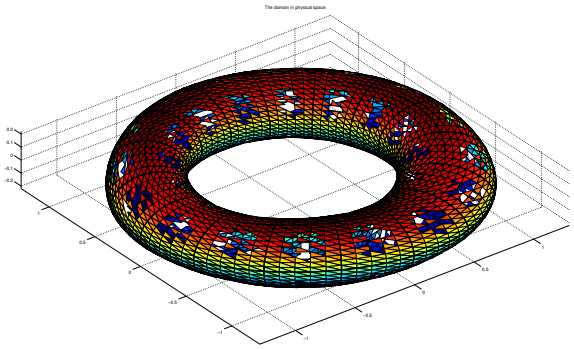
The original set of points. *After compressing the leaves (level 4).*

After compressing level 3. *After compressing level 2.*

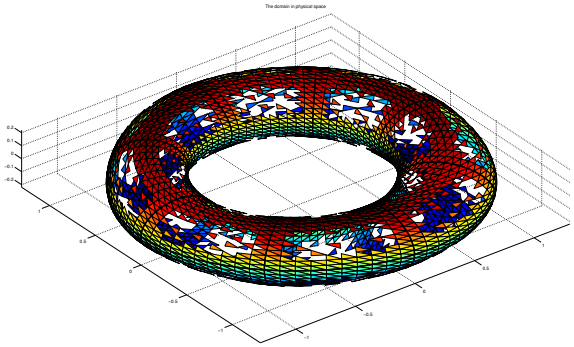
FIGURE 4.5. Recursive skeletonization on a contour in the plane.



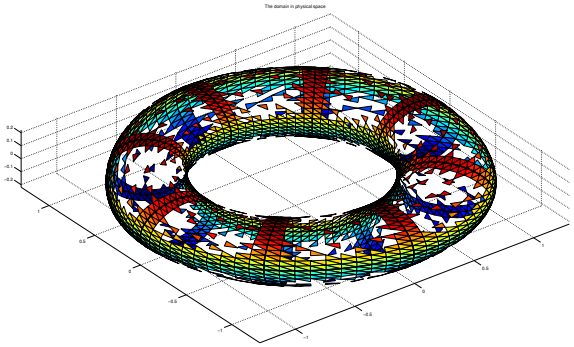
The original set of points.



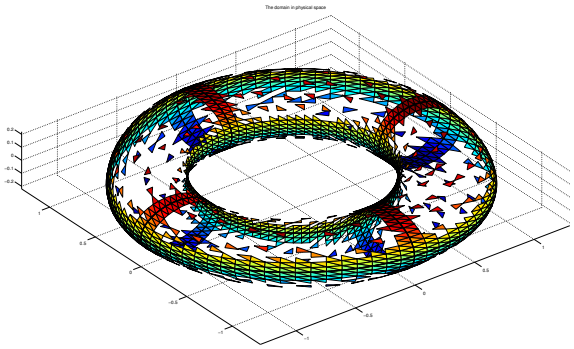
After compressing the leaves (level 6).



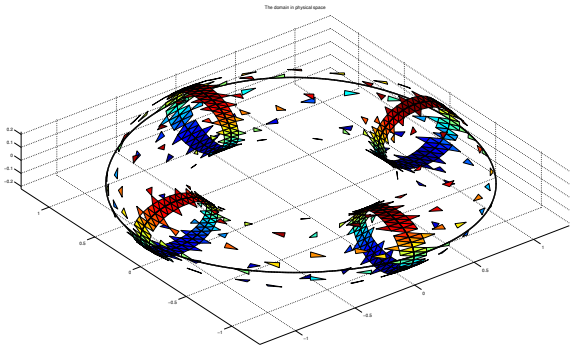
After compressing level 5.



After compressing level 4.



After compressing level 3.



After compressing level 2.

FIGURE 4.6. Recursive skeletonization on a surface in $\mathbb{R}^3$.

CHAPTER 5

Constructing a rank-structured representation of a matrix

In Chapters 1 – 4, we described a data-sparse format that is well suited for many boundary integral equations, and described how a matrix stored in such a format can efficiently be stored, applied to a vector, and even inverted. We have also (cf. Sections 1.4, 4.4, and 4.5) described some conceptually natural ways to compute the collection of small matrices that jointly form the rank-structured representation of the matrix. These techniques were based on explicitly building the full $N \times N$ matrix $\mathbf{A}$, then looping over various off-diagonal blocks that are of numerically low rank and compressing them using any of the techniques described in Chapter ??. While such an approach tends to lead to the most lean compressed representation possible, it is typically unaffordable with an $O(N^2)$ overall complexity. In this chapter, we describe techniques that in many situations achieve the same objective in $O(N)$ operations.

5.1. A heuristic description of the proxy surface compression technique

The key challenge in reducing the computational complexity in a compression algorithm is that computations need to be localized. A successful way to achieve localization is to exploit analytical properties in the kernels of the integral equations. We will describe the techniques in detail in Section 5.2, but just to introduce the key idea, let us suppose that we consider a BIE on a contour Γ, and seek to compress the interactions between a patch Γ_τ and the rest of the contour, as illustrated in Figure 5.1(a). The idea is to enclose Γ_τ by a "proxy" contour $\Gamma_\tau^{(\text{proxy})}$, cf. Figure 5.1(b). Then we know from potential theory that any harmonic field on Γ_τ caused by sources outside of $\Gamma_\tau^{(\text{proxy})}$ can be exactly replicated by a source distribution on $\Gamma_\tau^{(\text{proxy})}$ alone. This means that if we can build a local basis on Γ_τ that to some given precision ε spans every field generated by sources on $\Gamma_\tau^{(\text{proxy})}$, then this basis will automatically span all fields generated by sources on the parts of Γ that are outside of $\Gamma_\tau^{(\text{proxy})}$, and we are left with the inexpensive task of compressing only the interactions between points in Γ_τ and those in the "near" part of the contour $\Gamma_\tau^{(\text{near})}$. The analogous situation in 3D is illustrated in Figure 5.2.

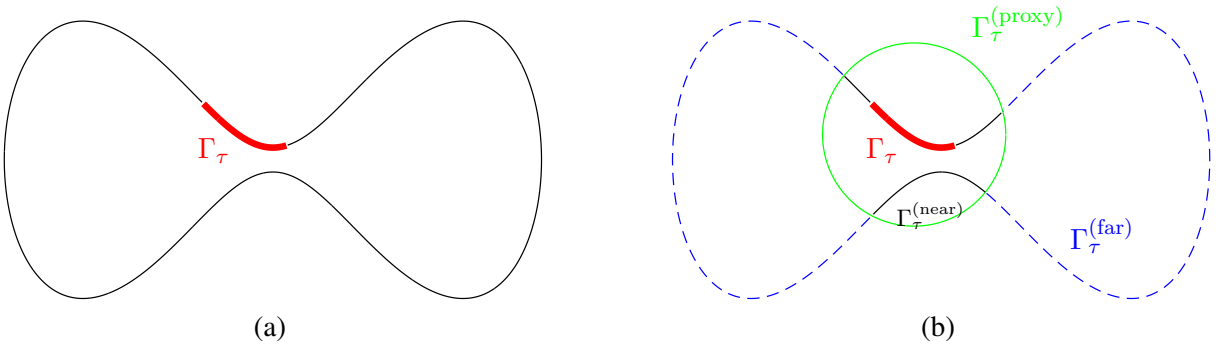


FIGURE 5.1. A contour Γ. (a) Γ_τ is drawn with a bold line. (b) The contour $\Gamma_\tau^{(\text{near})}$ is drawn with a thin solid line and $\Gamma_\tau^{(\text{far})}$ with a dashed line.

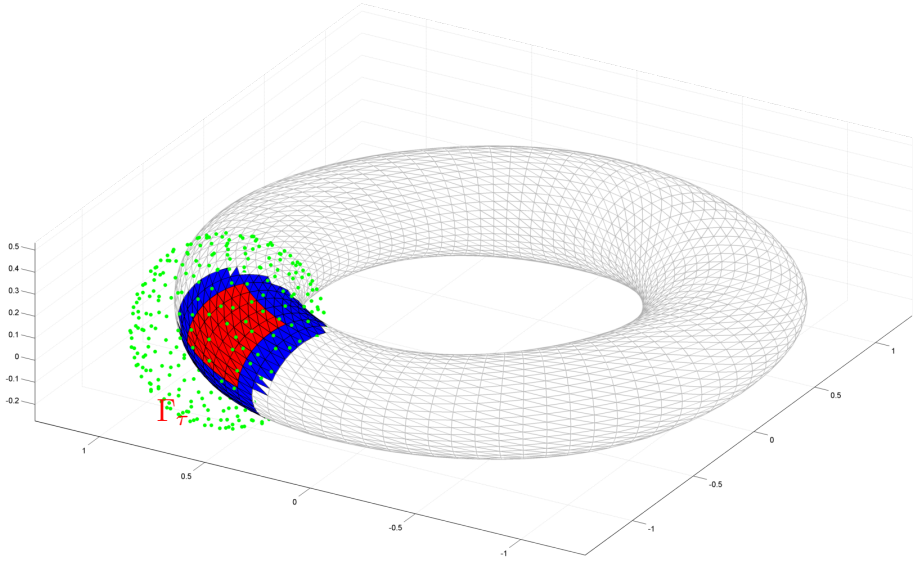


FIGURE 5.2. The geometry of the "proxy surface" described in Section 5.1 for a 3D compression problem.

5.2. The proxy surface compression techniques

Let us start by considering the single-level skeletonization technique described in Figure 4.3. We start by discussing how to compress a matrix $\mathbf{A}$ whose entries are given simply by the fundamental solution ϕ of the Laplace equation, so that

$$\mathbf{A}(i,j) = \phi(\boldsymbol{x}_i - \boldsymbol{x}_j), \qquad \text{for } i \neq j, \tag{5.1}$$

where $\{\boldsymbol{x}_i\}_{i=1}^N$ are the quadrature nodes on Γ. The expensive computation that we seek to eliminate is the formation and compression of the long block $\mathbf{W}_\tau$ on line (10). We first note that we do not actually need the off-diagonal blocks in $\mathbf{W}_\tau$ themselves; all we need is the matrix $\mathbf{U}_\tau$ and the index vector J_τ. We are going to construct these by factoring a matrix $\mathbf{W}_\tau^{\mathrm{proxy}}$ that is much smaller than $\mathbf{W}_\tau$, but has the property that the column space of $\mathbf{W}_\tau^{\mathrm{proxy}}$ contains the column space of $\mathbf{W}_\tau$. In other words

$$\mathrm{Col}(\mathbf{W}_\tau) \subseteq \mathrm{Col}(\mathbf{W}_\tau^{\mathrm{proxy}}).$$

To form $\mathbf{W}_\tau^{\mathrm{proxy}}$, we proceed as follows: A small number of columns in $\mathbf{W}_\tau$ represent interactions between points in I_τ and points that are "near" Γ_τ. These columns we bring in to $\mathbf{W}_\tau^{\mathrm{proxy}}$ unchanged. The remaining columns of $\mathbf{W}_\tau$ represent far-field interactions, these make up the vast majority of the columns. But these are all samples from a very low-dimensional space, namely the space of harmonic fields induced by charges that are distant from Γ_τ. In forming $\mathbf{W}_\tau^{\mathrm{proxy}}$, we replace all columns corresponding to far-field interactions with a much smaller set of vectors that exhaustively span all possible far fields (to within precision ε).

To formalize matters, we must first make precise what we mean by *far-field* and *near-field*. Let Ψ_τ denote a circle of minimal radius that encloses Γ_τ, and let $\Gamma_\tau^{(\mathrm{proxy})}$ denote a circle concentric to Ψ_τ but with a radius 1.5 times as large (the number "1.5" is chosen somewhat arbitrarily but is usually a good choice), see Figure 5.1. Now define I_τ^{near} as all points inside $\Gamma_\tau^{(\mathrm{proxy})}$, but not in I_τ, and let I_τ^{far} denote all remaining points. In other words,

$$I = I_\tau \cup \underbrace{I_\tau^{\mathrm{near}} \cup I_\tau^{\mathrm{far}}}_{=L_\tau},$$

forms a disjoint partition of the index vector I. Then

$$\mathbf{W}_\tau = \begin{bmatrix} \mathbf{A}(I_\tau, I_\tau^{\mathrm{near}}) & \mathbf{A}(I_\tau^{\mathrm{near}}, I_\tau)^* & | & \mathbf{A}(I_\tau, I_\tau^{\mathrm{far}}) & \mathbf{A}(I_\tau^{\mathrm{far}}, I_\tau)^* \end{bmatrix} \mathbf{P},$$

(1) Set $I^{(L)} = 1 : N$.
(2) **loop** over all levels, finer to coarser, $\ell = L,\ L-1,\ \ldots, 1$
(3) **loop** over all boxes τ on level ℓ,
(4) **if** τ is a leaf node
(5) Set $M_\tau = I_\tau$.
(6) **else**
(7) Let α and β denote the children of τ.
(8) Form the vector of skeleton points of the children: $M_\tau = [\tilde{I}_\alpha,\ \tilde{I}_\beta]$.
(9) **end if**
(10) Find the index vector I_τ^{near} of near-field points within $I^{(\ell)}$.
(11) Form the basis for the far field $\mathbf{A}_\tau^{\text{proxy}}$, as described in Section 5.2.
(12) Assemble the proxy matrix $\mathbf{W}_\tau^{\text{proxy}} = \begin{bmatrix}\mathbf{A}(M_\tau, I_\tau^{\text{near}}) & \mathbf{A}(I_\tau^{\text{near}}, M_\tau)^* & \mathbf{A}_\tau^{\text{proxy}}\end{bmatrix}$.
(13) Factor the proxy matrix: $[\mathbf{U}_\tau, J_\tau] = \texttt{id\_row}(\mathbf{W}_\tau^{\text{proxy}}, \varepsilon)$.
(14) Extract the reduced ("skeletonized") index vector $I_\tau^{\text{s}} = M_\tau(J_\tau)$.
(15) **end loop**
(16) Form the level ℓ compressed index vector: $I^{(\ell)} = \bigcup_{\tau \text{ is on level } \ell} I_\tau^{\text{s}}$.
(17) **end loop**

FIGURE 5.3. Fast recursive skeletonization: Given a computational tolerance ε, this algorithm builds the basis matrices and the index vectors that jointly form the ID version of the HBS representation of a matrix.

where $\mathbf{P}$ is an permutation matrix that reorders the columns. To form $\mathbf{W}_\tau^{\text{proxy}}$, we replace the columns to the right of the divider (which represent far-field interactions) by a matrix $\mathbf{A}_\tau^{\text{proxy}}$ that samples the space of harmonic fields that can be caused by sources outside $\Gamma_\tau^{(\text{proxy})}$. This sampling is done by placing a small number of "proxy charges" on $\Gamma_\tau^{(\text{proxy})}$. Then

$$\text{Col}\left(\begin{bmatrix}\mathbf{A}(I_\tau, I_\tau^{\text{far}}) \\ \mathbf{A}(I_\tau^{\text{far}} I_\tau)^*\end{bmatrix}\right) \subseteq \text{Col}(\mathbf{A}_\tau^{\text{proxy}}).$$

The matrix $\mathbf{W}_\tau^{\text{proxy}}$ is then given by the formula

$$\mathbf{W}_\tau^{\text{proxy}} = \begin{bmatrix}\mathbf{A}(I_\tau, I_\tau^{\text{near}}) & \mathbf{A}(I_\tau^{\text{near}}, I_\tau)^* & \mathbf{A}_\tau^{\text{proxy}}\end{bmatrix}.$$

When the proxy surface technique is incorporated into the algorithm for recursive skeletonization in Figure 4.4, we obtain the accelerated recursive skeletonization method of Figure 5.3.

So far, we have considered only the particularly simple case when the matrix entries are given by $\mathbf{A}(i,j) = \phi(\boldsymbol{x}_i - \boldsymbol{x}_j)$, cf. (5.1). Let us very briefly describe how the scheme can be modified to handle more realistic kernels. For instance, suppose that the matrix is associated with the double rather than the single layer potential, so that

$$\mathbf{A}(i,j) = \boldsymbol{n}(\boldsymbol{y}) \cdot \nabla_{\boldsymbol{y}} \phi(\boldsymbol{x} - \boldsymbol{y}).$$

In this case, the matrix $\mathbf{A}(I_\tau, I_\tau^{\text{far}})$ represents evaluation of a harmonic field on Γ_τ caused by dipole charges on Γ_τ^{far}. These can be represented in exactly the same way, so no change is required. However, $\mathbf{A}(I_\tau^{\text{far}}, I_\tau)^*$ is the *dual* of the double layer potential, and represents evaluation of *normal derivatives* of a field generated by monopoles on $\Gamma_\tau^{(\text{far})}$. We generate a basis by adding columns to $\mathbf{A}_\tau^{\text{proxy}}$ that represent normal derivatives of fields generated by monopoles on $\Gamma_\tau^{(\text{proxy})}$.

Another complication that arises in real applications is that the matrix entries also incorporate effects from the quadrature rule that is used to discretize the BIE. For the majority of the elements, this effect represents only a diagonal scaling, which presents no difficulty — we simply build a $\mathbf{A}_\tau^{\text{proxy}}$ that represents the

Generate an $N \times \ell$ Gaussian random matrix $\mathbf{R}$.
Evaluate $\mathbf{S} = \mathbf{A}\,\mathbf{R}$ using the fast matrix-vector multiplier.
loop over levels, finer to coarser, $\ell = L,\ L-1,\ \ldots,\ 2,\ 1$
  loop over all nodes τ on level ℓ
    if τ is a leaf node then
      $I_{\text{loc}} = I_\tau$
      $\mathbf{R}_{\text{loc}} = \mathbf{R}(I_\tau, :)$
      $\mathbf{S}_{\text{loc}} = \mathbf{S}(I_\tau, :) - \mathbf{A}(I_\tau, I_\tau)\,\mathbf{R}_{\text{loc}}$
    else
      Let ν_1 and ν_2 be the two children of τ.
      $I_{\text{loc}} = [\tilde{I}_{\nu_1},\ \tilde{I}_{\nu_2}]$
      $\mathbf{R}_{\text{loc}} = \begin{bmatrix} \mathbf{R}_{\nu_1} \\ \mathbf{R}_{\nu_2} \end{bmatrix}$
      $\mathbf{S}_{\text{loc}} = \begin{bmatrix} \mathbf{S}_{\nu_1} - \mathbf{A}(\tilde{I}_{\nu_1}, \tilde{I}_{\nu_2})\,\mathbf{R}_{\nu_2} \\ \mathbf{S}_{\nu_2} - \mathbf{A}(\tilde{I}_{\nu_2}, \tilde{I}_{\nu_1})\,\mathbf{R}_{\nu_1} \end{bmatrix}$
    end if
    $[\mathbf{U}_\tau, J_\tau] = \texttt{id\_decomp}(\mathbf{S}^*_{\text{loc}})$
    $\mathbf{R}_\tau = \mathbf{U}^*_\tau\,\mathbf{R}_{\text{loc}}$
    $\mathbf{S}_\tau = \mathbf{S}_{\text{loc}}(J_\tau, :)$
    $\tilde{I}_\tau = I_{\text{loc}}(J_\tau)$
  end loop
end loop
For all leaf nodes τ, set $\mathbf{D}_\tau = \mathbf{A}(I_\tau,\ I_\tau)$.
For all sibling pairs $\{\nu_1,\ \nu_2\}$ set $\tilde{\mathbf{A}}_{\nu_1,\nu_2} = \mathbf{A}(\tilde{I}_{\nu_1},\ \tilde{I}_{\nu_2})$.

FIGURE 5.4. Randomized compression of an HBS matrix.

"pure" kernel function, and then add the diagonal scaling at the end. Quadrature corrections that arise due to singular kernels are in general more fundamental than mere diagonal scalings. However, they typically affect only entries in the near field matrices, which means that the scheme as described handles them automatically, with no need for any modifications, except that it may be prudent to explicitly ensure at each step that all "modified" matrix entries truly are in the near field, and to include a mechanism for including these entries in $\mathbf{W}_\tau^{\text{proxy}}$ if needed.

The generalization from Laplace to other fundamental solutions is in principle straight-forward, but does require some care in practice. For instance, if ϕ is the fundamental solution of the Helmholtz equation, then it is not necessarily the case that every field ψ that satisfies $-\Delta\psi - \kappa^2\psi = 0$ near Γ_τ can be stably represented by placing monopoles on $\Gamma_\tau^{(\text{proxy})}$. The situation can be stabilized by placing *both* monopoles *and* dipoles on $\Gamma_\tau^{(\text{proxy})}$, or by adding a second proxy surface that is separated by a quarter wavelength from the first.

5.3. Randomized compression of HBS matrices

5.4. Randomized compression of HODLR matrices

5.5. Adaptive Cross Approximation

Build compressed representations of all off-diagonal blocks.
loop over levels $\ell = 0 : (L-1)$

Build the random matrices $\mathbf{\Omega}_1$ and $\mathbf{\Omega}_2$.
$\mathbf{\Omega}_1 = \texttt{zeros}(n, r)$
$\mathbf{\Omega}_2 = \texttt{zeros}(n, r)$
loop over boxes τ on level ℓ
Let $\{\alpha, \beta\}$ denote the children of box τ.
$\mathbf{\Omega}_1(I_\alpha, :) = \texttt{randn}(n_\alpha, r)$
$\mathbf{\Omega}_2(I_\beta, :) = \texttt{randn}(n_\beta, r)$
end loop

Apply $\mathbf{A}$ to build the samples for the incoming basis matrices.
$\mathbf{Y}_1 = \mathbf{A}\mathbf{\Omega}_2 - \mathbf{A}^{(\ell)}\mathbf{\Omega}_2$
$\mathbf{Y}_2 = \mathbf{A}\mathbf{\Omega}_1 - \mathbf{A}^{(\ell)}\mathbf{\Omega}_1$

Orthonormalize the sample matrices to build the incoming basis matrices.
loop over boxes τ on level ℓ
Let $\{\alpha, \beta\}$ denote the children of box τ.
$\mathcal{U}_\alpha = \texttt{qr}(\mathbf{Y}_1(I_\alpha, :))$.
$\mathcal{U}_\beta = \texttt{qr}(\mathbf{Y}_2(I_\beta, :))$.
$\mathbf{\Omega}_1(I_\alpha, :) = \mathcal{U}_\alpha$
$\mathbf{\Omega}_2(I_\beta, :) = \mathcal{U}_\beta$
end loop

Apply $\mathbf{A}^*$ to build the samples for the outgoing basis matrices.
$\mathbf{Z}_1 = \mathbf{A}^*\mathbf{\Omega}_2 - \left(\mathbf{A}^{(\ell)}\right)^*\mathbf{\Omega}_2$
$\mathbf{Z}_2 = \mathbf{A}^*\mathbf{\Omega}_1 - \left(\mathbf{A}^{(\ell)}\right)^*\mathbf{\Omega}_1$

Take local SVDs to build incoming basis matrices and sibling interaction matrices. We determine the actual rank, and update the $\mathcal{U}_*$ basis matrices accordingly.
loop over boxes τ on level ℓ
Let $\{\alpha, \beta\}$ denote the children of box τ.
$[\mathcal{U}_\alpha, \mathbf{B}_{\beta,\alpha}, \hat{\mathbf{U}}_\beta] = \texttt{svd}(\mathbf{Z}_1(I_\alpha, :), \varepsilon)$.
$[\mathcal{U}_\beta, \mathbf{B}_{\alpha,\beta}, \hat{\mathbf{V}}_\alpha] = \texttt{svd}(\mathbf{Z}_2(I_\beta, :), \varepsilon)$.
$\mathcal{U}_\beta \leftarrow \mathcal{U}_\beta \hat{\mathbf{U}}_\beta$.
$\mathcal{U}_\alpha \leftarrow \mathcal{U}_\alpha \hat{\mathbf{U}}_\alpha$.
end loop
end loop

Extract the diagonal matrices.
$n_{\max} = \texttt{max}\{n_\tau : \tau \text{ is a leaf}\}$
$\mathbf{\Omega} = \texttt{zeros}(N, n_{\max})$
loop over leaf boxes τ
$\mathbf{\Omega}(I_\tau, 1 : n_\tau) = \texttt{eye}(n_\tau)$.
end loop
$\mathbf{Y} = \mathbf{A}\mathbf{\Omega} - \mathbf{A}^{(L)}\mathbf{\Omega}$
loop over leaf boxes τ
$\mathbf{D}_\tau = \mathbf{Y}(I_\tau, 1 : n_\tau)$.
end loop

FIGURE 5.5. Randomized compression of a HODLR matrix.

Bibliography

[1] W.C. Chew, J.-M. Jin, E. Michielssen, and J. Song, *Fast and efficient algorithms in computational electromagnetics*, Artech House, 2001.

[2] Adrianna Gillman, Patrick Young, and Per-Gunnar Martinsson, *A direct solver $o(n)$ complexity for integral equations on one-dimensional domains*, Frontiers of Mathematics in China **7** (2012), 217–247, 10.1007/s11464-012-0188-3.

[3] K.L. Ho and L. Greengard, *A fast direct solver for structured linear systems by recursive skeletonization*, SIAM Journal on Scientific Computing **34** (2012), no. 5, 2507–2532.

[4] Y. J. Liu, *Fast multipole boundary element method: Theory and applications in engineering*, Cambridge University Press, 2009.

[5] Y.J. Liu and N. Nishimura, *The fast multipole boundary element method for potential problems: A tutorial*, Engineering Analysis with Boundary Elements **30** (2006), no. 5, 371 – 381.

[6] P.G. Martinsson and V. Rokhlin, *A fast direct solver for boundary integral equations in two dimensions*, J. Comp. Phys. **205** (2005), no. 1, 1–23.

[7] Zhifeng Sheng, Patrick Dewilde, and Shivkumar Chandrasekaran, *Algorithms to solve hierarchically semi-separable systems*, System theory, the Schur algorithm and multidimensional analysis, Oper. Theory Adv. Appl., vol. 176, Birkhäuser, Basel, 2007, pp. 255–294. MR MR2342902

[8] J. Xia, S. Chandrasekaran, M. Gu, and X.S. Li, *Fast algorithms for hierarchically semiseparable matrices*, Numerical Linear Algebra with Applications **17** (2010), no. 6, 953–976.

[9] Jianlin Xia, Shivkumar Chandrasekaran, Ming Gu, and Xiaoye S. Li, *Superfast multifrontal method for large structured linear systems of equations*, SIAM J. Matrix Anal. Appl. **31** (2010), no. 3, 1382–1411. MR 2587783 (2011c:65072)